

Computer Science 445—Parallel Processing

Department of Mathematics and Computer Science
Macalester College

What it is About

Most systems at Macalester are single processor one instruction at a time computers. Although fast (the machines in our lab can do about 500 million operations per second) they are not fast enough to solve many of the important, real world problems that researchers are investigating today. For example, 72-hour weather forecasting or generating high quality computer animations can require the execution of as many as 10^{15} or 10^{16} operations. Even at 500 million operations per second, a problem of that size would take about 8 months to complete! Most of us would not be willing to wait that long for our result. The only feasible solution is parallel processing—having multiple computers work cooperatively to solve a single problem. For example, if we had 500 processors, rather than 1, work on our weather forecast, then we could reduce the time it took to get our answer from 8 months to 11 hours—certainly a more reasonable value. Today, parallel processing is an essential computational tool for solving large-scale problems in such diverse areas as medical research, imaging, space science, environmental modeling, econometrics, and artificial intelligence.

What We Look At

The field of parallel processing is made up of three distinct areas—parallel architectures, parallel languages, and parallel algorithms—and in this class we take a look at all three. We study a number of different ways to build parallel hardware, including cluster computing—linking together independent, stand-alone machines. We then learn two different parallel languages, MPI and Linda, that can run on these machines,. Both of these languages are available in our Linux lab, so students can implement real parallel programs and measure actual speedups. Finally, we investigate a number of interesting parallel algorithms drawn from a range of interesting areas such as computational modeling, imaging, and scientific problem solving.

Projects/Examples

The following projects or examples have been used in recent semesters:

- Use the MPI programming language to design and write a parallel matrix multiplication algorithm used in computer graphics.
- Use the Linda parallel programming language and 16 computers in the Unix lab (256 Olin) to implement an edge finding algorithm used in imaging and animation
- Write a sorting problem using Quicksort, the fastest known sequential sorting technique. Then implement Parallel Merge Sort and run it using 16 computers. Did you get a 16-fold improvement in the running time? If not, what were the factors that hampered your ability to obtain the maximal speedup?