Leopold–Franzens–University
Innsbruck

Institute of Computer Science
Distributed and Parallel Systems

Apache Hadoop Karwendel Tutorial

Advanced Cloud and Grid Technologies (WS 2011/2012)

Supervisor: Dr. Radu Prodan

Martin Illecker

Innsbruck, January 17, 2012

Apache Hadoop Karwendel Tutorial

Martin Illecker

martin.illecker@student.uibk.ac.at

1 Introduction

This tutorial gives a short overview how you can run Hadoop on karwendel.dps.uibk.ac.at.
http://hadoop.illecker.at/Tutorial.pdf

2 Hadoop Java Versions

Hadoop requires Java 1.6+. It is built and tested on Oracle Java, which is the only "supported" JVM.
For further informations see http://wiki.apache.org/hadoop/HadoopJavaVersions.

2.1 Java - karwendel.dps.uibk.ac.at

You can find a suitable java version here:

```
$ /software/java/jdk1.6.0_20-x64/bin/java -version
java version "1.6.0_20"
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) 64-Bit Server VM (build 16.3-b01, mixed mode)
```

3 Downloading Hadoop System Package

First of all you should download this package to your home folder on karwendel and extract it!

```
$ wget http://hadoop.illecker.at/hadoop_system.tar.gz
$ tar xzvf hadoop_system.tar.gz
```

4 Customisation for karwendel.dps.uibk.ac.at

Then you have to modify the start script *hadoop* in your home folder. The *cog kit* lib path (here ~/cog-4_1_5/lib/*) should be adjusted to your path. *hadoop_system* uses the *cog kit* to submit slaves to the grid.

```
#!/bin/sh
# First argument: HDFS Port (default: 50001)
# First argument: JobTracker Port (default: 50002)
export JAVA_HOME=/software/java/jdk1.6.0_13-x86_64_sun
cd hadoop_system
$JAVA_HOME/bin/javac -cp ~/cog-4_1_5/lib/*:. Main.java Slave.java Config.java
$JAVA_HOME/bin/java -cp ~/cog-4_1_5/lib/*:. Main $1 $2
```

5 Hadoop Configuration Master

The master configuration files are located in `hadoop_system/hadoop-1.0.0/etc/hadoop`.

There exists three basic config files:

1. **core-site.xml**
2. **hdfs-site.xml**
3. **mapred-site.xml**

These config files are already configured and will be updated by the hadoop_system tool. So normally there is no need to change something!

5.1 core-site.xml

You can find all possible properties here `http://hadoop.apache.org/common/docs/current/core-default.html`.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.71.100:50001</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri's scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri's authority is used to
    determine the host, port, etc. for a filesystem.</description>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/lab406/hadoop_system/hadoop-1.0.0/tmp</value>
    <description>A base for other temporary directories.</description>
  </property>
</configuration>
```

5.2 hdfs-site.xml

You can find all possible properties here `http://hadoop.apache.org/common/docs/current/hdfs-default.html`.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
   </property>
   <property>
     <name>dfs.http.address</name>
     <value>0.0.0.0:0</value>
     <description>The address and the base port where the dfs namenode
     web ui will listen on. If the port is 0 then the server will start
     on a free port.</description>
   </property>
</configuration>
```

5.3 mapred-site.xml

You can find all possible properties here `http://hadoop.apache.org/common/docs/current/mapred-default.html`.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>192.168.71.100:50002</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
  </property>
  <property>
    <name>mapred.job.tracker.http.address</name>
    <value>0.0.0.0:0</value>
    <description>The job tracker http server address and port the server
    will listen on. If the port is 0 then the server will start on a free port.
    </description>
  </property>
  <property>
```

```
    <name>mapred.task.tracker.http.address</name>
    <value>0.0.0.0:0</value>
    <description>The task tracker http server address and port.
    If the port is 0 then the server will start on a free port.
    </description>
  </property>
</configuration>
```

6 Hadoop Configuration Slave

The slave configuration files are located in `hadoop_system/hadoop-1.0.0/etc/hadoop_slave`. These config files are also already configured and there is no need to change something!

6.1 core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.71.100:50001</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri's scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri's authority is used to
    determine the host, port, etc. for a filesystem.</description>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/lab406/hadoop_system/hadoop-1.0.0/tmp</value>
    <description>A base for other temporary directories.</description>
  </property>
</configuration>
```

6.2 hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
   </property>
   <property>
```

```
    <name>dfs.http.address</name>
    <value>0.0.0.0:0</value>
    <description>The address and the base port where the dfs namenode
    web ui will listen on. If the port is 0 then the server will start
    on a free port.</description>
  </property>
</configuration>
```

6.3 mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>192.168.71.100:50002</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
  </property>
  <property>
    <name>mapred.job.tracker.http.address</name>
    <value>0.0.0.0:0</value>
    <description>The job tracker http server address and port the server
    will listen on. If the port is 0 then the server will start on a free port.
    </description>
  </property>
  <property>
    <name>mapred.task.tracker.http.address</name>
    <value>0.0.0.0:0</value>
    <description>The task tracker http server address and port.
    If the port is 0 then the server will start on a free port.
    </description>
  </property>
</configuration>
```

7 HDFS Format

Before you can format your HDFS you have to run `./hadoop` once, this will update your paths in the config files. After that you can delete old settings and format the HDFS by executing the following commands.

```
$ ./hadoop
$ rm -R ./hadoop_system/hadoop-1.0.0/tmp/*
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop namenode -format
```

8 Hadoop System

You can control the hadoop system by running the start script *hadoop* in your home folder. You should set two parameters, because if some other student has running this system on the same port, hadoop wouldn't start. Ports from 50000 to 50100 are open.

```
$ ./hadoop 50001 50002
HadoopSystem started @ /home/lab406/hadoop_system
HadoopSystem config complete.
User: lab406
HDFS: hdfs//karwendel.dps.uibk.ac.at:50001
JobTracker: karwendel.dps.uibk.ac.at:50002
TempDir: /home/lab406/hadoop_system/hadoop-1.0.0/tmp
============================
|   HADOOP MENU SELECTION  |
============================
| Options:                 |
|  0. Exit                 |
|  1. Start Master         |
|  2. Stop  Master         |
|  3. Master Web PORTS     |
|  4. Start Slave          |
|  You have to kill Slaves |
|  by running qdel         |
============================
Select option:
```

1. Option 0 to exit. Attention: If you close this tool all daemons stay running! You have to select option 2 and `qdel` slaves

2. Option 1 starts all hadoop daemons

3. Option 2 stops all hadoop daemons

4. Option 3 prints the current port numbers of the web interfaces (JobTracker and NameNode). The port number for the JobTracker is updated some seconds later because the NameNode can be in safe node at the beginning.

5. Option 4 submits one slave, you can see the current slaves in the JobTracker Webinterface

In this first release of this hadoop_system tool you have to kill the submitted slaves manually. This is done by `qdel` with the jobid from the `qstat`. I'm sorry for this inconvenience.

9 Example Application

If you have chosen option 1 to start the master you can finally test your hadoop system by submitting a well known wordcount job. This example downloads some books from gutenberg, submit these text files to the hadoop distributed filesystem and finally starts a map reduce job. The following commands are executed in the home directory of karwendel.

```
$ mkdir gutenberg
$ cd gutenberg
$ wget http://www.gutenberg.org/cache/epub/20417/pg20417.txt
$ wget http://www.gutenberg.org/cache/epub/5000/pg5000.txt
$ wget http://www.gutenberg.org/cache/epub/4300/pg4300.txt
$ cd ..
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -copyFromLocal gutenberg /example/gutenberg
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -ls /example/gutenberg
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop jar
        ./hadoop_system/hadoop-1.0.0/bin/hadoop-examples-1.0.0.jar wordcount
        /example/gutenberg /example/gutenberg-output
```

9.1 Word Count Example

You can also download this Source from `http://hadoop.illecker.at/WordCount.java`

WordCount.java

```
1  import java.io.IOException;
2  import java.util.*;
3
4  import org.apache.hadoop.fs.Path;
5  import org.apache.hadoop.conf.*;
6  import org.apache.hadoop.io.*;
7  import org.apache.hadoop.mapred.*;
8  import org.apache.hadoop.util.*;
9
10 public class WordCount {
11
12   public static class Map extends MapReduceBase implements
13         Mapper<LongWritable, Text, Text, IntWritable> {
14
15     private final static IntWritable one = new IntWritable(1);
16     private Text word = new Text();
17
18     public void map(LongWritable key, Text value,
19         OutputCollector<Text, IntWritable> output, Reporter reporter)
20         throws IOException {
```

```
21
22                  String line = value.toString();
23                  StringTokenizer tokenizer = new StringTokenizer(line);
24                  while (tokenizer.hasMoreTokens()) {
25                          word.set(tokenizer.nextToken());
26                          output.collect(word, one);
27                  }
28      }
29  }
30
31  public static class Reduce extends MapReduceBase implements
32          Reducer<Text, IntWritable, Text, IntWritable> {
33
34      public void reduce(Text key, Iterator<IntWritable> values,
35          OutputCollector<Text, IntWritable> output, Reporter reporter)
36          throws IOException {
37
38                  int sum = 0;
39                  while (values.hasNext()) {
40                          sum += values.next().get();
41                  }
42                  output.collect(key, new IntWritable(sum));
43          }
44  }
45
46  public static void main(String[] args) throws Exception {
47          JobConf conf = new JobConf(WordCount.class);
48          conf.setJobName("wordcount");
49
50          conf.setOutputKeyClass(Text.class);
51          conf.setOutputValueClass(IntWritable.class);
52
53          conf.setMapperClass(Map.class);
54          conf.setCombinerClass(Reduce.class);
55          conf.setReducerClass(Reduce.class);
56
57          conf.setInputFormat(TextInputFormat.class);
58          conf.setOutputFormat(TextOutputFormat.class);
59
60          FileInputFormat.setInputPaths(conf, new Path(args[0]));
61          FileOutputFormat.setOutputPath(conf, new Path(args[1]));
62
63          JobClient.runJob(conf);
64  }
65 }
```

You need the hadoop-core library to compile the WordCount.java example.

```
$ wget http://mirror.sti2.at/apache//hadoop/common/hadoop-1.0.0/hadoop-1.0.0.tar.gz
$ tar xzvf hadoop-1.0.0.tar.gz
```

Then you can compile the WordCount.java source:

```
$ mkdir wordcount_classes
$ /software/java/jdk1.6.0_13-x86_64_sun/bin/javac -cp ./hadoop-1.0.0/hadoop-core-1.0.0.jar
-d wordcount_classes WordCount.java
$ jar -cvf wordcount.jar -C wordcount_classes/ .
```

These files need to be in HDFS:

```
$ echo "Hello World Bye World" > file01
$ echo "Hello Hadoop Goodbye Hadoop" > file02
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -mkdir /example/wordcount/input
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -copyFromLocal file0*
/example/wordcount/input
$ rm file01 file02
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -ls /example/wordcount/input/
/example/wordcount/input/file01
/example/wordcount/input/file02
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -cat /example/wordcount/input/file01
Hello World Bye World
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -cat /example/wordcount/input/file02
Hello Hadoop Goodbye Hadoop
```

Run the application:

```
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop jar wordcount.jar WordCount
/example/wordcount/input /example/wordcount/output
```

Output:

```
$ ./hadoop_system/hadoop-1.0.0/bin/hadoop dfs -cat /example/wordcount/output/part-00000
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```