SI458, High Performance Computing
Course Policy, Spring AY20

Coordinator: Prof. Rick Schlichting, x3-6805, schlicht@usna.edu

Course Description: High performance computing (HPC) aims to solve some of science's most difficult problems using the biggest, fastest computer systems in the world. As parallel systems, they use hundreds of thousands of processor cores working in concert to tackle a single problem. The fundamental question addressed by this course is: how do you program such systems? The course will begin with an overview of the area, including how these systems are used and common system architectures. We will then explore how to write parallel programs for different types of architectures, including shared memory machines, distributed memory architectures, and accelerators like GPUs. This will include structuring principles, as well as practice using standard software packages such as OpenMP, MPI, CUDA, and OpenACC. Lab exercises will involve the use of these technologies on multiple types of systems, ranging from Linux desktops in the lab to large-scale Cray machines with hundreds of thousands of compute cores available as part of the DoD High Performance Computing Modernization Program (HPCMP).

Credits: 2-2-3

Learning Objectives:

- Overview of high performance computing. Understand the the basic characteristics of HPC, as well as application areas and their impact on society.
- Hardware and system architectures. Understand different types of architectures and the tradeoffs between them.
- Parallel programming using shared memory. Understand the basic principles of writing parallel programming using shared memory.
- Visualization. Understand the importance of visualization to HPC and the properties of standard packages.
- Resource management. Understand how HPC machines are typically accessed and the characteristics of systems such as PBS and Slurm.
- OpenMP. Understand how to write programs for shared memory systems using OpenMP.
- Parallel programming using distributed memory. Understand the basic principles of writing parallel programs using distributed memory architectures, and how it differs from approaches based on shared memory.
- MPI. Understand how to write programs for distributed memory systems using the Message Passing Interface (MPI) library.
- Parallel programming using accelerators. Understand the basic principles of writing parallel programs that can exploit accelerators such as GPUs.
- CUDA and OpenACC. Understand how to write programs for GPUs using NVIDIA's CUDA programming model, and the OpenACC standard.
- Application areas and examples. Understand the various areas in which HPC is used, the scale of the applications, and the type of programming models used to achieve the optimum results.

Textbooks: There is no required textbook for this course, but the following are useful references:

- *High Performance Computing*. T. Sterling, M. Anderson, M. Brodowicz, Morgan Kaufmann Publishers, 2018 (ISBN: 978-0-12-420158-3).
- *An Introduction to Parallel Programming*. P. Pachecho. Morgan Kaufmann Publishers, 2011 (ISBN: 978-0-12-374260-5).

Online material will also be used extensively, including the following tutorial from Lawrence Livermore National Lab (LLNL):

- Introduction to Parallel Computing. https://computing.llnl.gov/tutorials/parallel_comp/.

Extra Instruction: Extra instruction (EI) is always available, and I encourage everyone to take advantage of it. I have an open door policy, and I'm usually happy to talk whenever you come by. If you want to be sure I'm available, however, send an email and we'll schedule a time. Students with an appointment for EI have precedence over drop-ins. EI is not a substitute lecture; come prepared with specific questions or problems.

Collaboration: The guidance in the Honor Concept of the Brigade of Midshipmen and the Computer Science Department Honor Policy must be followed at all times. See www.usna.edu/CS/resources/honor.php. Specific instructions for this course:

Labs. The following policies apply to the lab assignments in this class.

- Design and implementation of programs must be the work of the individual student handing in the final product. The coding must be your own, and you must be able to explain it in details to the instructor if asked.
- General collaborative conversations with regard to strategies, techniques, and solution methods are allowed. Any collaboration must be documented (name(s) and brief summary of collaboration).
- Questions to classmates about specific C-related programming issues are allowed. Again, however, the code you submit must be your own.
- Actual code may not be shared with others or copied (in writing or electronically.)
- You may not search for or use solutions from the Internet for the lab assignments.

Homework. Similar to labs, collaborative conversations with regard to strategies, techniques, and solution methods are allowed, but the answers must be the work of the individual student handing in the final product.

Exams. Exams will normally be closed book, individual effort. Use of notes and other materials will be at the discretion of instructor, but the default is no outside material of any kind.

All collaboration and outside sources should always be cited. The same rules apply for giving and receiving assistance. If you are unsure whether a certain kind of assistance or collaboration is permitted, you should assume it is not, work individually, and seek clarification from your instructor.

Classroom Conduct: The section leader will record attendance and bring the class to attention at the beginning and end of each class. If the instructor is late more than 5 minutes, the section leader will keep the class in place and report to the Computer Science department office. If the instructor is absent, the section leader will direct the class. Drinks are permitted, but they must be in reclosable containers. Food, alcohol, smoking, smokeless tobacco products, and electronic cigarettes are all prohibited. Cell phones must be silent during class, and not be a distraction to the owner or others.

Late Policy: Penalties for late submission of graded work for this course is as follows:

- Homework or labs submitted late will have their final graded score reduced by 25% for each 24 hours past the due date and time.
- Fractional days will be rounded up, weekends included.
- Work will not be accepted more than 2 days late without prior instructor consent.

Grading: Grades will be determined as following.

	6 weeks	12 weeks	16 weeks
Labs	80%	70%	55%
Homework	20%	10%	10%
Midterm		20%	15%
Project			20%
Total	100%	100%	100%

The final grade will then be 0.8 times the percentage score for 16 weeks + 0.2 times the percentage score on the final exam

Signed:

Prof. R. Schlichting

Approved:

CAPT M. Bilzor, USN