

**CS 380P Parallel Systems**

Lectures:	MW 2:00-3:30 MEZ 1.120
Unique number:	53610
Class Web Page:	<a href="http://www.cs.utexas.edu/users/lin/cs380p/">http://www.cs.utexas.edu/users/lin/cs380p/</a>

	Instructor	Teaching Assistant	Admin. Assistant
Name:	Calvin Lin	Akanksha Jain	Gem Naivar
Email:	<a href="mailto:lin@cs.utexas.edu">lin@cs.utexas.edu</a>	<a href="mailto:akanksha@cs.utexas.edu">akanksha@cs.utexas.edu</a>	<a href="mailto:gem@cs.utexas.edu">gem@cs.utexas.edu</a>
Office:	ACES 3.424	ACES 3.430	ACES 3.422
Office Hours:	MW 3:30-4:30	MWF 3:30-4:30	

**Course Objectives.** This course will take a broad look at parallel computing, touching on topics in computer architecture, programming models, languages, performance evaluation, algorithms, and applications. We will consider both small-scale and large-scale parallelism, and we will consider both traditional (scientific) and modern (e.g., graphics) applications. The specific goals are (1) to learn fundamental principles about parallel computing, (2) to be able to write a parallel program and obtain good performance, (3) to be informed of trends in the area, and (4) to explore in depth one aspect of parallel computing in a project of your choosing.

**Text.** *Principles of Parallel Programming.* Lin and Snyder. Addison-Wesley, 2008, ISBN-10: 0321487907, ISBN-13: 9780321487902.

**Prerequisites.** Familiarity with computer systems will be useful.

**Expectations.** The workload will be fairly heavy. There will be weekly readings, four programming assignments (plus two warmup assignments), an in-class midterm, and one course project. The various components will be weighted as follows:

Assignments:	45%
Quizzes and Midterm :	25%
Project:	30%

**Content.** We'll roughly cover the following material.

- Motivation
  - Traditional motivations for parallelism
  - The changing role of parallelism
- Basic Principles
  - Sources of inefficiency

- Metrics: Execution time, speedup, etc.
- Throughput vs. latency
- Scalability: massive parallelism, Amdahl’s Law, Gustafson’s Law
- Architectures
  - Parallel architectures
  - Trends in architectures, including CMPs, GPUs, and Grids
- Parallel Programming: Low Level Approaches
  - Threads
  - Message passing
  - Issues in scalability and portability
  - Transactional Memory
- Parallel Programming: Higher Level Approaches
  - ZPL
  - Automatic Parallelization and HPF
  - Chapel
  - MapReduce
- Parallel Algorithms
  - Models of parallelism: PRAM, CTA
- Applications

## Programming Assignments

We will have six programming assignments, with the following tentative schedule. For the first two assignments—which will count very little towards your grade—you may work with others, as the goal is to familiarize you with the language and the environment. For the subsequent assignments, you may work in teams of two.

Assignment	Available	Due
Pthreads warmup	Jan 25	Feb 2
MPI warmup	Feb 2	Feb 8
Pthreads: Sparse Matrix-Vector Product	Feb 9	Feb 22
MPI: Sparse Matrix-Vector Product	Feb 23	Mar 6
MPI: Barnes-Hut	Mar 7	Mar 29
Chapel: Barnes-Hut	Apr 6	Apr 12

### Assignment 0: Due January 24

1. Obtain a TACC account: <http://www.tacc.utexas.edu/general/newusers/>  
Scroll down to “Requesting a New TACC Account.”
2. Mail your TACC account ID to the TA.

## Quizzes

At the *beginning* of each class, you will submit a brief written response to the day's assigned reading. For example, for a technical paper, we might ask you to make at least three critical points about the paper's goals, assumptions and execution, or we might ask you to relate the work to some other piece of work that we've read.

We will have random quizzes that test your knowledge of the reading. You may opt out of any two readings (and their associated quizzes) by not submitting a written response.

## The Project

The projects may be done in teams and can take various forms, possibly involving design, implementation, experimentation, analysis, or evaluation. One broad class of projects is to implement some non-trivial parallel program, but more research-oriented projects are possible as well.

Below is a partial list of project ideas, but we encourage you to come up with your own ideas. For those of you who do not have good project ideas, we will provide suggestions.

- Write a parallel program to solve some problem in the language of your choice. Hopefully, the problem is something that you're interested in. Ideally, you will learn something from this project, including one or more of the following:
  - What factors make it difficult to produce a correct program?
  - What factors make it difficult to achieve good performance?
  - Does your solution require any of the following complications?
    - \* Dynamic parallelism
    - \* Pointer-based data structures
    - \* Task parallelism
    - \* Hierarchical data structures
    - \* Novel algorithms

Think about how you will evaluate performance.

Group projects that could compare different languages, approaches or hardware can be particularly interesting.

- Implement GPU support for MPI: Implement an extension of MPI that allows programs to exploit a system's GPUs as well as its CPUs.
- CMPs: Take an application that was designed for a multi-chip parallel computer and retarget it for a CMP. Will drastic changes be required to obtain good performance out of the memory system?
- Implement a parallel computer game with physics simulations. Perhaps use a locally developed thread scheduling system to do this.
- Evaluate design changes to the Chapel programming language.
- GPU Programming: Using the CUDA language, implement and evaluate the performance of some parallel applications running on a GPU.
- Implement a ray tracer in CUDA for a GPU. A multi-person group could implement a ray tracer in both CUDA and in Pthreads and try to make some sort of meaningful comparison of the two in terms of both programmability and performance. There are many issues to resolve when comparing performance across different architectures.

- Write a survey paper on race detection: Write a careful survey of the existing literature in some aspect of correctness tools, such as race condition detection, and suggest directions for future work. (Be careful, it's easy to write a superficial survey.)

The various components of the project are listed below, and I am prepared to offer guidance and help at every step.

1. Pre-proposal: This is a rough idea for a project, including ideas about the project's scope, nature and personnel. This is essentially a starting point for defining the project, and may even be as vague as "I'm interested in parallel debugging, and I only work alone because I'm a sociopath."
2. Proposal: This is a well-thought out plan of action that includes target deadlines for sub-pieces of the project. Some non-trivial amount of thought and/or background reading will be required to move from the pre-proposal to this proposal.
3. Checkpoints: These checkpoints are project-specific goals intended to keep the projects on track.
4. Final Report: The final report will be evaluated based on its presentation as well as its contents.

**Web page.** All announcements and handouts will be placed on the class web page or on the TA's web page.

**Office Hours/Open Door Policy.** My door is usually open when I am in, so please feel free to come to my office whenever my door is open. (If the door is slightly cracked, consider it open.) In the rare case that my door is completely closed, **please do not knock** unless you have an appointment with me.

**Administrative Assistant.** My administrative assistant is Gem Naivar, whose office is across the hall from mine. *See Gem when you need to retrieve unclaimed assignments or handouts.*

## Scientific Ethics

Basically, we'll apply the usual standards of the scientific community: collaboration is encouraged but must be cited. Cheating, including plagiarism, will lead to failure of the course. Please read the following brief document to make sure that you understand the definition of plagiarism:

<http://deanofstudents.utexas.edu/sjs/scholdisplagiarism.php>

Note a few points of emphasis from this document:

- While it's clear that the use of verbatim material without proper attribution constitutes plagiarism, other types of material can be plagiarized as well, such as ideas drawn from an original source or even its structure (e.g., sentence construction or line of argument).
- Minor revisions to borrowed text amounts to plagiarism. So by merely changing a few words or rearranging several words or sentences, you are not paraphrasing; you are plagiarizing.
- Material that is copied or slightly modified is considered to be plagiarized even if the original work is cited as a reference. To avoid plagiarism, you need to both cite the work and make it clear that the text was originally written by others.