CSCI 3366 (Introduction to Parallel and Distributed Processing), Spring 2005

Guidelines and Requirements for Projects

1 Overview

One of the requirements for this course is completion of a project. You may work individually or with one other person in the class. The project will count as 100 points of your total grade. It should be more ambitious than one of the homeworks, and if two people work together, the project should be about twice as ambitious as a solo project. All projects must be approved in advance by the instructor, who will be the final arbiter of whether the topic and level of difficulty are appropriate.

2 Suggestions for topics

Possible project topics include the following, or you may propose something else. If your project involves writing code, you may use any language/library that can be run on the department's network of Linux machines.

2.1 Parallel applications

Your project could be the design and implementation of a non-trivial parallel application. There are many, many possibilities here, mostly falling into one of two categories:

- Applications that use multiple processes/threads to improve performance. The textbook by Quinn has problems at the end of each chapter; those might be a good source of ideas. You should plan to collect at least minimal performance data for your application.
- Applications that use multiple processes/threads because they're inherently parallel. This category includes what are often referred to as "classical synchronization problems" (e.g., the bounded-buffer problem discussed in class). You should plan to demonstrate as well as possible that your application really solves the problem (e.g., for the bounded-buffer problem a process trying to read from an empty buffer waits).

2.2 Performance experiments

Your project can consist of a set of experiments designed to measure something about a parallel-programming platform or platforms, such as one of the following.

- Compare different languages/libraries, e.g., MPI versus Java RMI. For example, you might implement the same algorithm using two or more languages/libraries and compare the two implementations, with regard to both performance and ease of programming. Cross-language comparisons might compare both absolute performance (different implementations, same number of processes/threads) and scalability (different numbers of processes/threads, same implementation).

- Compare different algorithms. For example, you might compare the performance of some of the MPI collective-communication library functions (`MPI_Bcast()`, etc.) with user-written functions to accomplish the same things.
- Measure characteristics of the hardware/software platform. For example, you might measure the average time required to send a message and how it varies (if at all) depending on message length, identities of sending and receiving process, processor speed, etc.

3 What to turn in and when

Milestone	Points	When due	Description
Project proposal	5 points	April 19 at 5pm	A brief description of your project topic, no more than a paragraph, in the form of a short e-mail to the instructor. (Plain text is preferred over proprietary word-processor formats.)
Project status report	5 points	April 28 at 5pm	A brief report on progress to date, describing any problems or questions, again in the form of a short e-mail to the instructor.
Final written report	40 points	May 10 at 5pm (not accepted late)	A brief report (no more than five pages should be required, and two or three will suffice for many projects) describing your project's goals and outcome, in hard-copy form. It should address the following topics and include bibliographic references as appropriate. (1) Describe what problem you are solving and how (i.e., recap your project plan, including the design of your application, experiments, etc.). (2) Discuss your results, including graphs and tables as appropriate (e.g., to show performance as a function of number of processes/threads). (3) Describe any unusual or interesting difficulties you encountered, and/or what you learned from doing the project.
Source code	50 points	May 10 at 5pm (not accepted late)	Complete working source code for any program(s) you wrote as part of your project, submitted electronically as for homework. Be sure your code is readable and well-documented.