Developing MapReduce Application

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Agenda

- Data Types
- File Formats
- Driver, Mapper & Reducer Code
- Using Eclipse
- Writing Unit Tests
- Running Locally
- Running on Cluster
- Old API vs New API

Cloudwick Technologies

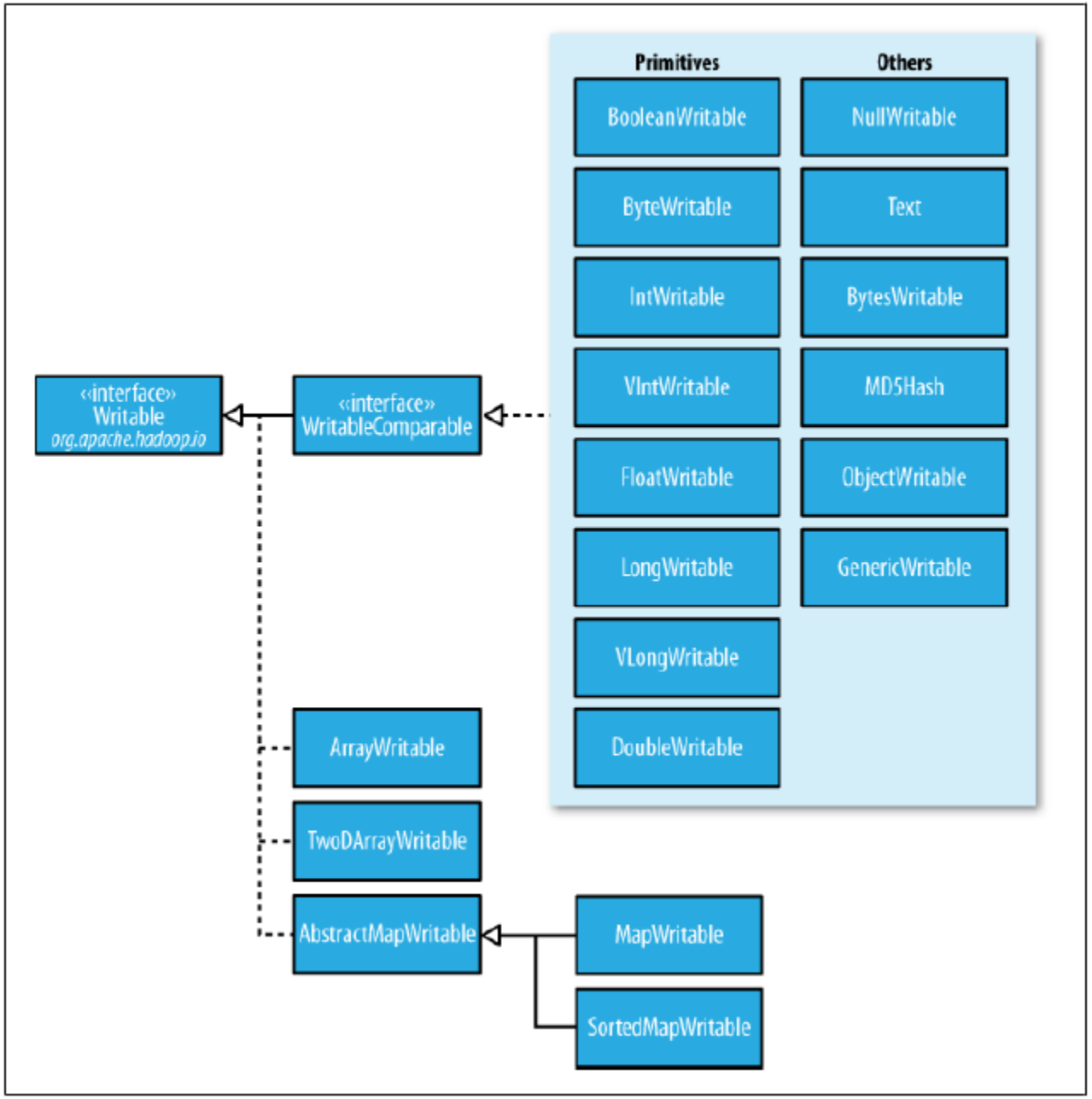
Copyright ©2012 Cloudwick Technologies

Data Types

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

A "WritableComparable" is a Writable which is also Comparable. Any Key (object) must implement WritableComparable 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



Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

File Formats

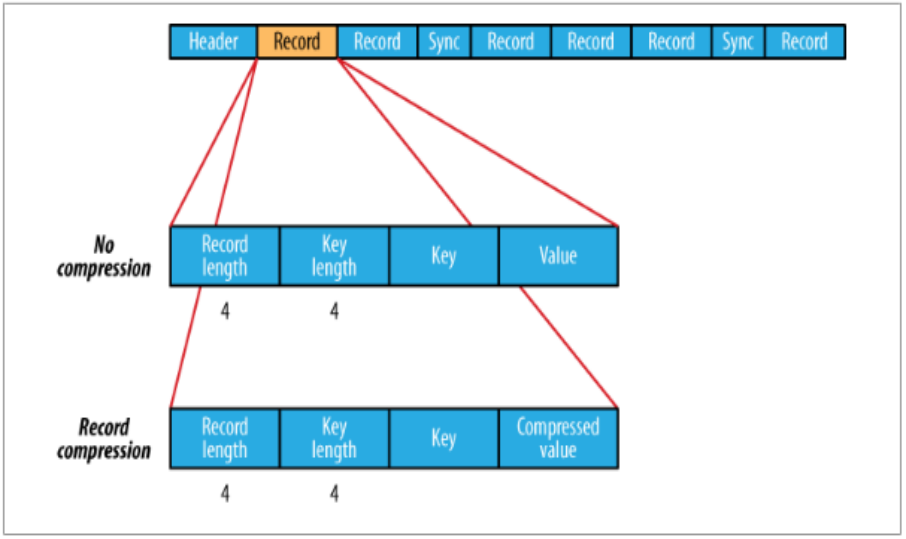
- **FileInputFormat**
  - The base class used for all file-based InputFormats
- **TextInputFormat**
  - The default
  - Treats each \n-terminated line of a file as a value
  - Key is the byte offset within the file of that line
- **KeyValueTextInputFormat**
  - Maps \n-terminated lines as 'key SEP value'
    - By default, separator is a tab
- **SequenceFileInputFormat**
  - Binary file of (key, value) pairs with some additional metadata
- **SequenceFileAsTextInputFormat**
  - Similar, but maps (key.toString(), value.toString())

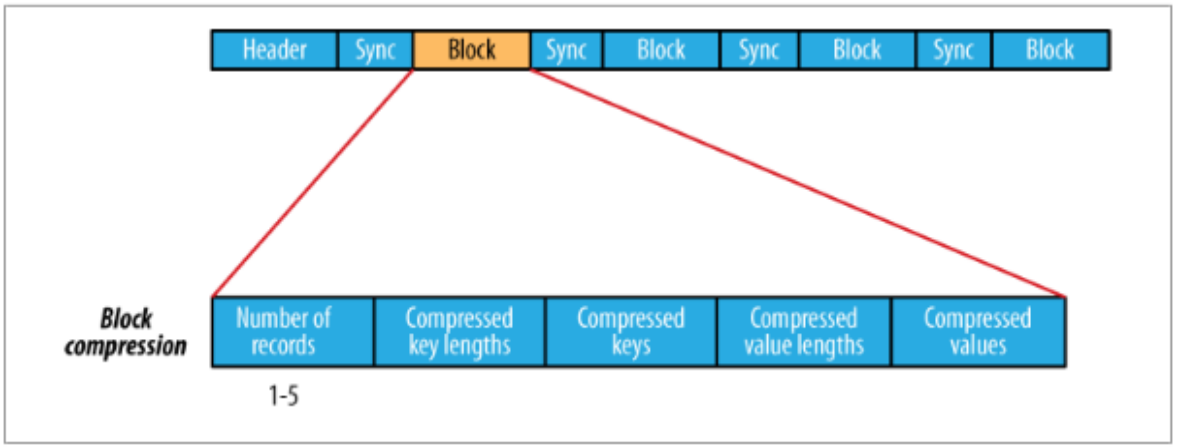
Copyright ©2012 Cloudwick Technologies

Sequence and Map Files

SequenceFile is a flat file consisting of binary key/value pairs. internally, the temporary outputs of maps are stored using SequenceFile.

- Splittable. So they work well with MapReduce: each map gets an independent split to work on.
- Compressible. By using block compression you get the benefits of compression (use less disk space, faster to read and write), while keeping the file splittable still.
- Compact. SequenceFiles are usually used with Hadoop Writable objects, which have a pretty compact format.
- A MapFile is an indexed SequenceFile, useful for if you want to do look-ups by key.





Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

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());
```

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Driver Code - Continued

```
    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);
  }
}
```

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

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));
      }
    }
  }
}
```


Copyright ©2012 Cloudwick Technologies

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));
  }
}
```

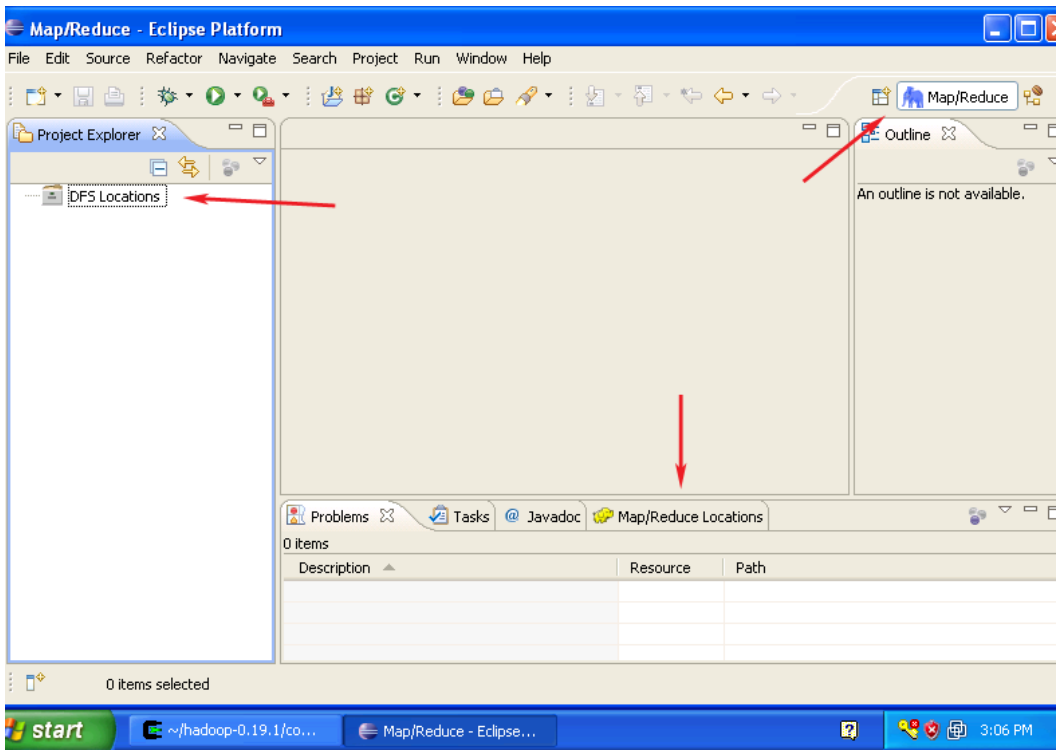
Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Using Eclipse

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

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



Copyright ©2012 Cloudwick Technologies

Unit Tests

- **Unit Tests helps in faster development**
- **MRUnit which is built on top of Junit is used for Unit Testing.**
- **It can be used to test, Mapper, Reducer or entire M/R code**
- **@Test Annotation tells that the method is Test method.**
- **@Before @ After executes after every @Test method.**
- **MRUnit throws a run time exception when a test fails.**

```
@After
public void tearDown() throws Exception {
}
@Test
public void testMapper() {
 mapDriver.withInput(new LongWritable(1), new
Text("1 XYZ 1950"));
 //expected output from the mapper
 mapDriver.withOutput(new Text("1950"), new
Text("XYZ"));
 //run the test
 mapDriver.runTest();
}
@Test
public void testReducer(){
 List<Text> values = new ArrayList<Text>();
 values.add(new Text("XYZ"));
 values.add(new Text("ABC"));
 reduceDriver.withInput(new Text("1950"), values);
 reduceDriver.withOutput(new Text("1950"), new
IntWritable(2));
 reduceDriver.runTest();
}
```

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Hadoop Streaming

- Hadoop provides an API to MapReduce that allows you to write your map and reduce functions in languages other than Java. Hadoop Streaming uses Unix standard streams as the interface between Hadoop and your program, so you can use any language that can read standard input and write to standard output to write your MapReduce program.
- Map input data is passed over standard input to your map function, which processes
- It line by line and writes lines to standard output.
- A map output key-value pair is written as a single tab-delimited line.
- Input to the reduce function is in the same format—a tab-separated key-value pair—passed over standard input.
- The reduce function reads lines from standard input, which the framework guarantees are sorted by key, and writes its results to standard output.

```
% hadoop jar $HADOOP_INSTALL/contrib/streaming/hadoop-*-streaming.jar \
  -input input/ncdc/sample.txt \
  -output output \
  -mapper ch02/src/main/ruby/max_temperature_map.rb \
  -reducer ch02/src/main/ruby/max_temperature_reduce.rb
```

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

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

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Running on Cluster

Command to execute:

$ hadoop jar WordCount.jar WCDriver /input /output

- 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)

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

New API

- **When Hadoop 0.20 was released, a 'New API' was introduced**
  - Designed to make the API easier to evolve in the future
  - Favors abstract classes over interfaces
- **The 'Old API' was deprecated**
- **However, the New API is still not absolutely feature-complete in Hadoop 0.20.x**
  - The Old API should not have been deprecated as quickly as it was
- **Most developers still use the Old API**

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Old API vs New API

Old API	New API
`import org.apache.hadoop.mapred.*`	`import org.apache.hadoop.mapreduce.*`

Driver code:
 	
`JobConf conf = new JobConf(conf, Driver.class);`
`conf.setSomeProperty(...);`
`...`
`JobClient.runJob(conf);`
	Driver code:
	
	`Configuration conf = new Configuration;`
	`Job job = new Job(conf);`
	`job.setJarByClass(Driver.class);`
	`job.setSomeProperty(...);`
	`...`
	`job.waitForCompletion(true);`

Mapper:
 	
`public class MyMapper extends MapReduceBase implements Mapper {`
 	
`public void map(Keytype k, Valuetype v, OutputCollector o, Reporter r) {`
`...`
`o.collect(key, val);`
`}`
 	
`}`
	Mapper:
	
	`public class MyMapper extends Mapper {`
	
	`public void map(Keytype k, Valuetype v, Context c) {`
	`...`
	`c.write(key, val);`
	`}`
	
	`}`

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Old API vs New API - Cont'd

Old API	New API
`import org.apache.hadoop.mapred.*`	`import org.apache.hadoop.mapreduce.*`

Driver code:
 	
`JobConf conf = new JobConf(conf, Driver.class);`
`conf.setSomeProperty(...);`
`...`
`JobClient.runJob(conf);`
	Driver code:
	
	`Configuration conf = new Configuration;`
	`Job job = new Job(conf);`
	`job.setJarByClass(Driver.class);`
	`job.setSomeProperty(...);`
	`...`
	`job.waitForCompletion(true);`

Mapper:
 	
`public class MyMapper extends MapReduceBase implements Mapper {`
 	
`public void map(Keytype k, Valuetype v, OutputCollector o, Reporter r) {`
`...`
`o.collect(key, val);`
`}`
 	
`}`
	Mapper:
	
	`public class MyMapper extends Mapper {`
	
	`public void map(Keytype k, Valuetype v, Context c) {`
	`...`
	`c.write(key, val);`
	`}`
	
	`}`

Cloudwick Technologies

Copyright ©2012 Cloudwick Technologies

Questions?