Marshall Lanning

CECS 625 Parallel Programming Homework Assignment #4

October 21, 2019 (100 points)

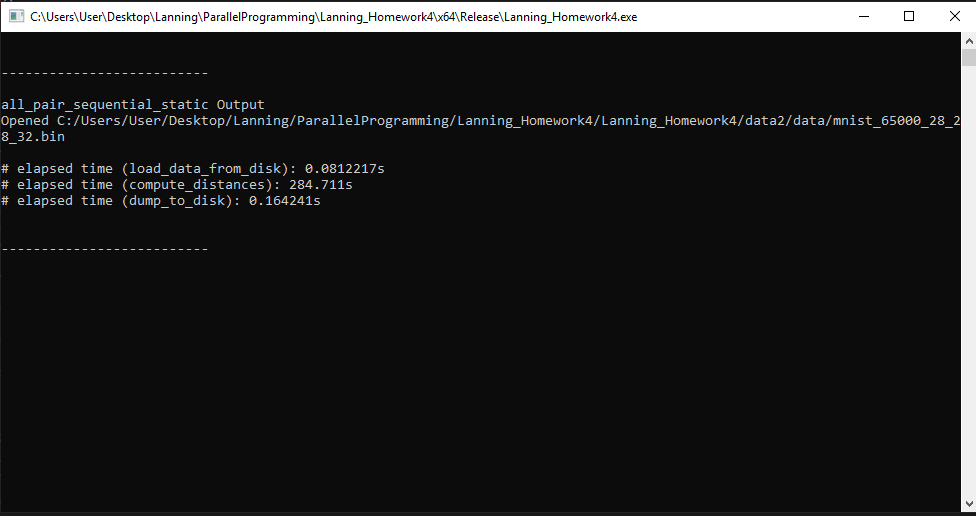
Due: November 6 (Wed) midnight

(Submit your project report and any required VS 2015 project to the Blackboard.)

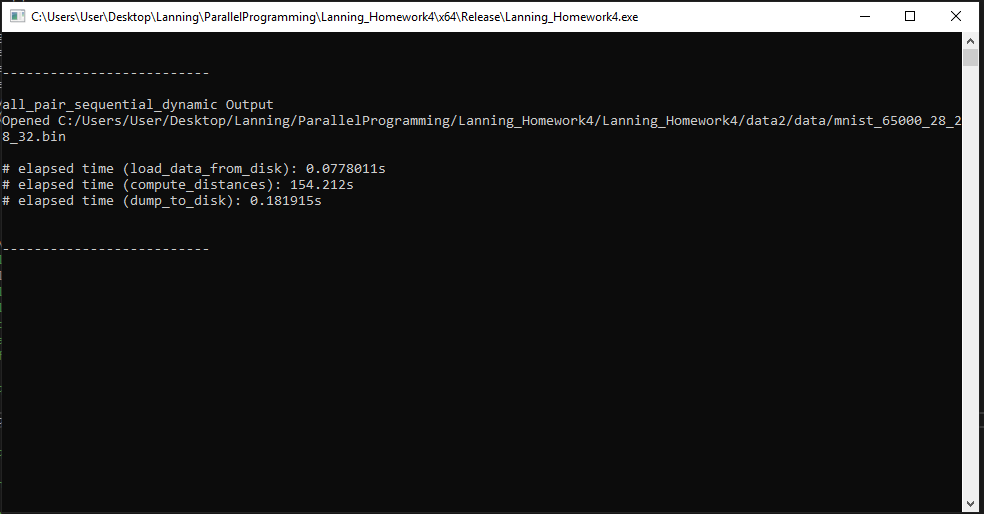
Assignment Description OpenMP-related problems

1 (20 points) Consider the source file, all\_pair.cpp, in Section 4.4. Parallelize the   
 sequential version, sequential\_all\_pairs, using OpenMP with three differnet   
 scheduling polycies, static, dynamic, and s (see page 190 of the textbook).   
 Comprare timing performance of these three OpenMP scheduling implementation and   
 the C++ multi-threading version, dynamic\_all\_pairs. Show and discuss the timing   
 results in the project report.

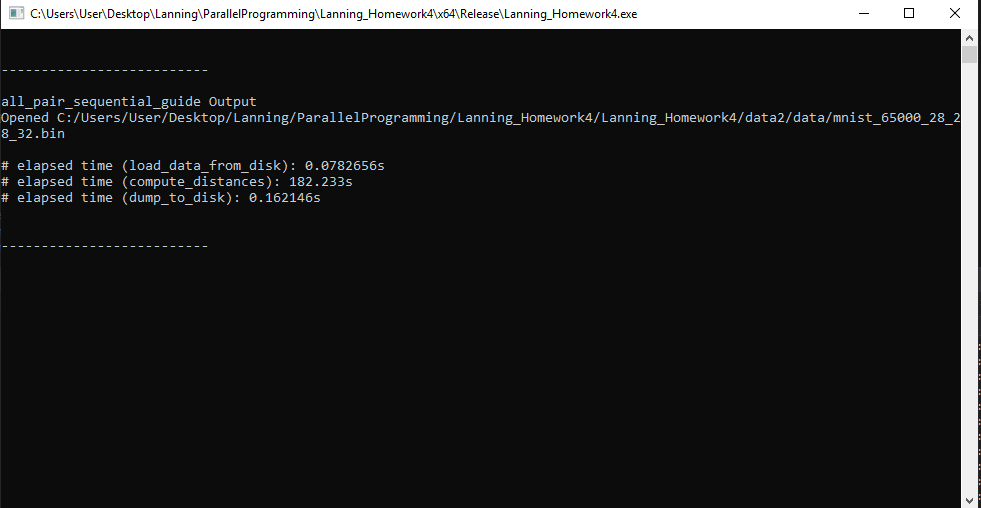
Static Scheduling



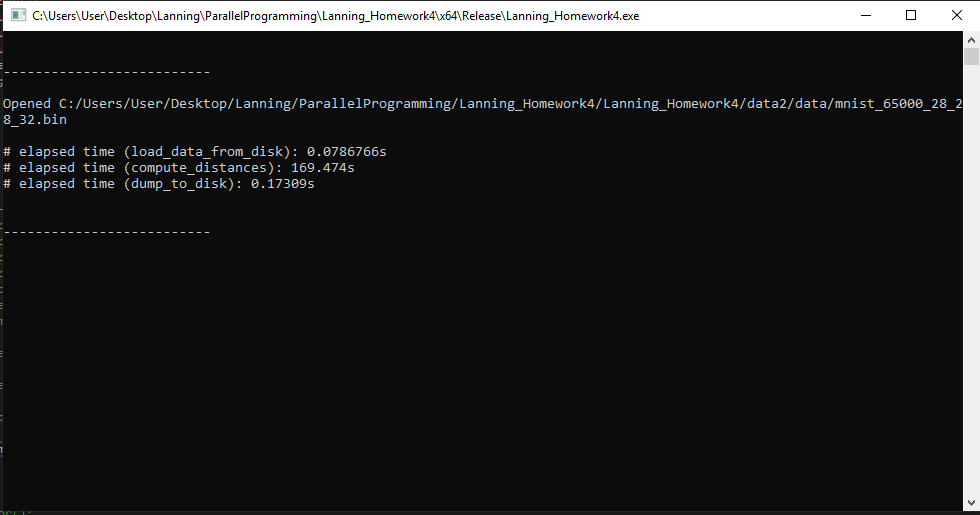
Dynamic Scheduling



Guided Scheduling



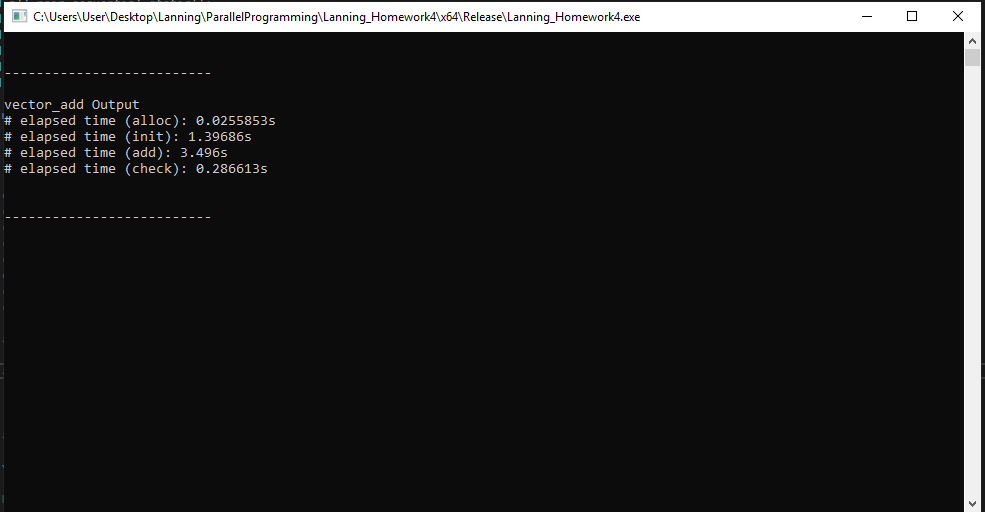
Multi-Threaded Dynamic



As you can see the best performer here was sequential\_all\_pair with OpenMP scheduling set to dynamic. The next best was the Multi-Threaded Dynamic implementation then guided then static. Static does the worst here because it divides the work for all\_pair up in many chunks and causes idling at the end when there are threads still doing work. Dynamic scheduling does best here because there is basically no idle time. With dynamic scheduling, work is distributed evenly and work is given to the threads as soon as they are ready for more work. Guided does a decent job here and divides the work load up into chunks where each successive chunk decreases in size. You can really visualize the workload when viewing the CPU usage in the Diagonistic Tools when running the program.

Similar to the Project 3, create an OpenMP C++ project (called chapter6) in VS 2015 and copy the indclud folder (as in Project 3) and the data folder (downloaded from the Blackboard; note this data folder is not the same as the data folder used in Project 3) to the new project home folder. Use this new project to do problems 2-4.

2 (10 points) Add the source file, vector\_add.cpp (listed and explained in Section  
 6.2), from the textbook’s souce code website. Modify vector\_add.cpp so that it will   
 run in VC++ and OpenMp 2.0. Explain your modifications and discuss the timing   
 results in your project report.



The only modifications needed to be made for this are to replace any uint64\_t declarations with int64\_t. When this is ran without OpenMP parralelization the time to add is 10.47 seconds, when it is ran with OpenMP parralelization the time to add is 3.49 seconds. This proved the effectiveness of OpenMP’s parallelization features of for loops.

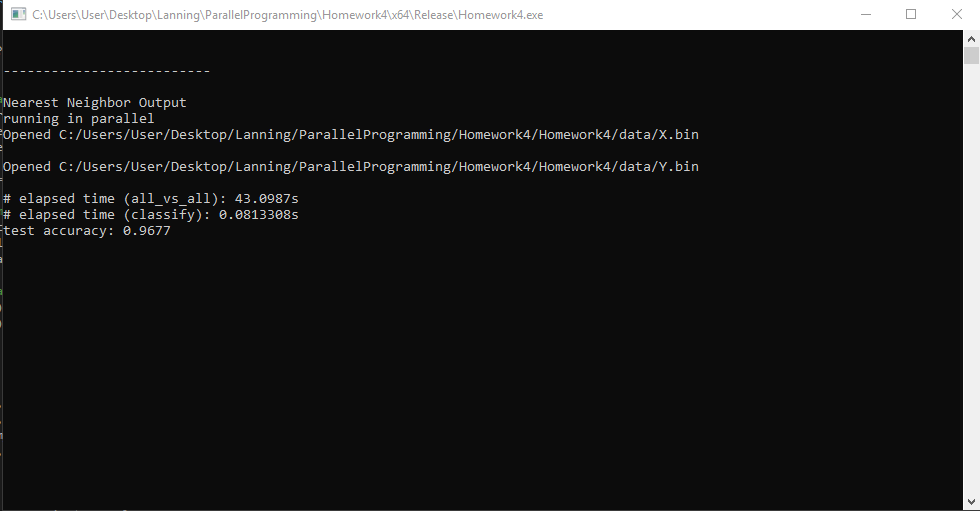
3 (25 points) Add the source file, 1NN.cpp (listed and explained in Section 6.3), from the   
 textbook’s souce code website. Modify 1NN.cpp so that it will run in VC++ and   
 OpenMp 2.0.

1. Explain your modifications in your project report.

The modifications I made for 1NN were to first replace any uint64\_t declarations with int64\_t. I then removed any collapse() calls made when using OpenMP because it is not supported in OpenMP 2.0. I then replaced main with my own function declaration to be called in my own main function. I then replaced the fulepath calling the X and Y binary files to match exactly where I put them on the PC.

1. Show and explain the outputs from 1NN.cpp.

The output shows the timing result of an all\_vs\_all method implementation with OpenMP parralelization applied. As you can see it ran in under a minute. The all\_vs\_all method is used to calculate the distance between a given input X and 65,000 images from the mnist data set. Parallelization is very effective here because of the need to iterate through all 65,000 images. The accuracy output shown below is to display the result of how accurate the Nearest Neighbor algorithm guess what a number was according to how big the distance was from the guessed number in terms of probability.

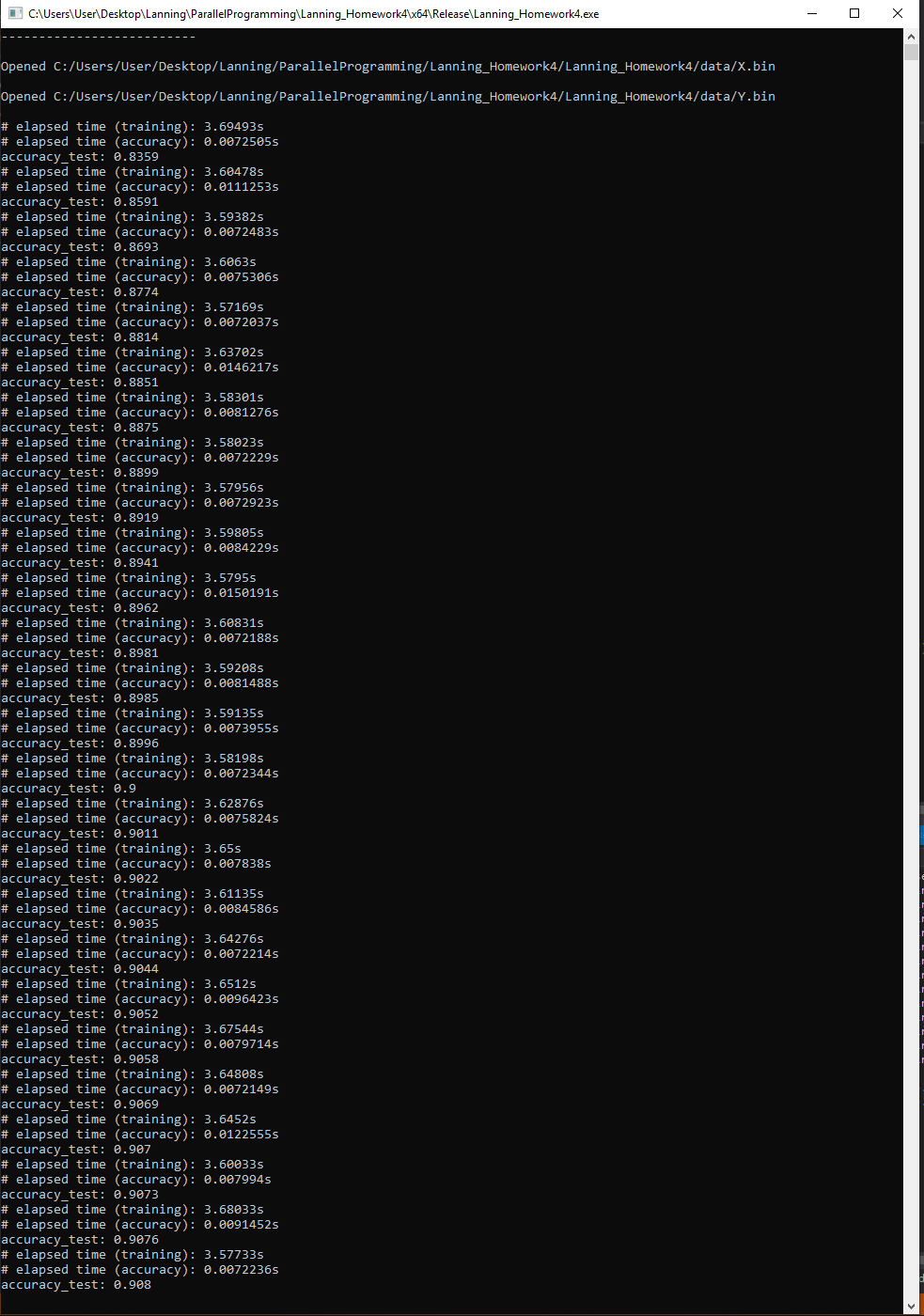


4 (45 points) Add the source file, softmax.cpp (listed and explained in Section 6.5),   
 from the textbook’s souce code website. Modify softmax.cpp so that it will run in   
 VC++ and OpenMp 2.0.

1. Explain your modifications in your report.

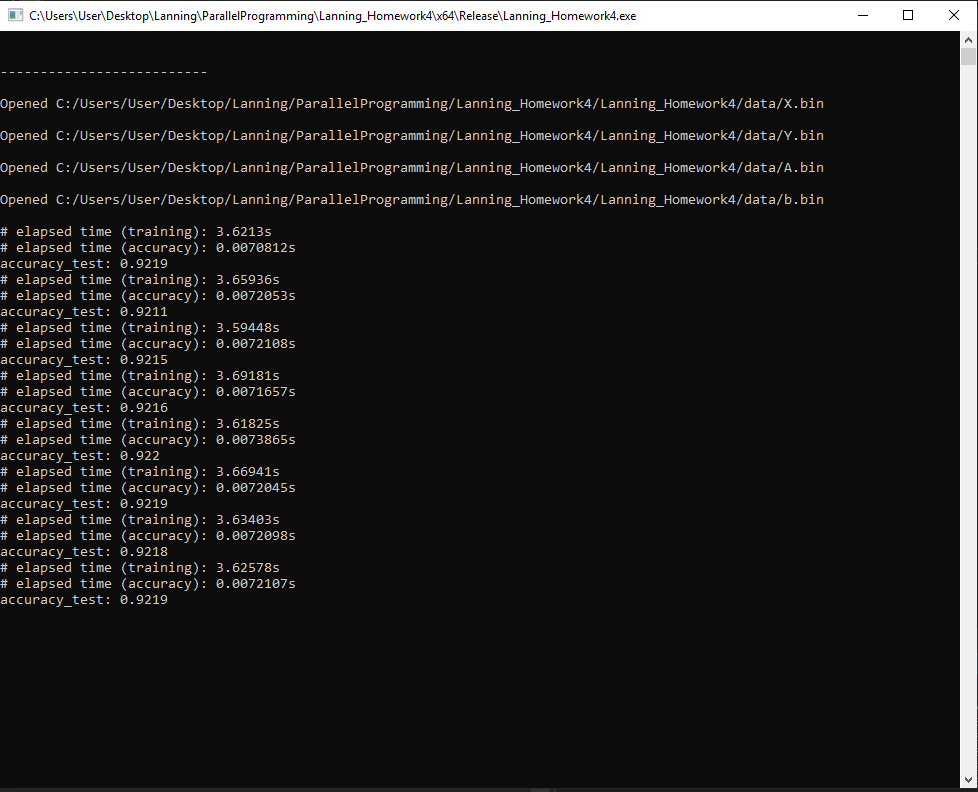
To get this working I first replaced any uint\_64 declarations with int\_64. I then renamed main to match my own function calls in my own main. I then needed to include the algorithm library for the std::max functions that are called. In the accuracy function I needed to replace value\_t output[num\_classes]; with value\_t output[10]; because of the error it threw about the array size being declared as not a constant. I chose 10 because the number of classes will always be 10 in the case of the dataset being numbers from 0 – 9. I then needed to make the function calls to softmax\_regression and argmax in accuracy to have this <float, int64\_t> after the function name. This was because of the way the functions were created with templates. The value\_t and index\_t items were to be floats and int64\_t values when making those calls. I then needed to change the filepath to the X, Y, A, and b files.

1. Show and explain the outputs from softmax.cpp.



What this function is doing is making 32 iterations at a time through the mnist data set to train the machine to know the handwritten numbers. The more it trains the more accurate it becomes, hence the increase in accuracy each time.

1. Use the pre-trained softmax (as given in data/A.bin and data/b.bin) and get its prediction accuracy. (Note A and b are the required parameters to run softmax regression.) Compare and discuss the accuracy with those obtained in (b).



The pretrained softmax weights and biases show a consistent and effective accuracy throughout each iteration because the weights and bias values are from a pre-trained data set. With the dataset already trained we are expected to have consistent and mostly accurate results when making guesses at numbers.

1. Compare the advantages and disvantages of using 1NN vs softmax as the classifier for MNIST data.

Softmax is better due to its speed and ability to be well trained. Softmax mimics a neural network for machine learning that can literally learn things like a human can. In this case it learns the values of handwritten digits. Another common image recognition application this can be used for is differentiating between cats and dogs, and other various animals. It is a powerful achievement in computing and a very interesting topic that we have been able to go over.