Lab 12: Map Reduce using Hadoop

Instructor: Vivek Sarkar

Resource Summary

Course wiki: https://wiki.rice.edu/confluence/display/PARPROG/COMP322

Staff Email: comp322-staff@mailman.rice.edu

Clear Login: ssh *your-netid*@ssh.clear.rice.edu and then login with your password

Sugar Login: ssh *your-netid*@sugar.rice.edu and then login with your password

Linux Tutorial visit http://www.rcsg.rice.edu/tutorials/

As in past labs, create a text file named `lab_12_written.txt` *in the* `lab_12` *directory, and enter your timings and observations there.*

1 Map and Reduce Operations on Sets of Key-Value Pairs

Map Reduce is a simple data processing paradigm introduced in Lecture 8. To process a data set, the user just needs to provide implementations of map and reduce functions. A single stage of a map-reduce computation typically consists of a map phase, a shuffle phase and a reduce phase. The input and output of a Map Reduce computation is represented as (key, value) pairs. (The input and output pairs can be of different types.) Hadoop Map Reduce is an open source implementation of the MapReduce paradigm originally proposed by Google. It is currently the go to platform for many Big Data applications.

2 Hadoop Map Reduce Environment Setup

1. Log on to Sugar for today's lab.
2. In this lab, we are NOT using java8. Do NOT run the set up script for the previous labs. Type in the following command,
   "`java -version`"
   you should see it is running java 1.6.0 if it is not running the java 1.6.0, run the following command
   "`source /users/COMP322/lab12Setup.txt`"
3. Copy the tar file to your home directory using the following command:
   "`cp /home/yz17/comp322-lab12.tar .`"
4. Run the following command in the `lab_12/` directory to set up the environment for executing Hadoop MapReduce programs:
   "`tar -xvf comp322-lab12.tar`".

   You should now see a directory named "hadoop-1.0.3". You will work in this directory for the rest of the lab.

3 Write a Hadoop Map Reduce Program

1. Study the WordCount.java file, and fill in the code for the map() and reduce() functions as indicated in the comments labeled TODO.

2. Compile the program by using the command:
   "`./compileWordCount.sh`".
   The script will compile the Java file into class files and package them into a jar file. You should see a jar file named WordCount.jar after successfully running the script.

4 Run Hadoop Map Reduce in Standalone mode

1. Obtain a Sugar compute node using the qsub command as in past labs.
2. Go to the hadoop-1.0.3 directory.
3. Change your distribution to Standalone mode using the following command
   "`./changeToStandAlone.sh`"
4. Run the WordCount program in Standalone mode using the following command:
   "`bin/hadoop jar WordCount.jar WordCount inputFiles output`"

   In Standalone mode, the Hadoop FileSystem is not started. It is a mode often used for debugging.
5. Check your output from your word count program with the command:
   "`cat output/*`"
   If you need to modify the WordCount implementation, remove the output directory. Make sure you finish debugging your program in the Standalone mode. Do NOT move to the next section unless you are sure that your program is correct.

5 Run Hadoop Map Reduce in Pseudo Distributed mode

In this section, we will run wordcount in a "Pseudo Distributed" mode. This will create a full fledge Hadoop MapReduce system with multiple processes on a single Sugar node. We will demonstrate utilization of multi-cores in this section.

1. First, convert your Hadoop MapReduce system from Standalone mode to Pseudo Distributed Mode
   "`./changeToPD.sh`"
2. Run your wordcount program using the following script
   "`./runWordCountPD.sh`"
3. Record the running time output from the script

   Hadoop utilizes multi-cores in a machine by having the user specify a parameter in the configuration file. The `mapred.tasktracker.map.tasks.maximum` property determines how many map tasks can run in parallel in the system, thereby using multiple cores.
4. Open "`conf/mapred-site.xml`" using any text editor, and change the value of the property "`mapred.tasktracker.map.tasks.maximum`" to 2. This means that you are running twp map tasks in parallel, using only 2 cores. Previously, it was set to 1, indicating that you are using only a single core for execution.
5. Run your wordcount program using the following script and record the running time of the program "`./runWordCountPD.sh`"

   This running time is for executing two map tasks in parallel.
6. Repeat the process and set the value of the parameter to 4, 8 and record the running times in `lab_12_written.txt`. There are many reasons that you don't see a linear speed up as you use more cores. For example, the reduce phase is not parallelized.

7. Take a few minutes to look at the runWordCountPD.sh file. It shows how you work with the Hadoop File System. Hadoop Map Reduce is built on top of HDFS.

6 Turning in your lab work

1. Submit ONLY the `WordCount.java` and `lab_12_written.txt` files.You should have recorded running time results in the textfile.

2. Use the turn-in script to submit the file as a new `lab_12` directory in your turnin repository as explained in the first hadnout. You can always examine the most recent contents of your svn repository by visiting `https://svn.rice.edu/r/comp322/turnin/S14/`*your-netid*.