ENSC 351

Summer 2018

Lab 3

MapReduce

Deadline: October 17th, 2018
Last revision: October 11, 2018

Karol Swietlicki
Simon Fraser University

1 Map reduce!

Threading goes pretty far, but for the largest of problems there is a need for multiple computers collaborating on one problem.

MapReduce is a way of spreading gigantic loads across multiple machines, but there is a catch...

Our MapReduce will be a bit different than the one usually done, but it will still be a valuable lesson.

2 The task at hand

Implement a word count twice and then come up with a new task.

3 First word count

Output a program that reads in a text file and outputs a word count for that file. Have the output be sorted by counts, descending.

This program MUST NOT be multi-threaded.

4 Second word count

Divide your program into four pieces.

You need to obey a very specific interface.

The input reader produces input items. The mapper takes in input items and outputs key-value pairs. The reducer both takes and outputs key-value pairs. The outputter takes the key-value pairs and outputs the output to the screen.

Your architecture needs to be modular. You need to write things in two layers. The first layer is the core map-reduce algorithm. The second layer is just the four pieces. Replacing the four pieces with different ones should change the task that is being executed.

4.1 Task-specific parts

For the word count, here are your task-specific pieces:

4.1.1 Input reader

Read in your text file. Divide it into chunks by dividing the input file into one-word segments.

4.1.2 Map

Transform each word into a key-value pair. Have the key be the word and the value to be one.

Return the key-value pair.

4.1.3 Reduce

Given a list of key-value pairs, all of which have the same key, reduce all the values into one.

Produce a single key-value pair, where the keys are still the words and the values are the sums of the

4.1.4 Output

Take in the key-value pairs from the previous step and print the word counts.

4.2 The generic parts

The rest of the program needs to handle calling the four functions, creating threads and feeding the results from one stage to another. This part is immutable and does not change from task to task. You have complete freedom here.

The general idea of the generic part is that it will read the input file and feed it into the input reader. The problem-specific input reader will return an array of something to the generic part. The generic part will then feed that array into map, spreading the workload over many threads. Key-value pairs will be collected from the map stage, grouped together when the key is common, and groups will be fed into the reduce stage. The key-value pairs from the reducers will be finally fed into the output stage.

5 A better fit

Invent a workload that will fit the MapReduce framework better than word counts. Experiment.

Your new workload must be faster in MapReduce than in a single-threaded setting. Your new workload must be non-trivial. Both map and reduce stages need to be doing work.

6 The deliverable

Write a lab report which explains your workload, how did you come up with it and why is it faster than a single-threaded implementation.

Comment on the efficiency of MapReduce of word counts as compared to the efficiency of the single-threaded implementation.

Comment on which kind of workload is best suited to MapReduce.

MapReduce is often run with the map phase spread across multiple computers. How would this impact the execution speed?