Instructor: Vijay Karamcheti  
vijayk@cs.nyu.edu
Lecture: Wednesdays, 5:00pm-7:00pm  
102 CIWW
Office Hours: Tuesdays, 4:00pm-5:00pm  
715 Broadway, Room 704  
phone: 8-3496
Mailing List: To subscribe, follow instructions at  
http://www.cs.nyu.edu/mailman/listinfo/g22_3033_004_fa01  
To post, send e-mail to  
g22_3033_004_fa01@cs.nyu.edu
Web Page: http://www.cs.nyu.edu/courses/fall01/G22.3033-004

Prerequisites
Senior undergraduate-level or introductory graduate-level courses in computer architecture and operating systems.

Description
Parallel computing is a critical component of current-day computing technology, and is likely to grow in importance with the proliferation of multiprocessor PC desktops and servers (consisting of 8-16 CPUs on a shared bus), and scalable clusters of commodity workstations.

This course shall examine the organizing principles behind parallel computing both from an architectural and a programming perspective. The course consists of two parts, organized around a common set of issues relevant to all parallel systems: naming, synchronization, latency, and bandwidth. The first part will discuss how modern parallel computer architectures deal with these issues, both at the small (shared memory multiprocessors) and large (scalable multiprocessors) scales. The second part of the course will discuss how the issues are dealt with in several common programming paradigms including message-passing, shared-memory, and higher-level approaches. The focus in this part of the course will be on both programming expression and programming for performance.

The intended audience for this course is doctoral students with research interests in computer architectures, software systems (programming languages, compilers, operating systems), and applications.

Textbooks (Recommended)
The lectures will draw upon material from two recommended textbooks, supplemented with current research papers. I strongly recommend those of you with research interests in computer architecture, or parallel and distributed processing to invest in these books as a resource for the future.

- David Culler and Jaswinder Pal Singh with Anoop Gupta,  
  *Parallel Computer Architecture: A Hardware/Software Approach*,  
• Gregory R. Andrews,  
*Foundations of Multithreaded, Parallel, and Distributed Programming*,  
Addison-Wesley, 2000.

**Workload**

• **Class:** Students are expected to attend all lectures.

• **Readings:** Students are responsible for completing suggested readings: this would help both in understanding the lecture material, and in conducting meaningful discussions.

• **Homeworks:** There will be five homeworks, four of which will involve writing a simple parallel program and analysing its performance. Tentatively, the homeworks will cover the following topics:
  1. Homework 1: Basic architectural design choices, and how these are affected by technology trends, and requirements of the target market.
  2. Homework 2: Threads-based parallel programming models, and small-scale shared memory architectures.
  3. Homework 3: Message-passing programming models, and large-scale distributed memory architectures.
  4. Homework 4: Shared-memory programming models, and large-scale shared memory architectures.
  5. Homework 5: Remote memory access-based (put/get) programming models, and distributed memory cluster architectures.

Each homework will be due **two weeks** after it has been assigned. I shall make every effort to grade your assignments and return them to you by the following week.

The parallel programming libraries that you will be using (pthreads, MPI, parmacs, and VIA) all have C/C++ bindings, so that will be the programming language used for all of the assignments. You are expected to be very familiar with C/C++. In case you are not, the course will leave you little time to pick up required knowledge on your own during the semester.

• **Project:** All students are required to do the course project (in groups of 2-3). This is a major portion of the overall grade. See the course project handout (Handout #2) for additional details.

The course will cover a lot of ground in a relatively short time, so it is important to keep up with the work. The homeworks and the project supplement the lectures, so you will get significantly more out of the course by registering for it rather than just auditing it.

**Supercomputing Resources**

For the homeworks and the class project, each student will receive accounts on at least two parallel machines—the SGI Cray Origin 2000 at the National Center for Supercomputing Applications (NCSA) in Champaign, IL, and a Beowulf PC cluster at NYU. The former enables experimentation with both large-scale distributed memory and shared memory architectures, while the latter is a distributed memory machine made up of both SMP and uniprocessor nodes. Both machines support multiple parallel programming models—small-scale threads programming, message passing, and shared memory.

Both the hardware and software environments on these machines are likely to be less robust than the uniprocessor environments most of you are used to. So, plan on starting early for each of the assignments.

Note also that accounting for resources on the Origin 2000 is done on a per-class basis, which requires each of you to be careful about monitoring program resource utilization. A later handout will provide details about logging into these systems, developing, debugging, and running programs on them, and monitoring resource utilization.
Policies and Grading Criteria

Final grades will be computed based upon the following weights: **Homeworks 60%** (HW1: 6%, HW2: 8%, HW3: 12%, HW4: 12%, HW5: 12%), **Project 40%**. See the project handout (Handout #2) for details about a breakdown of the project grade.

_There are no exams_, so it means that you are required to turn in each homework on time, as well as complete all project requirements in order to receive a grade for the course. That being said, _if you're worrying about grades, you're missing all the fun!_

Important Dates

Homeworks/project writeups are assigned or due on the following dates. Note that no homeworks have been assigned in the last 4 weeks of the course to allow you to work on the project.

- **09/06:** HW1 assigned
- **09/19:** HW1 due, HW2 assigned
- **10/03:** HW2 due, HW3 assigned
- **10/10:** 2-page project proposal due
- **10/17:** HW3 due, HW4 assigned
- **10/31:** HW4 due, HW5 assigned
- **11/14:** HW5 due
- **11/28:** First part of project report due
- **12/05:** Project presentations
- **12/12:** Final project report due

Syllabus

A tentative syllabus is detailed on the following sheets. I may modify the order and/or emphasis given to specific topics based on class interest and reaction. The topics can be broadly divided into three categories: parallel programming models (Lectures 2-3), parallel architectures (Lectures 4-7), and programming for performance (Lectures 8-11). The latter category explores the interaction between the programming model and the underlying architecture.