CSC367H1: Parallel Programming
Winter 2019

Instructor

Maryam Mehri Dehnavi
mmehride@cs.toronto.edu

Course Overview

This course provides discussions on parallel computing and its applications. A combination of lectures, assignments, labs, and the course project will expose students to parallel programming models such as OpenMP, Pthreads, MPI, and CUDA, as well as parallel algorithms. Students will learn how to write fast code for various scientific computations to execute on modern high-performance architectures.

Required Prerequisites:

CSC258H1: Computer Organization, CSC209H1: Software Tools and Systems Programming

Recommended Prerequisites:

CSC369H1: Operating Systems

Sections:

Lectures: Mondays and Fridays 14:00-15:00 PM

Prof. Maryam Mehri Dehnavi

mmehride@cs.toronto.edu

Office hours**:** Mondays 15:30-16:30 PM in Pratt 398A

Tutorial/Labs**:** Wednesdays 14:00-15:00 PM

Discussions:

Piazza for questions and discussions. You will be automatically added to the class Piazza.

Quercus for all course material along with important announcements. You should already be in Quercus.

Important dates:

Item	Release date	Due date
Assignment one	Monday, January 14	Monday, January 28
Assignment two	Monday, January 28	Monday, February 11
Assignment three	Monday, February 11	Friday, March 1
Project	Monday, February 25	Phase 1 (SD!): Monday, March 11 Phase 2: Friday March 22
Assignment four	Monday, March 18	Friday, April 5

Item	Date
In-tutorial test	Wednesday, March 6
In-class test	Monday, April 1

Class Evaluation:

The final grade is based on two in-tutorial/in-class tests, the course project, the assignments, and labs. There will be 4 programming assignments along with a programming project. There are approximately 8 labs.

%	Item
Assignments	40%
Project	20%
Labs	10%
In-tutorial test	15%
In-class test	15%

Late Policy:

All labs are due at 10 PM on the due date. Late lab submissions get 0 marks.

Assignments and the project are due at 10:00 PM on the due date. Check the handouts for final due dates. For assignments and the project each student will have 10 “grace” tokens. Each grace token is worth a 2-hour extension without any penalty. The 10 tokens are for the entire term, not per assignment/project. After you use all the 10 tokens, any late submissions will get a mark 0. For each 2-hours of being late, a token gets deducted from *both* partners. Since a token gets deducted from both partners, if you repartner with someone else for a later assignment, you should be aware that you can only use up to the minimum between the tokens they each if you have left.

Schedule (tentative):

Week	Topics
1	Introduction and overview of memory hierarchies
2	Performance modeling and shared memory
3	Shared memory parallelism and Pthreads
4	Introduction to OpenMP
5	Distributed memory programming and MPI
6	Sources of parallelism and locality
7	Reading week!
8	Performance measurement and parallel algorithms
9	Parallel algorithms continued
10	Introduction to GPUs
11	CUDA programming
12	CUDA programming
13	Introduction to cloud computing

Important policies:

- You have to read the announcements posted in Quercus.
- Violation of Scinet policies will lead to failing the course. You are responsible to read the *Introduction to Scinet* documentation (released during the first week of class) and carefully follow all rules.
- Please do not send email directly to the TAs. They will not be replied. Emails to the instructor will be replied within 48 hours but these emails have to be limited to urgent and personal enquiries.
- Piazza should be used to answer and discuss questions about assignments, the project, and labs. The TAs will be monitoring it.
- Except for the GPU related labs and assignments, all your code and solutions will be graded on the Scinet cluster. We will post announcements if Scinet experiences downtimes. If needed we will ask you to use other computing resources for Scinet downtimes. Carefully follow all announcements!
- Your submissions and code should (1) follow submission instructions carefully to avoid penalties; (2) compile! If your code doesn't compile you will get 0 marks.

Academic Integrity: Plagiarism

- Plagiarism is an extremely serious academic offence with server penalties.
- Do not post your code publically! Your repositories have to be private! Even after the course is over.
- Read the licences in all lab, assignment, and project handouts carefly.
- Do not search for solutions - we will use technology to detect this!
- You should not use code fragments from public places, even if it is open source.
- You and your partner are both responsible for any detected plagiarism.
- Do not give your code to anyone!
- Code or answers to assignment questions should not be posted in Piazza.

Strongly Recommended Readings

- Introduction to Parallel Computing - A. Grama, A. Gupta, G. Karypis, V. Kumar
- CUDA by example: An introduction to General-Purpose GPU programming – J. Sanders, E. Kandroot
- Computer Architecture: A Quantitative Approach- David A. Patterson and John L. Hennessy

Recommended Additional Reading

- An Introduction to Parallel Programming - Peter Pacheco
- Parallel programming in C with MPI and OpenMP - Michael J. Quinn
- The Art of Multiprocessor Programming - Maurice Herlihy and Nir Shavit
- D.H. Bailey, R.F. Lucas and S. Williams 2010. *Performance Tuning of Scientific Applications*, CRC Press.
- G. Hager and G. Wellein 2010. *Introduction to High Performance Computing for Scientists and Engineers*, CRC Press.
- Lin, C., and Snyder, L. 2008. *Principles of Parallel Programming*, Addison Wesley.

- Mattson, T. G., Sanders, B. A., and Massingill, B. L. 2004. *Patterns for Parallel Programming*, Addison Wesley.
- Grama, A., Gupta, A., Karypis, G., and Kumar, V. 2003. *Introduction to Parallel Computing*, Second Edition, Addison Wesley.
- Dongarra, J., et al. 2002. *The Sourcebook of Parallel Computing*, Morgan Kaufmann.
- Goedecker, S., and Hoisie, A. 2001. *Performance Optimization of Numerically Intensive Codes*, Society for Industrial Mathematics.