Programming Assignment 2:
Message Passing and Cloud Computing

ECSE 420 – Parallel Computing – Fall 2015

Released: October 7th, 2015
Due: October 19th, 2015 at 11:59 pm

1 We <3 Cats

Through shady means, you have acquired a database of 5 million social media users' likes and profile information, encoded in the convenient form of a 5,000,000 x 5,000,001 matrix of double-precision floating-point numbers.

You and your cutting-edge machine learning company, Neural Petwork, have developed an algorithm to determine, on the basis of this matrix, which social media users are likely to click on ads for cat food.

When the user matrix is put in reduced row echelon form (RREF), each column except the last represents a person, and the final entry of the row in which that person's entry is equal to 1 represents a certain metric for that person. By dividing all of these metrics by the metric with the greatest magnitude and taking the absolute value of the resulting matrix, you get the probability that each person will click on an ad. For example:

$$\left(\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}\right)$$

$$\downarrow \text{RREF}$$

$$\left(\begin{array}{cc|c} 1 & 0 & -1 \\ 0 & 1 & 2 \end{array}\right)$$

$$\downarrow * \frac{1}{Max\{|-1|,|2|\}}$$

$$\left(\begin{array}{cc|c} 0.5 & 0 & -0.5 \\ 0 & 0.5 & 1 \end{array}\right)$$

$$\downarrow \text{ABS}$$

$$\left(\begin{array}{cc|c} 0.5 & 0 & 0.5 \\ 0 & 0.5 & 1 \end{array}\right)$$

You plan to use the information generated by this bogus algorithm to get a list of users with high clicking probabilities for We <3 Cats, a store which sells cat food. For every user in your list who clicks on an ad, We <3 Cats will pay you $1. Riches await you!

1.1 Coding (40%)

Write a parallel program to calculate all of the clicking probabilities given a user matrix. The user matrices have the same format as the test matrix supplied with this assignment: text (not binary), spaces between values, and newlines between rows. Your parallel program should have the following command line invocation:

mpirun -np<number of processes for mpirun to create> ./cp_mpi <path to text file containing user matrix>

Running your program should create a text file called "clicking_probabilities.txt" with the clicking probabilities separated by newlines. You can use the Linear Algebra Toolkit (http://www.math.odu.edu/ bogacki/cgi-bin/lat.cgi) to check whether your program is correct for small test matrices. As in Programming Assignment 1, if the number of processes does not divide the matrix size evenly, make sure that the remainder elements are handled by one or more processes.

For this assignment, you can make the simplifying assumption that the matrix will always be of size <number of users> x <number of users + 1>, and that row reduction will not result in inconsistent rows (rows in which all entries except for the last entry = 0) or redundant rows (rows in which all entries = 0). You can use the provided sequential program as a starting point.

Note: You cannot use any shared memory techniques. Each process should read its portion of the user matrix from the user matrix text file. If a process reads a value from the file, this implies that all other processes must use MPI calls to access that value. This constraint will allow you to scale the program to run on a cluster computer.

1.2 Competitor Analysis (20%)

Through equally shady means, your competitors, SaaS-Whole, have acquired this same user matrix, and your definitely illegal corporate espionage team estimates that SaaS-Whole will finish solving their equivalent problem in 3 days.

a. Assume that the sequential version of your program is 99.9% parallelizable. You estimate that the runtime without any parallelism would be 300,000,000 seconds. Do you have any hope of finishing the computation before your competitors? What if the estimated runtime were 150,000,000 seconds?

b. Repeat **a.** using the constrained version of the Amdahl bound that you derived in Theoretical Assignment 2, assuming that OV(n) = 10n. (You can either use calculus to find the answer analytically or write a program to find the answer numerically.)

c. Estimate what percentage of your program's runtime is parallelizable using the stopwatch commands provided in Programming Assignment 1 or some other timing method. In your report, explain how your measurement works and justify your method. Repeat **a.** using your program's percentage instead of 99.9%.

1.3 Deployment (20%)

You decide to deploy your code on a large cluster computer. Your IaaS provider, Nebulous Contract, charges $0.01 per compute-hour (i.e., the cost of running two CPU instances for one hour is $0.02). Assuming once more that your program is 99.9% parallelizable and takes 150,000,000 seconds to run, using the constrained Amdahl bound, how much would it cost to buy enough CPU instances from Nebulous Contract to run your MPI program and achieve i) half and ii) three-quarters of the theoretical maximum speedup?

1.4 Optimization (20%)

We <3 Cats inform you that they will charge you $2 for every user you provide who does NOT click on a cat food ad. You decide not to give the whole list to We <3 Cats, but rather only the users whose clicking probability is above a certain threshold (the "acceptance threshold"). Your program should calculate, for a given user matrix, which of the following acceptance thresholds maximizes your expected profit:

i) 0.2
ii) 0.4
iii) 0.6
iv) 0.8
v) 1.0

(For example, given three users, if the clicking probabilities for users 1, 2, and 3 are 0.1, 0.5, and 1, respectively, your expected profit for each of these acceptance thresholds would be:

i) 0.2 → $0.50
ii) 0.4 → $0.50
iii) 0.6 → $1.00
iv) 0.8 → $1.00
v) 1.0 → $0

so your best bet would be to use an acceptance threshold of 0.6 or 0.8. For this assignment, if two acceptance thresholds result in the same expected profit, choose the lower threshold, e.g. 0.6 in this case.)

You should include this step at the end of your parallel clicking probability calculator. Use the MPI_Reduce function to do the calculations. Your program should simply print which of these 5 acceptance thresholds is best for the given user matrix, followed by a newline.

Deliverables

Provide a zip file containing a PDF report, your code, and a Makefile for your code. You may use any programming languages, libraries, or compiler optimizations you like, as long as i) running "make" inside your submission folder on the Trottier Unix computers (i.e. the ones listed here: http://www.mcgill.ca/ece/department/itservice/faq) produces the correct executables, ii) your executables produce the correct outputs when run on these computers, and iii) no process accesses more than half of the user matrix directly (i.e. without MPI calls).

Bonuses:

1) The fastest submission will win 10 bonus points.

2) For 20 more bonus points, prove in your report that you got your code to run in parallel on at least three hosts. (Anyone aiming for this bonus can borrow some of my Raspberry Pis- first come, first serve.)

Note: you only get bonus points if your program produces the correct outputs.