**CSE-443/543: High Performance Computing**

**Lab #06**

Max Points: 44

**Objective**: The objective of this exercise is to:

* Build experience with compiling code at the command line and constructing batch shell scripts to run a parallel program
* Gain familiarity with compiling and running parallel code using OpenMP.
* Gain familiarity with the concepts of parallel program performance discussed in class, in particular Amdahl’s law.

Fill in answers to all of the questions in this document and the corresponding spreadsheet. You may discuss the questions with your instructor and the TA.

## Background

In this lab you will be running code that takes advantage of parallelism provided by OpenMP. The code itself is quite simple, initializing an array with a table of sines, then iteratively making changes to elements of this table. But the particular code is not important: The relevant thing is that it is simple and can demonstrate the basic functionality of using OpenMP to parallelize code.

Recall that the OpenMP application programming interface (API) consists of three components:

1. Environment variables
2. Compiler directives embedded in the source code
3. Runtime library routines

This lab will give you exposure to the first two of these. The OMP\_NUM\_THREADS environment variable will be used to run the code with differing numbers of threads, and the source code itself uses the most commonly used OpenMP directive #pragma omp parallel for.

Although the program provided is potentially almost perfectly parallelizable, it has been tweaked to include portions that are run serially. This is to help demonstrate Amdahl’s law and how it describes the limit of parallel program speedup when it includes serial regions.

You will be determining the parallel speedup ratio Sn, the parallel efficiency En, and the parallel cost Pn in this lab. Refer to the Session 12 presentation if you need a refresher in how to calculate these quantities.

## Experiment

1. Download the file Lab06.cpp from Canvas.
2. Write a bash shell script that accomplishes the following:
   1. Invokes /bin/bash as the shell
   2. Determines the model name and speed of the processor
   3. Determines the total amount of memory available on the machine
   4. Compiles the Lab06.cpp code to create a program called Lab06 using g++ with these options. (Note: you have to include the g++ command – those are just the options you give it!) Also, please ignore the warning message about the unknown pragma.

-g –O3 –std=c++14 -Wall

* 1. Runs Lab06 twice and times it using /usr/bin/time. To make your life easier, you may want to recall how to change the output format output of /usr/bin/time…
  2. Compiles the Lab06.cpp code to create a Lab06\_omp program using g++ with these options:

-g –O3 –std=c++14 –Wall -fopenmp

* 1. Sets the number of threads to run to 1 (via export OMP\_NUM\_THREADS=1)
  2. Runs Lab06\_omp twice and times it using /usr/bin/time
  3. Sets the number of threads to run to 2
  4. Runs Lab06\_omp twice and times it using /usr/bin/time
  5. Repeats i and j, but sets the number of threads to 3, 4, 5, 6, 7 and 8

# Apparatus (platform for experiment)

The first experiment documented in this report was conducted on the following platform (fill in the Details column using information determined in your shell script):

|  |  |
| --- | --- |
| Component | Details |
| CPU Model | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz |
| Main Memory (RAM) size | 8071532 kB |

# Observations – Lab06

Record timing information from /usr/bin/time in the appropriate tables in the Lab06 – Results Google spreadsheet:

# Analysis – Lab06

Using the definitions in the [Session 12 presentation](https://docs.google.com/presentation/d/1noK7mMx14QScNMZ7DtF-wdjsF5g6YFvMOoNkeM7mqQY/edit#slide=id.g6460560ab5_10_208), compute the parallel speedup ratio, the parallel efficiency, and the parallel cost for this experiment using the associated Google spreadsheet. Remember to use the times from the parallel version of the code run with OMP\_NUM\_THREADS=1 as the single worker numbers.

# Submit files to Canvas

When you complete the lab, download this document from Google Drive as a Microsoft Word (.docx) file with the naming convention Lab06 - MUid.docx (example: Lab06 - ferrenam.docx).  You should save the corresponding Google spreadsheet file as a Microsoft Excel (.xlsx) file with the naming convention Lab06 – Results MUid.xlsx (example: Lab05 - Results - ferrenam.xlsx)

Then, submit the following files to Canvas:

1. The Microsoft Word file you downloaded from Google Drive.
2. The Microsoft Excel file you downloaded from Google Drive.
3. The Lab06.bash shell script you created for this lab.