tutorial_2_hadoop_mr_build

Tutorial 2 - Hadoop MapReduce

Access the Cluster

In order to access the DIMA cluster, open a shell and establish a SSH session with the following command (instead of insert the username you received in class):

```
ssh <user>@ibm-power-1.dima.tu-berlin.de -L 8088:ibm-power-1.dima.tu-berlin.de:8088
```

Input the password you received in class to establish a SSH connection and the port 8088 will be forwarded to your localhost (we will shortly see why this is important). *ibm-power-1.dima.tu-berlin.de* is the DNS name of the first host in the cluster, consisting of 10 nodes. This node is running the HDFS namenode, the secondary namenode and the MapReduce job tracker.

To check the status and other information regarding the Hadoop cluster, open your browser and go to http://localhost:8088 (). The page gives an overview over the cluster status, submitted, running and finished jobs and scheduler related information.

Running a MapReduce Job

The Hadoop MapReduce framework expects the user program to be packaged into a *.jar* file containing the code for the mapper, reducer and any additional dependencies. This is then deployed to all nodes of the cluster during execution. For this purpose the framework provides a distinct application, *hadoop*, to submit a jar file to the cluster. The application expects the path to the jar file as an argument, followed by the main class (if not specified in the manifest) and any additional arguments used in the MapReduce job.

```
hadoop jar <path to jar> <name of main class> [additional arguments]
```

Lets assume we wrote a MapReduce job to count the number of words in all files of a given HDFS directory and writing the result to a different output directory. The package structure is de.tu-berlin.dima and the main class is named WordCount. The command to submit the job to the cluster would look like the following:

```
hadoop jar wordcount.jar de.tu-berlin.dima.WordCount inputDir outputDir
```

Creating a JAR file

With Maven

Without IDE

In order to build the project, just clone the repository and change the directory to the exercise you want to build. The hadoop dependencies are already specified inside the *pom.xml* and will be downloaded as soon as the jar file is created by:

```
mvn package
```

IntelliJ Idea

In order to import the project into IntelliJ, open IntelliJ Idea and **select import**. In the following dialog **select the directory** of this repository and continue with the next dialog, where you need to **select Maven** then continue with next and select the artifacts as well as the Java SDK. After finishing the import, open **Maven Projects** (bottom left) and go to *Lifecycle*. Use **package** to generate the artifacts, in this case the jar file contains all MapReduce jobs for the specific exercise.

Without Maven (not recommended for larger projects)

Creating a MapReduce job jar file requires some hadoop libraries, which are part of any release and can be downloaded at http://apache.mirror.iphh.net/hadoop/common/ (). Unpack the tar file and create an environment variable (in this example *$HADOOP_HOME*) pointing to the path.

Assuming you've already created a java file for the wordcount MapReduce job inside the directory *de/tu-berlin/dima/* named *WordCount.java*. The following commands create a jar file from the source file(s). When invoking the Java compiler also the hadoop libraries must be added to the classpath in order to ensure the files can be found. Additionally the output directory for the class files and the input file(s) must be specified.

```
javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.2.0.jar:$HADOOP_HOME/share/h
```

Afterwards the jar file can be packed by:

```
jar -cvf wordcount.jar -C wordcount_jar_dir/
```

Last edited by Christoph Alt 2 months ago