Lab 4 – Hello Hadoop

CC5212-1

April 9, 2015

We're going to get a simple word count up and running on Hadoop (an open-source implementation of the MapReduce distributed processing abstraction). The core goal in this lab is for you to learn how to interact with HDFS (the Hadoop Distributed File System), upload some data, code a basic Hadoop job in Java, launch it on the cluster, and get the results back.

These instructions are for Windows. But if you prefer Linux **and** are familiar with commands like `ssh` and `scp`/`sftp`, feel free to use Linux.

- First we want to SSH onto the server:
  - Head to `http://aidanhogan.com/teaching/cc5212-1/tools` and download the tools found there. These are standalone Windows tools for SSH and SFTP/SCP.
  - Open PuTTy. In host-name type `cluster.dcc.uchile.cl`. Click Save. Click Open.
  - If the above doesn't work just try `cluster` or the IP: `172.17.69.99`.
  - Username and password will be provided on the board.
  - You are now on the master server of a five-server cluster.
  - Since you are all using one username, please, please be considerate. **Think twice before typing `rm -r`, `kill` and similar commands.** Please.
- Next we want to look at the distributed file server and upload some data to it:
  - Type `hdfs dfs`.[1] This shows you the options to interact with HDFS.
  - Type `hdfs dfs -ls /` to see the root contents. These files are stored across the servers.
  - Next create a sub-folder of `/uhadoop` for your stuff. Type `hdfs dfs -ls /uhadoop` to see the contents. Then think up a user-name (use first letter of first name, full last name: e.g., `ahogan` for me). Type `hdfs dfs -mkdir /uhadoop/[username]` replacing `[username]` with your username, e.g., `hdfs dfs -mkdir /uhadoop/ahogan`.[2] This is now your personal folder ... please keep your files on the DFS in this folder. Also beware that anyone can look at your data. Don't upload sensitive medical records.
  - Now on the local file system, go to the directory `cd /data/2014/uhadoop/`. Make a directory for yourself `mkdir /data/2014/uhadoop/[username]` with the same username as before.
  - Go to directory `cd /data/2014/uhadoop/shared`. Here you'll find the file we want to do a word-count on stored locally. You need to get that onto DFS. Type `hdfs dfs -copyFromLocal /data/2014/uhadoop/shared/es-abstracts.txt /uhadoop/[username]/es-abstracts.txt`
  - Check that it's there now: type `hdfs dfs -ls /uhadoop/[username]`.
- Now that we've gotten the data on the DFS, we need to code the Hadoop job to do the word count. Since this is your first time coding a Hadoop job, you can follow the slides from Monday's lecture as "inspiration". Grab them from the homepage.

[1] One can also type `hadoop fs` but it seems `hdfs fs` is preferred.

[2] Yes, people last year did create a folder literally called `username`, hence the careful instructions this time. If you wish to honour that tradition, feel free! But afterwards create a meaningful folder as well and use that.

  - Download the project from `http://aidanhogan.com/teaching/cc5212-1/code/mdp-lab4.zip` and open it in Eclipse.
  - Open up the `WordCount` class. Here we're going to put the mapper class, the reducer class and the main method.
  - To start with, implement the `WordCountMapper` class. This class decides how the data should be split across machines. You will find a very similar example in Monday's slides (2. Map). (In this case, you include all the classes into one class file, but the code is otherwise very similar. To parse the input line, use `String[] rawWords = value.toString().split(SPLIT_REGEX);`. Make sure the words are not empty and to lowercase each one. For each word in that array, output the word with a value containing 1.)
  - We don't need to do anything special with comparisons, sorts, partitioning, etc. All the default settings will work fine for us so we can skip to ...
  - ... implementing the `WordCountReducer`. Again, the example in Monday's slides (6. Reduce) is *nearly* identical to what you need.
  - Finally we need a main method that sets the job configuration and the control flow. Again, you will find a *near*-identical version in Monday's slides (Control Flow). The first argument should be then In this case, can we use `WordCountReducer` as a combiner?

- Finally we need to package and run the code.
  - In the project in Eclipse, right click on `build.xml`, then `Run As ...`, then make sure `dist` is clicked and hit `Run`. If it fails, you may need to manually make a new `dist` folder in the project root. Refresh the project (F5) and make sure you have the JAR file.
    - If you get an error mentioning javac1.8 or similar, you need to right click on the `build.xml`, go to External Tools Configurations and add the line `-Dbuild.compiler=javac1.7` to arguments.
    - If you get a new error saying something about `JAVA_HOME` not found, follow: `Window > Preferences >` Expand `Ant` tree `> Runtime > Highlight Global Entries > Add External Jars >` Then find and add `tools.jar` in the `lib/` folder of (e.g., `C:/Program Files (x86)/Java/jdk.../lib/tools.jar`).
  - We need to copy the JAR file to the server. Head to WinSCP. Set `SFTP`. For the hostname, enter `cluster.dcc.uchile.cl`. Enter `uhadoop` as username. Don't enter the password yet. Click save. When it prompts, enter same password as for `PuTTy`. Copy your `.jar` file into `/data/2014/uhadoop/[username]`.
  - Now we just need to call the Hadoop job. Go back to PuTTy. Run (all one command): `hadoop jar /data/2014/uhadoop/[username]/mdp-lab4.jar WordCount /uhadoop/[username]/es-abstracts.txt /uhadoop/[username]/wc/`.[3] Hopefully you will see the Map/Reduce progress as it happens. Likewise you should see the results of the counter.
  - When it's finished, it's time to look at the results.
    - Run `hadoop fs -ls /uhadoop/[username]/wc/` to see the output.
    - Run `hadoop fs -cat /uhadoop/[username]/wc/part-0000 | more` to look into the file.
    - Try find the count for "de": run `hdfs dfs -cat /uhadoop/[username]/wc/part-00000 | grep -P "^de\t" | more`. Did you get 4,575,144?

- OPTIONAL: now try code a second MapReduce job to sort the words by occurrence. ☺

- Submit the `WordCount` class to u-cursos before the lecture on Monday.

[3] The output directory must *not* exist prior to running the job!