Developing MapReduce Application

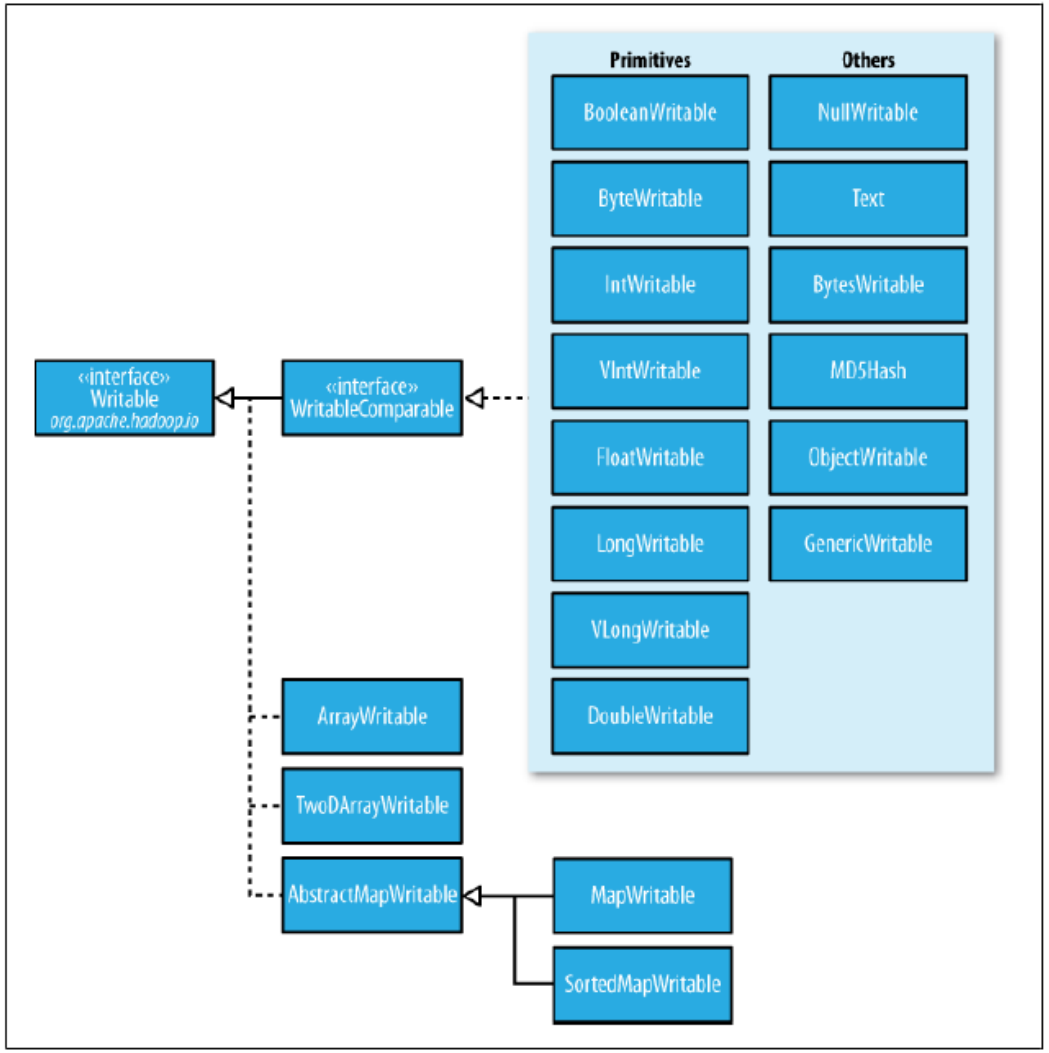
Contents

Data Types

The “**Writable**” interface makes serialization quick and easy. Any Value (object) type must implement Writable Interface.

A “**Writable Comparable**” is a Writable which is also Comparable. Any Key (object) must implement Writable Comparable Interface.

Java primitive	Writable implementation	Serialized size (bytes)
boolean	BooleanWritable	1
byte	ByteWritable	1
short	ShortWritable	2
int	IntWritable	4
	VIntWritable	1–5
float	FloatWritable	4
long	LongWritable	8
	VLongWritable	1–9
double	DoubleWritable	8



File Formats

- **File Input Format**
  - The base class used for all file-based Input Formats
- **Text Input Format**
  - The default
  - Treats each \n-terminated line of a file as a value
  - Key is the byte offset within the file of that line
- **Key Value Text Input Format**
  - Maps \n-terminated lines as ‘key SEP value’
  - By default, separator is a tab
- **Sequence File Input Format**
  - Binary file of (key, value) pairs with some additional metadata
- **Sequence File As Text Input Format**
  - Similar, but maps (key.toString(), value.toString())

Driver Code

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

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool
{

   @Override
   public int run(String[] args) throws Exception {

     if (args.length != 2) {
       System.out.printf(
           "Usage: %s [generic options] <input dir>
<output dir>\n", getClass()
                   .getSimpleName());
       ToolRunner.printGenericCommandUsage(System.out);
       return -1;
     }

     Job job = new Job(getConf());
     job.setJarByClass(WCDriver.class);
     job.setJobName(this.getClass().getName());
```

Driver Code contd…

```
    FileInputFormat.setInputPaths(job, new
Path(args[0]));
    FileOutputFormat.setOutputPath(job, new
Path(args[1]));

    job.setMapperClass(WCMapper.class);
    job.setReducerClass(WCReducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    if (job.waitForCompletion(true)) {
      return 0;
    }
    return 1;
  }

  public static void main(String[] args) throws Exception
{
    int exitCode = ToolRunner.run(new WCDriver(), args);
    System.exit(exitCode);
  }
}
```

Mapper Code

```
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 WCMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
  @Override
  public void map(LongWritable key, Text value, Context
context)
      throws IOException, InterruptedException {
    String s = value.toString();
    for (String word : s.split("\\W+")) {
      if (word.length() > 0) {
        context.write(new Text(word), new IntWritable(1));
      }
    }
  }
}
```

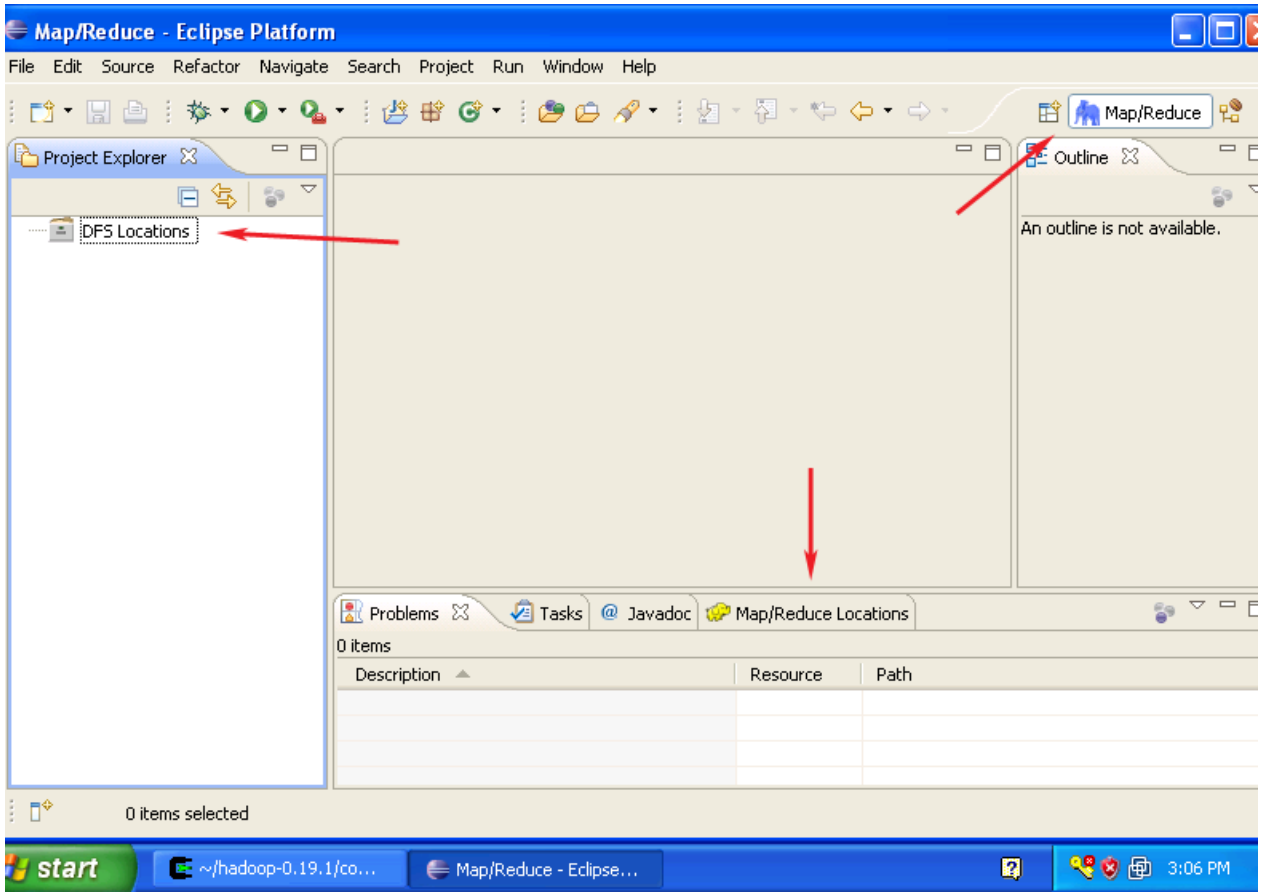
Reducer Code

```
import java.io.IOException;

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

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

  @Override
  public void reduce(Text key, Iterable<IntWritable>
values, Context context)
      throws IOException, InterruptedException {
    int wordCount = 0;
    for (IntWritable value : values) {
      wordCount += value.get();
    }
    context.write(key, new IntWritable(wordCount));
  }
}
```

Using Eclipse

http://wiki.apache.org/hadoop/EclipsePlugIn

- Download eclipse plugin for windows
- Copy the plugin to plugin folder.
- Start Eclipse
- Click on the open perspective icon ,which is usually located in the upper-right corner the eclipse application. Then select **Other** from the menu.
- Select **Map/Reduce** from the list of perspectives and press "OK" button
- As a result your IDE should open a new perspective that looks similar to the image below.
- You can configure the cluster connection.



Running Locally

- Run Locally during development phase for faster execution times.
- Once Tested locally, Run on cluster.
- The biggest difference between Local /cluster is local cannot run more than one reducer, even if you have set multiple reducers.

Command to execute:

$ hadoop jar WordCount.jar WCDriver -fs file:/// -jt local Input Output

-fs file:/// - Use local file system instead of HDFS

-jt local - Use Local Job Runner

Running on Cluster

Command to execute:

$ hadoop jar WordCount.jar WCDriver input output

- Input and output must be files or directories in HDFS only
- Before starting job, Job ID is printed.
- Once job is completed, Job Counters, FileSystem Counters and Map-Reduce framework info is printed
- Job ID: job_201209280931_0002 means the job has started on 2012 Oct 28th at 09:31. 002 means it's a second job run by job tracker (Job IDs starts with 0001).
- Task ID: task_201209280931_0002_m_000001 means the it's a second map task (Task IDs starts with 000000).
- Task Attempt: task_201209282149_0002_m_000002_0 (First Attempt)

Next Session

How Shuffle/Sort Works?