Topics in Distributed Computing:
Lowly Parallel Computing
CS 756H, Winter 1997

Instructor: Thomas Kunz
DC 3508, x4820, tkunz@uwaterloo.ca

The subject of this course are **lowly** parallel computers. Such computers are characterized by the following two main features: few processors and a relatively slow interconnection network. A concrete example of such a parallel machine is a network of Unix workstations, an alternative name for such an arrangement is **cluster**.

Particular emphasis is given to the issue of programming such machines to achieve high performance. While parallel and distributed machines offer many potential advantages over a traditional sequential computer, such as increased reliability, incremental extensibility, and a good fit to "naturally" distributed applications, the prime motivator for *going parallel* is the speedup of an application. In many research labs around the world, clusters or farms of off-the-shelf workstations are used to execute complex numerical and other applications (such as the Grand Challenge Problems). The use of off-the-shelf hardware (for processors as well as the network) helps to keep the costs down and enables the end-users to install many of these farms. Today, even relatively small companies can afford to install a small network of workstations to achieve turnaround times for their complex simulation models are below one day.

While setting up such a lowly parallel computer is trivial, programming it to achieve maximal (or even reasonable) performance is everything but. A lot of research effort has been invested in developing appropriate programming languages, run-time environments, and development tools to facilitate the coding effort. The course will focus on the different programming environments proposed for lowly parallel computing. We will discuss distributed shared memory environments, message passing environments, and others (such as Linda or distributed objects).

The course is organized as follows. To get everyone up to speed, we will start with about one month worth of lectures. The current set of lecture topics is as follows:

1. Introduction and examples of workstation clusters/farms
2. Hardware issues
3. Performance models and metrics
4. Performance comparisons: clusters vs. multiprocessors, workstation vs. PC cluster
5. Programming environments: message-passing
6. Programming environments: shared memory
7. Programming environments: others
8. Parallelizing an existing application
9. Parallelizing the same application for two different programming environments

Then, I expect every student to give a full-length presentation in class on a topic related to the course. Possible topics are the review of a specific programming language or environment, tools that simplify the development effort, methods to parallelize existing applications, hardware issue, experience reports with the deployment of parallel machines, etc.

The majority of the marks will be given for the course project. Each student (or potentially a group of two students, depending on the number of students who register for credit) will be given the same set of applications, called the **Cowichan II** problems. This is a set of 14 small applications designed to be used in assessing and comparing parallel programming systems. If people work independently, the problem set will be reduced. Sequential implementations of these applications in C will be available. The task of each group is to implement and tune these applications for maximum performance on a network of Unix workstations (IBM RS6k workstations, running AIX 4.1 and connected by FDDI, in our lab). Each application will be implemented twice, using different parallel programming environments (PVM plus a not-yet-decided distributed-shared-memory system). The final report should include a description of the implementation, results from performance measurements, and an analysis of the different implementations in light of the performance results achieved. At the end of the term, each individual or group will give a short (30 minutes) presentation about their project, the solutions, and results.

The marks will be assigned as follows: 30% for the in-class presentation, 15% for the presentation of the project results, and 55% for the project itself (based on the software, performance achieved, and project documentation).

Given the emphasis on concrete, hands-on programming, students should have a solid background in parallel programming. For example, I will assume that the concepts taught in CS342, Concurrent Programming, are familiar to everyone in the course.

Literature:

- Gregory F. Pfister, *In Search of Clusters: The Coming Battle in Lowly Parallel Computing*, Prentice Hall, New Jersey, USA, 1995. ISBN 0-13-437625-0.
- Ian Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1995. ISBN 0-201-57594-9. Also available on-line, at URL `http://www.mcs.anl.gov/dbpp`.
- Description of the original (and too complex) Cowichan problem set: `http://www.cs.vu.nl/~bal/cowichan.html`. While this course will use the simplified set of problems described in the next bullet, the papers found under this URL give an indication of the development and motivation behind the problem set.
- Description of the simplified set of Cowichan problems used in this course: `http://ccnga.uwaterloo.ca/~tkunz/cowichan1.ps.Z` and `http://ccnga.uwaterloo.ca/~tkunz/cowichan2.ps.Z`
- Periodicals to look for papers relevant to this course:
  – IEEE Parallel & Distributed Technology
  – IEEE Computer
  – IEEE Transactions on Parallel and Distributed Systems
  – Proceedings of ACM Supercomputing
  – Proceedings of the n-th IEEE International Conference on Distributed Computing Systems
  – Proceedings of the n-th IEEE Int. Symposium on High-Performance Distr. Computing
  and similar conferences and workshops on Parallel and Distributed Computing