COMP38311: MapReduce Lab

This document outlines the specifications for the Advanced Distributed Systems laboratory work, which gives some practical experience with MapReduce. This lab work is not assessed, but MapReduce will feature in the exam.

The coursework will be supported by a laboratory session (in which there will an academic and Graduate Teaching Assistants present), and questions can be emailed to norman.paton@manchester.ac.uk.

You are welcome to work in pairs on this lab. You are advised to use the lab machines and not your laptop; installing hadoop is not something that always goes smoothly, and we can't support you with that on arbitrary platforms. Hadoop is installed on the lab machines.

Getting Started

There are several steps to go through to get started in this lab. A video in the *Week 8* folder on Blackboard also walks through the steps involved in *Running Hadoop on Linux*. Note that the paths in the video are not the same as the paths needed for the lab computers. Note also that some of the setup below will already have been done on the lab machines, but the full details are here for anyone who tries on their own machine.

1. *Download the projects*. The lab needs two Java projects that are available from the *Week 8* folder on Blackboard, for the WordCount and InvertedIndex exercises.
2. *Set JAVA_HOME*. Hadoop should work with different versions of Java. Java 17 is installed on the lab machines. Here we assume you are running bash as your shell. Add the following to your ~/.bashrc file, and then start a new shell:

```
export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-amd64"
```

3. *Make the WordCount project run on the command line.*
    a. *Set the location of Hadoop*. Change into the WordCount directory. Look at build.xml. Set *hadoop-location* to the location of Hadoop on the machine you are using. For the lab machines, this is:

```
<property name="hadoop-location" value="/opt/hadoop-3.4.0/bin/" />
```

    b. Use ant to run Hadoop for the WordCount project, using build.xml. From within the WordCount folder, at the command line this is: *ant run*. When you type this, you should see a collection of log entries produced, with a final BUILD SUCCESSFUL message. If so, the result will be in the *output* folder. If not, look in the log for error messages that will hopefully tell you what to do next.
    c. Note that, in principle, you can complete the lab from the command line, without recourse to an IDE, but we describe how to get going with Eclipse below.
4. *Make the WordCount project run in Eclipse*.
    a. Start Eclipse and from the *File* menu, from *Import* > *General* import the existing WordCount project into the workspace.
    b. Configure Eclipse to use ant to run this project. From the Run menu select External Tools, and then External Tools Configurations. Select *Ant Build*. If necessary, browse to the build file (build.xml) in the workspace. Then in the tabs:
        i. In the Environment tab, set JAVA_HOME, to the value given above.

   ii. In the Resource tab, set Text file encoding to UTF-8[1].
   iii. In the *Target* tab, check *run*.
   iv. Then you can run the resulting configuration to compile and execute the Hadoop project. You may have to refresh the package explorer to see the output folder.
5. Explore the source of the WordCount program.

Exercise 1: Word Count

Aim

The aim of this exercise is to become familiar with how MapReduce jobs are written using Java and run using Hadoop. In essence, you will run an existing WordCount program using Hadoop and make several changes to it.

Learning Outcomes

A student who successfully completes this exercise will be able to

- use the software that the exercises require.
- read the control flow of a MapReduce program.
- compile and run a MapReduce job.
- make some modifications to a MapReduce program.

Summary

For this exercise, you must import the framework to be used for developing Hadoop applications into Eclipse, and build the framework using Ant. Once built, you should run the WordCount program and evaluate the output. Once you have the basic framework running, you should then make some modifications to the WordCount program to increase the cleanliness of the output and to implement a summarization pattern.

Description

You should carry out the following steps for this lab:

1. Download the WordCount MapReduce project from Blackboard.
2. Compile and run the supplied WordCount program by following the instructions above.
3. Modify the program to change the *map* operation so that it cleans up the data it is acting on, for example by removing non-alphabetic characters and converting all letters to lower case. What is the word count for *Bart*? How easy is that question to answer.
4. Implement a summarization pattern by using the reduce operation as a combiner.

[1] This was necessary in the past, but on the most recent test on lab machines the Resource tab wasn't available, and everything still worked. So, this step may not be available or necessary; fingers crossed.

Exercise 2: Inverted index in MapReduce

Aim

The aim of this exercise is to develop a MapReduce program that builds an inverted index. An inverted index, given a collection of documents, supports lookups of the form term → documentId. Web indexes are inverted indexes.

Learning Outcomes

A student who successfully completes this exercise will have:

- designed the map and reduce operations for an inverted index.
- understood how to integrate these into a Java template for use with Hadoop.
- run the resulting application using Hadoop.
- implemented a mapper that uses a filtering pattern.
- implement a mapper that uses a summarisation pattern.

Summary

For this exercise you import the framework that will be the starting point for your development into Eclipse, and build it using Ant. Once built, you should complete the inverted index program and evaluate the output.

Description

You must write a MapReduce job that creates an inverted index from a set of the Wikipedia entries for The Simpsons episodes (supplied to you in the *input* folder). The tokens for the inverted index can be created by splitting a string on spaces.

You should carry out the following steps for this lab:

1. Download the InvertedIndex MapReduce project from Blackboard.
2. From Eclipse, compile and run the supplied inverted index template program in the same way as you compiled and ran WordCount in Exercise 1.
3. Design the map and reduce operations for building an inverted index. The suggestion here is that you write them first using pseudo-code, as in the workshop, and then convert this pseudo-code to Java.
    - Hint 1: the comments for the map and reduce operations give an insight as to their expected complexity for this exercise.
    - Hint 2: various tasks that need to be undertaken here have analogues in WordCount.
    - Hint 3: look at all the operations you have been provided with in the template, as some contain code that does things for you that most likely you do not know how to do!
4. Implement a basic MapReduce inverted index program, compile it, and run it over the datasets from Wikipedia that have been provided. The output should look something like:

    ability *[Bart_the_Murderer.txt.gz]*

    able *[Bart_the_Genius.txt.gz, Bart_the_Murderer.txt.gz]*

    about *[Bart_the_Fink.txt.gz, Bart_the_General.txt.gz, Bart_the_Genius.txt.gz, Bart_the_Lover.txt.gz, Bart_the_Mother.txt.gz, Bart_the_Murderer.txt.gz]*

    above *[Bart_the_Genius.txt.gz]*

…

5. Implement a filtering pattern that removes stop words. Note that we provide a stop word analyser that you can use.
6. Implement a summarization pattern that avoids writing the same documentId for a token from a mapper more than once.