G22.3033-003: Architecture and Programming of Parallel Computers
Handout #1: Course Information

Instructor: Vijay Karamcheti
vijayk@cs.nyu.edu

Lecture: Thursdays, 5:00pm-7:00pm
102 CIWW

Office Hours: Fridays, 11:00am-12:00noon
715 Broadway, Room 704
phone: 8-3496

Mailing List: g22_3033_003_#98@cs.nyu.edu


Prerequisites

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

Description

Parallel computing is a critical component of the computing technology of the 90s, and is likely to grow in importance with the proliferation of multiprocessor PC desktops and servers (consisting of 2-8 Pentium II’s 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, data-parallel, as well as 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*
• George Almasi and Allan Gottlieb,  
  *Highly-Parallel Computing*, 2nd Edition  

**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 four homeworks, three 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: Data 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.

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

• **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**

To use for the homeworks and the class project, each student will receive accounts on two parallel machines—the Convex Exemplar, and the SGI Cray Origin 2000—at the National Center for Supercomputing Applications (NCSA) in Champaign, IL. The former provides a small-scale SMP architecture, while the latter enables experimentation with both large-scale distributed memory and shared memory architectures. Both machines support all three dominant parallel programming models—data parallel, message passing, and shared memory.

Note that accounting for these resources 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 50%** (HW1: 10%, HW2: 10%, HW3: 15%, HW4: 15%), **Project 50%**. 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 6 weeks of the course to allow you to work on the project.

09/10: HW1 assigned
09/24: HW1 due, HW2 assigned
10/08: HW2 due, HW3 assigned
10/15: 2-page project proposal due
10/22: HW3 due, HW4 assigned
11/05: HW4 due
11/19: 1st part of project report due
12/10: Project presentations
12/14: Final project report due.

Syllabus

The following is a tentative syllabus. 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.

9/10 Lecture 1 Introduction
why parallel computing, motivating applications,
history and convergence, course organization

9/17 Lecture 2 Parallel Programs
decomposition, assignment, orchestration, mapping
case studies: Ocean, Barnes, Ray Tracing, Data Mining
parallel system components: architecture, OS and compilers,
programming models, applications

9/24 Lecture 3 Models of Parallel Computation
analytical: PRAM, LogP
operational: data parallel, message passing, shared memory
common issues: naming, synchronization, latency, bandwidth
tutorial: data-parallel programming

10/01 Lecture 4 Small-scale Shared Memory Machines
bus-based architectures, snoopy caches
case study: Convex Exemplar
tutorial: programming with threads

10/08 Lecture 5 Large-scale Distributed Memory Machines
scalable networks, processor-network interfaces
support for put/get, remote memory access  
case study: Cray T3E  
tutorial: *programming with MPI*

10/15 Lecture 6 **Large-scale Shared Memory Machines**  
directory-based coherence  
case study: SGI Origin 2000

10/22 Lecture 7 **Large-scale Shared Memory Machines (contd.)**  
programmable protocol processors  
case study: Stanford FLASH

10/29 Lecture 8 **Programming for Performance**  
common issues: load-balance, synchronization, locality  
data-parallel models: compilation technology, data layout

11/05 Lecture 9 **Programming for Performance (contd.)**  
message-passing models: messaging layers, multithreading,  
message pipelining and aggregation

11/12 Lecture 10 *(this lecture will be rescheduled)*  
**Programming for Performance (contd.)**  
shared memory models: lock aggregation, dynamic task assignment,  
layout to minimize cache conflicts

11/19 Lecture 11 **Hardware/Software Tradeoffs**  
shared virtual address space  
latency tolerance for various communication abstractions

11/26  
— **Thanksgiving Vacation** —

12/03 Lecture 12 **Future Directions**  
hardware and software trends  
high-level programming models and compilation issues  
parallel programming tools  
concluding remarks

12/10 Lecture 13 **Project Presentations**