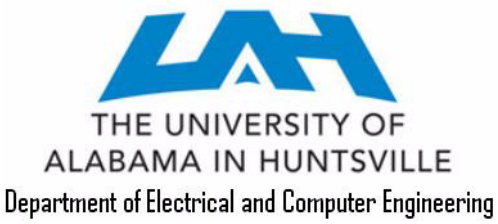

CPE 412/512 Final Exam

Fall Semester 2015
Optional Take-Home Component

Name _____

INSTRUCTIONS: **The following represents two take-home type problems which CPE 512/412 students may elect to work and apply for credit toward their final exam. The due date for this material is by 11:30 AM on Tuesday, December 8, 2015 on the UAH CPE 512/412 Canvas™ site. Each problem worked can be used to replace one of the problems on the in-class portion of the Final Exam. The total number of equally-wighted problems on the in-class portion of the final exam is expected to be five (5) or six (6), with CPE 412 students being assigned one problem less than the CPE 512 students. Students will indicate during the in-class portion of the Final Exam which problem(s) (if any) from this take home exam will be substituted for specific problem(s) the in-class portion of the Final Exam.**

Statement of Compliance with UAH Academic Misconduct Policies

I _____ **certify that I have worked independently of**
(sign your name)
others on this test and the work that I am presenting is my own. I am familiar with the UAH academic misconduct policy as outlined in the UAH student handbook and have agreed to abide by the policies that are stated in this document.

Problem 1: OpenACC -- Laplace Example & Matrix/Matrix Multiplication

Problem 1 is divided into two main parts, the first being more tutorial in nature closely following the lecture slides. The second part allows you to create yet another matrix/matrix multiplication program but this time the program will execute using a GPU instead of executing on a multi-core general-purpose CPU environment.

The OpenACC portion of this assignment must be executed on the Alabama Supercomputer Center's DMC system (dmc.asc.edu). The steps necessary to compile an OpenACC compliant C++ program and execute it in batch mode using one of the DMC's NVIDIA Fermi or Kepler GPU accelerators are discussed in the separate document. This document is also posted on the course Canvas site under the Final Exam Take-Home Component Assignment.

Part 1, Problem 1: For this part of Problem 1 you are directed to the OpenMP Jacobi Iteration example and the OpenACC Jacobi Iteration template file *laplace2d_acc.cpp* and its associated timer.h file that is posted on the class Canvas™ website under the "Optional Final Take home Component" Assignment. These files are also located under the /home/shared/wells/final/prob1 directory on the dmc.asc.edu system. This problem is very similar to the two-dimensional heat transfer problem discussed several times in class in that it is an iterative method that employes a finite element two-dimensional nearest-neighbor structure.

Action 1.1:

-- Execute the OpenMP Jacobi Iteration example using the standard batch queue (run_script) on the dmc system. Record the Execution time reported for the 1, 2, 4, and 8 core implementation. Make at least 5 runs for each case and report the minimum execution times that are reported. Use these values for comparison purposes with the OpenACC implementations.

Action 1.2

-- Add the necessary OpenACC pragmas to execute the two main loops as separate kernels on the GPU. Measure the performance of this implementation. Compare the run-time performance obtained for this implementation with the 1, 2,4, and 8 core OpenMP minimum run time implementations that you made in Action 1.1 of this problem. As part of you answer to this question highlight the changes that you made to the original file.

Action 1.3

-- Add to the code you implemented in Action 1.2 any necessary OpenACC pragmas to reduce the data transfer time between the host and GPU. Measure the performance of this implementation. Compare your performance to the performance of the 1, 2,4, and 8 core OpenMP implementations that you made in Action 1.1 of this assignment. As part of you answer to this question highlight the changes that you made to the Action 1.2 source file.

Action 1.4

-- Add the *loop gang* and *vector* parameters as part of additional OpenACC pragmas that you have added to your OpenACC code from Action 1.3 and experimentally determine the best values for these parameters for this application. **Note that the tile clause is <u>not</u> available in the version of openacc that you are using on the dmc.** Measure the performance of this implementation. Com-

pare the run-time performance of the implementation with the implementation that contains your best set of parameters with the 1, 2,4, and 8 core OpenMP implementations that you made in Action 1.1 of this assignment. As part of you answer to this question highlight the changes that you made to the Action 1.3 source file.

Part 2, Problem 1: Modify the serial matrix/matrix multiplication program, matrix_mult_serial2.cpp, that is also posted that is posted on the class Canvas™ website under the "Optional Final Take home Component" Assignment. This file is also located under the /home/shared/wells/final/prob1 directory on the dmc.asc.edu system. This file is a bit different from the matrix_mult_serial.cpp file used in HW3, and HW4 in that it transposes one of the B matrix to provide better serial cache performance on many single core systems.

Action 2.1

-- Create an efficient OpenACC implementation by inserting the necessary compiler pragmas. Optimize as best you can for data and loop control. Then evaluate the performance of your implementation on the NVIDA Fermi or Kepler GPU card (your choice) on the dmc system. Measure this performance for matrix sizes of 8000. As part of you answer to this question highlight the changes that you made to the original matrix_mult_serial2.cpp file using the OpenACC C++ extensions along with your run-time data.

Action 2.2

-- Compare the performance of your OpenACC implementation you made in Action 2.1 with the run time that you have measured for the single core, s*erial matrix_mult_serial2.cpp*, implementation when it is run on the dmc.asc.edu system for a data size of 8000 x 8000 x 8000. Use a batch queue to measure this serial run time. What is the speedup of your best representation relative to this run time?

Problem 2: Jetson Network Communication Attribute Determination. Message Passing Bandwidth and Startup Latency Characterization

On the UAH Jetson Cluster develop a ping-pong type program that can be used to empirically measure the basic inter-MPI node communication attributes of the system for point-to-point type communication assuming the linear model for communications that was discussed in the class and in the text. In this linear communication model data transfer time is composed of a latency component, $T_{startup}$ and a component that incorporates a per byte transfer rate component, T_{data}. Set up your program so that it measures the round trip time it takes one node to send a specified number of bytes from one MPI process to another a targeted MPI process and then have that process send this same data back to the first MPI process. You are to use standard locally blocking ***MPI_Send*** and ***MPI_Recv*** routines. The two processes must be located on separate nodes of the UAH Jetson cluster. Also to account for the relatively coarse granularity of the timer, the time it takes to perform 1000 consecutive ping-pong type data transfers should be measured by the program. This data should then be used to determine the effective average time for each one way send/receive combination. This average can then be used to compute the effective $T_{startup}$ and T_{data} parameters for a single paired ***MPI_Send*** and ***MPI_Recv*** communication using standard linear algebraic methods.

Record the execution times for data sizes of 16384, 32,768 65,536 131,072 262,144 524,288 1,048,576, 2,097,152, 4,194,304 bytes. Then plot this data on a graph. Show how this data is used to calculate estimates for the Tstartup and Tdata parameters and clearly indicate the values of these parameters.

Due date for is Tuesday December 8, 2015, 11:30 AM