4003-531/4005-735	Parallel Computing I	Handout 2
Term 20102		November 30, 2010

Term Team Project

Overview

Students will underake a term project in teams of 2 or 3 students. The term project will consist of proposing a topic for investigation, developing a parallel program and measuring its performance characteristics, submitting the developed software and a written report, and giving a presentation and demo during class. Students taking the course for graduate credit will be expected to include a more significant research component than those taking the course for undergraduate credit.

Team Formation

By December 6, form a team of two or three students to collaborate on the term project. A team of three students is strongly encouraged.

Students taking the course for undergraduate credit (4003-531) must form a team with other students taking the course for undergraduate credit. Similarly, students taking the course for graduate credit (4005-735) must form a team with other students taking the course for graduate credit.

One member of the team must e-mail the instructor with the names of the team members. (Optionally, include a team name.) Project presentations will be scheduled in the reverse order of team formation; that is, the first team to form will be the last team to present (and the earlier your form your team, the longer you will have to prepare your presentation).

Students that fail to form a team will be randomly formed into teams.

Proposal

By December 13, propose a topic for investigation. The topic should be a computational problem (or class of related computational problems) that requires significant computation to solve (and, therefore, a parallel program could potentially solve the problem in less time). See **Suggested Topics** for some suggested topics. Your team will implement both a sequential program and a parallel (SMP or cluster) program to solve the problem and measure and compare their performance. **4005-735 Only:** Your team will also conduct a literature search and analyze research papers related to the problem.

Write a one to two page proposal for your project. The proposal should include the following information:

- A description of the computational problem.
- **4005-735 Only:** Complete citations to two or more *research* papers that you plan to analyze. The papers must be published in refereed research journals or conference proceedings and must be dated 2000 or later (i.e., published in the last 10 years). For help with your literature search, make use of the Wallace Library's Computer Science Databases search and/or contact the GCCIS specialist reference librarian, Roman Koshykar.
- Complete citations to any papers, books, web sites, etc. with information about your problem that you plan to use.
- A description of the sequential and parallel programs you plan to deliver.
- A description of the performance metrics you plan to measure.

Submit the proposal, in PDF format, to the `Project Proposal` Group Dropbox on MyCourses by the due date.

Development

Throughout the quarter, your team should make steady progress on your project. You will develop a sequential program to solve your problem. You will also develop a parallel (SMP or cluster) program to solve your problem. While it is *strongly* recommended that you develop your programs in Java using the Parallel Java library, it is not required. What is required is that your sequential and parallel programs be executable on one of the RIT Computer Science Department's parallel computers (see Parallel Java on the RIT CS Parallel Computers). You will acquire or develop suitable inputs for your programs, measure your programs running times on the inputs, and compare their performance.

As you develop your programs, be mindful of following good software design principles. You should also write a developer's manual (i.e., exact instructions for how to build the software) and a user's manual (i.e., exact instructions for how to prepare the input, how to run the software, how to interpret the output, etc.). These manuals should be plain text files named `DevManual.txt` and `UserManual.txt` respectively. (These need not be lengthy documents, but sufficient for someone not familiar with your software to build and use it.)

Software

By February 14, submit the developed software.

Submit the software artifacts, including all source code, developer's and user's manuals, test cases, input files, output files, test results, etc. that were developed during your investigation, as a single `.zip` file, to the `Project Software` Group Dropbox on MyCourses by the due date.

Report

By February 14, submit a written report. The report should be a complete description of your investigation. The report should include the following information:

- A title.
- Your names, e-mail addresses, and affiliations.
- An abstract (a one or two paragraph summary of the report).
- A description of the computational problem, including background and related work.
- **4005-735 Only:** Analyses of two or more research papers. A thorough analysis of a research paper should include:
  - A description of the specific problem or problems the research paper is addressing.
  - A description of the novel contributions the research paper makes toward solving those problems.
  - A description of how you used the research paper's results in your investigation.
- A description of your solution to the problem, describing the software design and any special characteristics of the sequential and parallal programs.
- A thorough comparision of the performance of the sequential and parallel programs. This comparison should include a description of the performance metrics that were measured. Be sure to include Speedup, Efficiency, and EDSF results as either figures or tables.

- A conclusion that summarizes what was learned from the investigation and discusses directions of possible future work.
- A bibliography of works cited.

The report must be typed, single-spaced, 12pt font, 1in margins, and between 6 and 12 pages in length (including figures, tables, references). The report must be in clear English prose, utilizing proper spelling and grammar.

Submit the report, in PDF format, to the `Project Report` Group Dropbox on MyCourses by the due date.

Presentation

During Weeks 9 and 10, give a presentation and demo during class. The presentation should be a brief synopsis of the information in the report. The presentation should be prepared electronically (e.g., PowerPoint, OpenOffice, KeyNote, Beamer); however, note that a demonstration of your software will necessarily occur off of the slide deck.

Teams of students taking the course for undergraduate credit (4003-531) should expect approximately 20mins for the presentation. Teams of students taking the course for undergraduate credit (4005-735) should expect approximately 25mins for the presentation.

Submit the presentation materials to the `Project Presentation` Group Dropbox on MyCourses by the due date.

Grading

Grades for the project will be assigned based on the following grading scheme:

Team Formation:	10.0%
Proposal:	15.0%
Software:	20.0%
Report:	35.0%
Presentation:	20.0%

All members of a team will receive the same grade for all components of the project.

Important Dates

December 6 (Mon.):	Project team formation due
December 13 (Mon.):	Project proposal due
February 14 (Mon.):	Project software and report due
February 15&17 (Tue.&Thu.):	Project presentations
February 17 (Thu.):	Project presentation materials due

Suggested Topics

- Sorting: bubble sort, merge sort, quick sort, bitonic sort, shell sort, odd-even merge sort, odd-even transposition sort, rank sort, counting sort, radix sort, sample sort
- Searching: depth-first search, breadth-first search, best-first search, A* search
- Dynamic programming: shortest-path, 0/1 knapsack, longest common subsequence
- Graph algorithms: single-source shortest path, all-pairs shortest paths, minimum spanning trees, transitive closure, connected components

- Matrix algorithms: matrix-vector multiplication, matrix-matrix multiplication, matrix inversion
- Solving a system of linear equations
- Parallel data structures (sets, maps)
- K-means clustering, K-nearest neighbors classifying
- Delaunay triangulation
- Image processing
- Parallel ray tracing
- N-body simulation
- Protein sequence querying
- Genetic algorithms
- Hill climbing
- Fast Fourier transform
- Parallel file systems
- Parallel data management
- Parallel programming utilities

Other suitable topics may be found by scanning the course text book, related books from the library, or books from the bibliography.