CSCI 3366 (Parallel and Distributed Processing), Fall 2017
Project

Credit: 50 points.

1 Honor Code Statement

Please include with each part of the assignment the Honor Code pledge or just the word "pledged", plus one or more of the following about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For written assignments, it should go right after your name and the assignment number; for programming assignments, it should go in comments at the start of your program(s).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

2 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 50 points of your total grade. It should be comparable in length and difficulty to the longest of the homeworks (Homework 2), 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.

3 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.

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

3.1 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. 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.

3.2 Programming environments

Your project could focus on comparing/contrasting different languages/libraries, possibly including ones we didn't discuss in class, or discussed only briefly. The idea here is similar to the first suggestion under "Performance experiments" but with the focus on programmer ease of use rather than performance. If you choose this option and include in the mix of languages/libraries something we didn't discuss in class, you should include in your project a short discussion of its major features — something along the lines of the discussion in class of how Java sockets/RMI and POSIX threads map onto the *Implementation Mechanisms* framework.

A hybrid project combining this idea with the ones in "Performance experiments" might also work well.

3.3 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). You could include in the mix of languages/libraries ones we didn't do much with in class.
- 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.

4 What to turn in and when

- Project proposal (5 points)

  Due before you begin serious work on what you propose. A brief description of your project topic, no more than a paragraph. Submit by e-mail, and allow 24 hours for a response.

- Written report (20 points)

  Due December 14 at 11:59pm. A brief report (two or three pages should be enough) describing your project's goals and outcome. It should address the following topics and include bibliographic references as appropriate:

  - Describe what problem you are solving and how: 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.
  - Discuss your results, including graphs and tables as appropriate (e.g., to show performance as a function of number of processes/threads).
  - Describe any unusual or interesting difficulties you encountered, and/or what you learned from doing the project.

  Normally I would ask for these in hardcopy, but I think this year I'd prefer that you send them by e-mail. You can make my job a bit easier if you send me PDF and not a Word file or the like.

- Source code (25 points)

  Due December 14 at 11:59pm. 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.