COMP/CS 605: Introduction to Parallel Computing (Spring, 2017)

Course Syllabus

This document was last Updated: 01/27/2017

[Return to course website]

Course Information:

- Description
- Prerequisites
- Textbooks
- Grading
- Assignments
- Add Policy
- Cheating
- Course Links & Information

Course Description:

COMP 605 provides experience in designing and running parallel programs in a modern academic cluster setting. One goal is to make our way through as many parallel programming exercises as time will allow in an attempt to tackle computationally intense problems from both the CPU and GPU side, and to benchmark the results. This is parallel computing by doing, primarily in a Linux cluster environment, using C and Fortran programming languages. A broad goal is to provide students with tools and expertise that will help further the computational aspect of their research efforts.

Translating a mathematical description of a problem into a computer program description is a prerequisite skill for the course, as are: fluency with the objects, language, and methods of linear algebra & undergraduate calculus, program development (using line editors such as vi, and makefiles) in the Unix command environment, and writing C-Language computer programs. (Note: basic programming is not taught in this class but see COMP 526, where Unix/Linux command line environment is also taught). Here it is assumed that the student is an accomplished C programmer, and that one is able to expertly navigate the Unix command line interface.

COMP 605 is hands-on, with weekly and semi-weekly programming assignments.

The course will consist of the following modules, based on Pacheco's 2011 book as a guide, and through programming GPU devices using CUDA. These modules consist of:

1. Introduction to Parallel Computing and SciComp Basics (Unix, performance, benchmarking, analysis, resource management)
2. Distributed Computing with Message Passing Interface
3. Shared-Memory Programming with Pthreads and OpenMP
4. Cuda Programming

Each of these modules will have 1-2 homework assignments with an exam on the topic. See Homework and Course Policies for more details.

[Back to Top]
Course Prerequisites:

The ability to program well in C is a requirement for the class. In particular, the NVIDIA CUDA compiler for GPU programming is a C compiler (with extensions). C language or Fortran 90 are supported for use in programming MPI. Familiarity with navigating, working with files, and compiling programs in a Unix or Linux environment is also assumed. Mathematical knowledge at the level of performing matrix operations in linear algebra and derivatives and (multiple) integrals in calculus is required. The course will utilize C (required) and Fortran 90 (optional) programming languages along with MPI and CUDA extensions to emphasize key parallel programming concepts. C language is required.

This course is designed for upper division/graduate students enrolled in a scientific discipline such as Computer Science, Computational Science, Engineering, Bioinformatics, with suitable preparation. Students must have senior or graduate standing.

- Students should have experience with:
  - C, and optionally Fortran
  - Basics of Unix OS
  - fluency with the objects, language, and methods of linear algebra & undergraduate calculus.
  - Writing 3-5 page lab report papers

- Students ideally are familiar with:
  - PDE's
  - principles of computational science
  - concepts of parallel programming (thread management, queuing systems, multiple cpu programs)

Note: Computer Science 501, 520, 0525; Computational Science 526, or equivalent Unix OS experience is helpful.

Recommended Textbooks:

- GPU/CUDA Programming: [CUDA By Example, by Sanders and Kandrot](http://example.com) [gzip file]

Grading Policy:

Grades will be based on completion of all assignments and exams (TBD). Approximate weighting of class work is shown below:

- Exams: 50%
- Homeworks: 50%
This is a course that counts towards graduate degrees, and cannot be taken on a "Credit/NoCredit" basis.

Assignments:

All assignments must be on time, or points will be lost. Specific requirements for each assignment will be explained.
Grades will be based on several things, including:

- timeliness
- attendance & participation in class
- status of working code
- quality of project writeup, presentations, reports, etc.

Notes:

1. There will be 4-5 modules in this class, and each one will have 1-2 homework assignments
2. There will be in-class exams for each module.
3. Please see the Homework page of this course for details.

Add Policy

- Students are added via the SDSU Waitlist System - the instructor for this class has NO control over the list or the order of students who are on the list.
- Enrolled students must attend the first 2 classes or they will be dropped.
- Students must pass an initial exam to remain in the class.

Cheating:

Plagiarism, cheating, and "oversharing" of collaborators or classmates code and work are not allowed. There are ways to credit sources of information and inspiration, which will be clearly described in the lectures. If you don't understand what this means, read these articles:


Your code will be run through a code checker such as Google (you would be amazed what can be found), and the Standford Moss System for Detecting Software Plagiarism. The first time you are caught, you and anyone who shared or worked with on your code/project will get a zero on that assignment. Anyone caught cheating a second time will be reported to the Center for Rights and Repsonsibilities for Academic Dishonesty, and will receive an F based on the academic review process.

Course Links:
Course Links and other informations: http://thomas.sdsu.edu/courses/sp16/comp605/links.html