1 Instructor

Dr. Tom Nurkkala
Associate Professor, Computer Science and Engineering
Director, Center for Missions Computing

Office	Euler Science 211	
E-Mail	`tnurkkala@cse.taylor.edu`	
Phone	765/998-5163	
Hours	T	10:00–11:00
	W	3:00-4:00
	R	9:00-11:00

2 Course Overview

Most—if not all—of the courses you have taken so far in computer science have focused on algorithms and programs for a single processor. This approach to learning makes a great deal of sense. It's plenty hard to write and debug programs on a single processor; why complicate matters with more than one?

For a variety of reasons that we'll discuss, the world hasn't run exclusively on one-processor computers for a long time. Back in 2001, IBM released the first two-core microprocessor—the Power4.[3] As we'll see, since that watershed introduction, multi-core, many-core, and massively parallel processors have taken the world by storm. You must understand these architectures and be able to write and debug programs on them in order to be competitive in the modern computing landscape.

In this course, we will study multi-core CPUs, many-core GPUs, and a variety of tools, technologies, and techniques that will help you design and write software that maximizes the performance of modern computer systems.

3 Learning Objectives

Upon successful completion of this course, you should be able to:

1. Explain the history of high-performance computing and place modern parallel and distributed computing in its historical context.
2. Differentiate parallel computing from distributed computing.
3. Differentiate parallelism and concurrency.
4. Explain Flynn's taxonomy of parallel architectures.
5. Differentiate uniprocessor, many-core, and multi-core computer architectures.

6. Apply decomposition and design strategies for various parallel problems and architectures.
7. Formulate and implement efficient parallel versions of sequential algorithms.
8. Measure parallel algorithm performance.
9. Calculate parallel speedup and efficiency.
10. Employ a threaded model of parallel computation, including common parallel programming constructs like semaphores, shared memory, and monitors.
11. Build correct and performant software for both shared- and distributed-memory multicomputers using OpenMP and MPI.
12. Understand the host-device model for modern, general-purpose graphics processing units (GPUs).
13. Build correct and performant software for GPUs.
14. Construct hybrid parallel programs that execute cooperatively on both CPU and GPU cores.

4 Texts

There is one **required** text:

- Barlas [1]

You may also wish to acquire access to the following **recommended** texts:

- Cheng, et. al. [2]
- Wilt [6]
- Storti and Yurtoglu [5]

You can order the textbook online, but you may also wish to consider joining Safari Books Online [4]. Over the course of the semester, Safari will cost about the same as the price of Barlas, but will give you access to the other resources mentioned above *and* a ton of other great technical books, videos, and other resources. I have been a member of Safari for many years, and I find it to be an invaluable technical resource.[1]

[1]I receive no compensation from Safari for this endorsement.

5 Evaluation

The grading breakdown for the course is shown in table 1. Refer to my *Periodic Table of the Grades* (on Moodle) for the grading scheme. I reserve the right to award a higher grade than strictly earned; outstanding attendance and class participation figure prominently in such decisions.

Category	Weight
Homework	35%
Exam 1	20%
Exam 2	20%
Final	25%
Total	100%

Table 1: Grading details

6 Course Expectations

Following are my expectations regarding the course.

6.1 Attendance

You are required to attend all class sessions. I will be in class each day, and I expect you to be there also.

In general, I am very understanding about students who must miss class due to a sanctioned Taylor activity, medical appointment, job interview, family emergency, and the like. If possible, let me know in advance that you will not be in class; I will work with you to arrange make-up instruction, homework, exams, etc.

6.2 Late Work

All course assignments will include an unambiguous due date. Usually, assignments are due at the beginning of class on the due date. If there are multiple sections of a class, the assignment is due at the beginning of the earliest such section. Barring exceptional circumstances like those mentioned in section 6.1, I expect your work to be submitted *on the due date*. Late work will *not* be accepted.

This policy on late work is intended to prepare you for real-world experience after graduation. In the marketplace, late work is not merely an inconvenience. Missing a deadline may alienate your customer, upset your manager, ruin your project, or terminate your employment! *Now* is the time to learn the self discipline and time management skills required to complete your work when it is due.

6.3 Conduct

I expect you to be prepared, awake, aware, and participatory during class. I will not hesitate to ask you to stand or move if you are distracted or sleepy.

I expect you to join in discussions, respond to questions from me and from your colleagues, and ask questions of me. I expect you to hold my feet to the fire if I am being unclear, unkind, or contradictory.

6.4 Gizmos

You may not use a laptop, tablet, or similar device to check e-mail, engage in social networking, surf the web, or any other activity not directly relevant to current classroom activity. If you use an electronic gizmo during class for legitimate academic purposes (e.g., note taking), be prepared to demonstrate relevant use on demand at any time.

7 Moodle

The Computer Science and Engineering department uses Moodle as our Learning Management System. The URL for Moodle is `https://moodle.cse.taylor.edu`. To sign on to the course site for the first time, you will need an enrollment key. The key for this course is `nerds4christ`.

You are responsible for checking Moodle regularly to keep up with assignment due dates and other announcements. For due dates, *the Moodle calendar is your friend.*

8 Academic Integrity

As a student at an institution whose goal is to honor Christ in all that it does, I expect you to uphold the strictest standards of academic integrity. You must do your own work, cite others when you present their work, and never misrepresent your academic performance in any way. Violation of these standards stains the reputations of you as a student, Taylor as an institution, and Jesus as our Lord. A violation of academic integrity may result in your failing the course and other disciplinary action by the University. Refer to the Taylor catalog for the official statement of these ideas.

References

[1] Gerassimos Barlas. *Multicore and GPU Programming: An integrated approach.* Elsevier, 2014.

[2] John Cheng, Max Grossman, and Ty McKercher. *Professional Cuda C Programming*. John Wiley & Sons, 2014.

[3] IBM. *Power 4: The First Multi-Core, 1GHz Processor*. 2011. URL: https://www-03.ibm.com/ibm/history/ibm100/us/en/icons/power4/.

[4] O'Reilly Media, Inc. *Safari Books Online*. 2016. URL: https://www.safaribooksonline.com/.

[5] Duane Storti and Mete Yurtoglu. *CUDA for Engineers: An Introduction to High-Performance Parallel Computing*. Addison-Wesley Professional, 2015.

[6] Nicholas Wilt. *The Cuda Handbook: A Comprehensive Guide to GPU Programming*. Pearson Education, 2013.