(../../index.php)

# CS 475: Parallel Programming
*Computer Science Department (http://www.cs.colostate.edu)*
Fall 2016
CS 475 Assignment 01 -
This is not a test.


(http://welcome.colostate.edu/)

## Objectives

The objective of this homework is to write three OpenMP programs, to debug and test them on a Capital machine, and to experimentally determine the gains you get in running it in parallel with 1, 2, 3, 4, 5, 6, 7, and 8 threads. The parallelizations are relatively simple, and the results should be interesting in terms of speedup. You should measure and plot the performance of your parallelization as a function of the number of threads, and analyze your observations. Can you see a correspondence between memory (re)use and max. speedup?

- 1. Jacobi Stencil 1D
  Parallelize the Jacobi stencil computation from the provided sequential code.
- 2. Jacobi Stencil 2D
  This is a 2D extension of the previous program; the data is updated using the values of four neighboring elements.
- 3. Matrix-vector Product
  Parallelize the provided sequential program for the Matrix-vector product.

## Provided Code

Download and untar this tarball (PA1.tar) The input usage for each program is
- jacobi_1D 4000 200000
- jacobi_2D 800 2000
- mat_vec 15000 10000

## Submission Instructions:

- Code Submission:
  You need to submit your source code in a single tarball named PA1.1.tar and PA1.2.tar for phase-1 and phase-2 respectively. The tarball should contain the following list of files named EXACTLY as listed here.
  - jacobi_1D.c
  - jacobi_2D.c
  - makefile

- mat_vec.c

During testing we will run make clean; and make; followed by a series of automated tests on your executables. Your executables must be named: jacobi_1D, jacobi_2D, and mat_vec.
We will be performing automated testing on your output. Do not change the output format from the existing format.

Below is an example:

```
$ jacobi_1D 4000 200000
data[400]: 1366.611240
data[800]: 142.298734
data[1200]: 5.074897
data[1600]: 0.058852
Data size : 4000  , #iterations : 200000 , time : 0.924085 sec

$ jacobi_2D 800 2000
Data : 800 by 800 , Iterations : 2000 , Time : 0.524361 sec
Final data
56111.442113 28472.061997 28460.727609 28460.727563
28472.061997 23.332803 11.666517 11.666470
28460.727609 11.666517 0.000094 0.000047
28460.727563 11.666470 0.000047 0.000000

$ mat_vec 25000 10000
N=25000, M=10000
c[0] = 49995000.000000
c[3125] = 81245000.000000
c[6250] = 112495000.000000
c[9375] = 143745000.000000
c[12500] = 174995000.000000
c[15625] = 206245000.000000
c[18750] = 237495000.000000
c[21875] = 268745000.000000
elapsed time = 0.050035
```

If you use the website to submit your assignment there will be initial tests (Preliminary Testing) performed. These tests do not indicate your final grade. They can however catch small mistakes in your submission. You can re-submit your file until you get 100% on the Preliminary Testing. Failure of preliminary testing at due-date(time) will lead to a zero in your final grade. So, please make sure to submit in advance and pass on the Preliminary Testing.

- Report Submission:
  You are responsible for submitting a report presenting and explaining your performance results.

1. Algorithm Description (include a figure when appropriate).
2. Description of parallelization approach.
3. Experimental setup:
   1. Describe the machine (name, number of cores, cache sizes).
   2. List experiments planned (core count).
4. Experimental results.
   1. Compare sequential and threaded times.
   2. Include tables and graphs of execution times, speedup, and efficiency.
   3. Make observations about speedup.
5. Conclusion: Can you see a correspondence between memory (re)use and max. speedup?

- You can see the Grading Logistics here (PA1_Grading_Logistics.pdf)