CS 6240: Homework 1

Goals: (1) set up Hadoop on your development machine, (2) become familiar with AWS, and (3) start working with MapReduce.

This homework is to be completed individually (i.e., no teams). You have to create all deliverables yourself from scratch. In particular, it is not allowed to look at or copy someone else's code/text and modify it. (If you use publicly available code/text, you need to cite the source in your report!)

All deliverables for this HW are due as stated on Canvas. For late submissions you will lose two percentage points per hour after the deadline. This HW is worth 100 points and accounts for 10% of your overall homework score. Early work is encouraged and you will receive a 5-point bonus if you submit your solution on or before "Early Work Deadline" given by your instructor. (Notice that your total score cannot exceed 100 points, but the extra points would compensate for any deductions.) Please submit your solution through the Blackboard assignment submission system and make sure your report is a **PDF** file. Always package all your solution files, including the report, into a single ZIP file. Make sure you are using standard ZIP, i.e., the format also used by Windows archives.

Sign Up For an AWS Account

Go to Amazon Web Services (AWS) and create an account. You should be able to find out how to do this on your own. Enter the requested information and you are ready to go. Sometime soon we will hopefully receive the AWS codes worth $100 per student. We recommend that you wait for now with the AWS part of this assignment until the end of this week or until you received the code from us. However, there is no reason to delay the non-AWS parts of this assignment.

Set Up the Local Development Environment

We recommend using Linux for Hadoop development. (We had problems with Hadoop 1.* on Windows.) If your computer is a Windows machine, you can run Linux in a virtual machine. We tested Oracle VirtualBox: install VirtualBox (free) and create a virtual machine running Linux, e.g., Ubuntu (free). (If you are using a virtual machine, then you need to apply the following steps to the virtual machine.)

Download a Hadoop 1.* (e.g., 1.2.1) distribution directly from http://hadoop.apache.org/ and unzip it in your preferred directory, e.g., /usr/local. That's almost all you need to do to be able to run Hadoop code in standalone (local) mode from your IDE, e.g., Eclipse. Make sure your IDE supports development in Java. We tested Java 1.6, but 1.7 might work as well.

If you use a development environment like Eclipse, you might consider installing a plug-in for MapReduce development with Hadoop. However, trying to get a plugin to work adds another layer of complexity. Here is our simple setup for Eclipse (should be similar for other IDEs):

1. Create a standard Java project. Add the Hadoop core jar file and all needed jar files from the lib directory ($hadoop-dir/lib) as referenced libraries to the project. Many of those latter jar files might not be needed, but adding them all is fast and makes sure you are not missing any.
2. Create packages, code etc. as usual in the src subdirectory of the project.
3. If you want to "copy" example source code into your project, create a new file in the src folder and then copy-and-paste the text from the example file into the newly created file in your project. Make sure naming and package path are correct (Eclipse will complain and suggest fixes anyway).
4. You can then run and debug Hadoop programs directly from Eclipse. In particular, you can use the debugging perspective and step through program execution in Hadoop like in any standard Java program.
5. You can export your project as a jar file and then run it from the commandline. However, for that you usually need to set a few more environment variables as explained in the textbook and Web tutorials.

Running Word Count

Find the example Word Count code in the Hadoop release you downloaded. Get it to run in the IDE on your development machine. Notice that you will need to provide an input directory containing text files and a path to an output directory (that directory should not exist). Once the program runs fine, look at it closely and see what Map and Reduce are doing. Use the debugging perspective, set breakpoints in the map and reduce functions, then experiment by stepping through the code to see how it is processing the input file. Make sure you work with a small data sample.

Export the program as a jar file and get it to run following the instructions on Amazon for setting up and running a Hadoop job using Elastic MapReduce. In short, you need to tell EMR what jar file you want to run, what its input parameters are, where the input directory is located, and where the output should go (some location on S3). Use the data file that comes with this assignment. Unzip it and upload into an S3 bucket in your Amazon account. To access the data from Elastic MapReduce, provide the bucket location, e.g., s3n://myPath/hw1.in, as the commandline parameter for the input path of your job. Before writing any MapReduce code, try to look at the content of the file. It is in plain text format.

WARNING: Leaving large data on S3 and using EMR will cost money. Read carefully what Amazon charges and estimate how much you would pay per hour of computation time before running a job. Use only the small machine instances. And remember to test your code as much as possible on your development machine. When testing on AWS, first work with small data samples.

More Efficient Way Of Using AWS

Moonyoung, who was TA for CS6240 in Spring 2013, kindly offered to share his own setup. Take a look at http://yerihyo.wikidot.com/courses:2. This should make things more efficient, but is not required for the course. We recommend you try it after succeeding with the approach described above.

Report

Write a brief report about your findings, using the following structure.

Header

This should provide information like class number, HW number, and your name.

Local Execution

Show a screenshot of your IDE window. It should contain the following information:

- Project directory structure, showing that the WordCount.java file is somewhere in the src directory. (20 points)
- The console output for a successful run of the WordCount program inside the IDE. The console output refers to the job summary information Hadoop produces, not the output your job emits. Show at least the last 20 lines of the console output. (20 points)

AWS Execution

Show a similar screenshot that provides convincing evidence of a successful run of the WordCount program on AWS. Make sure you run the program using at least three small machines, i.e., one master node and two core nodes (those with HDFS) as workers. (20 points)

Once the execution is completed, you can click the 'Debug' button on the Elastic MapReduce panel, pick the 'Custom Jar' step and analyze the 'syslog' output which is under the 'Log Files' section. Make sure you save the controller and syslog log files after the program completed successfully.

Deliverables

1. The report as discussed above. (1 PDF file)
2. The log files (controller and syslog) for a successful run on AWS. (plain text files) (20 points)
3. The final result file(s) produced on AWS. (plain text file) (20 points)