CSCI 3366 (Introduction to Parallel and Distributed Processing), Spring 2002

Guidelines and Requirements for Projects

Contents

1 Overview **1**

2 Suggestions for topics **1**
2.1 Parallel applications 1
2.2 Performance experiments 2

3 What to turn in and when **2**

1 Overview

One of the requirements for this course is completion of a project. The project will count as 100 points of your total grade; i.e., it counts as much as an exam and about 2.5 times as much as a homework. Students may work individually or in groups of two. A project can consist of writing a non-trivial parallel program, or performing a set of experiments exploring some aspect of parallel programming, or anything else relevant to the course and approved by the instructor. It should be something about 2.5 times as ambitious as one of the 40-point 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, but are not limited to, the following. 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 can be the design and implementation of a non-trivial parallel application. There are many, many possibilities here, most if not all of which fall into one of the following two categories.

- Applications that use multiple processes to improve performance. See the problems and the end of each chapter in the textbook for some ideas. You should plan to collect and analyze at least minimal performance data for your application.
- Applications that use multiple processes 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 processors) and scalability (different numbers of processors, 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 thing.
- 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 11 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 plan	5 points	April 18 at 5pm	A more detailed description of your project, up to a page, again in the form of an e-mail to the instructor. (Plain text is preferred over proprietary word-processor formats.) If you are writing an application, describe what problem you are solving and the design of your parallel algorithm. If you are conducting performance experiments, describe what you are trying to measure and the experiments you will use to measure it.

continued on the next page

Milestone	Points	When due	Description
Final oral presentation	20 points	May 4, 9:30am – 11:30am (the time slot for this course's final, starting an hour late)	An oral presentation, up to 10 minutes, about your project. In this presentation, you should: (1) Describe what problem you are solving and how (i.e., recap your project plan, including the design of your application, experiments, etc.). (2) Present your results. Depending on your project, this might involve showing experimental results, demonstrating a running application, etc. (3) Describe any unusual or interesting difficulties you encountered, and/or what you learned from doing the project.
Final written report	35 points	May 6 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. This should be an expanded version of your oral presentation, describing your project's goals, design, and results. It should include bibliographic references as appropriate.
Source code	35 points	May 6 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.