University of Central Florida
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

COP 6616 Multicore Programming
Fall 2011

Professor Damian Dechev August 27, 2011

RESEARCH PROJECT GUIDELINES

I. Schedule

a) Problem Definition and Progress Report
Due Date: Friday, Sept. 9th

Purpose: state the problem, describe briefly the related work, explain the approach taken to solve the issue and the anticipated results, describe the progress up to that point and the remaining tasks necessary to successfully complete the project.

b) Presentation

Date: one 10-35-minutes presentation per group, scheduled between September 26th, 2011 and December 3rd, 2011.

Purpose: discuss the targeted problem and its solution with the class, introduce to the class the key issues and challenges in your selected field and the critical enabling technologies, include a mini-tutorial in your presentation of at least one critical technology related to parallel and distributed computing that is pivotal to the success of your project.

d) Mid-term Research Report

Due Date: Friday, October, 14th, 2011.
Purpose: research paper and source code of the project up to this point of time. Please see Section II for detailed description and requirements. In an Appendix section, briefly list the challenges, the set of completed tasks, and the set of remaining tasks for the project.

c) Final Report

Due Date: Friday, November 18th, 2011 (this deadline can be extended only by a Temporary Incomplete authorized by the student's residential college dean).

Purpose: final course project paper and source code, please see Section II for detailed description and requirements.

I expect that successful final projects will result in a submission to a competitive international conference or workshop in parallel programming, software engineering, or a related field.

A list of relevant conferences is available and will be constantly updated at the COP 6616 Webcourses/Conferences folder. It is recommended that we pursue an initial conference submission prior to the end of the semester (December, 2011), preferably in November.

II. Description and Requirements

The purpose of the project is to give you a chance to explore in some depth a topic that is related to the subject matter of this course. The project consists of two parts, an activity and a report. Your final report must present original work on a challenging problem. Your solution should be innovative and should either 1) not be described in the literature or 2) present a clearly defined improvement over an existing solution. My recommendation is for you to rather pick a problem that is difficult and ambitious than a small and easy to complete task.

1. The activity involves scholarly work—exploring a topic, reading relevant literature, writing software and running experiments, and so forth—and must include either a substantial amount of programming or a strong theoretical mathematical treaty of issues related to multicore programming.

2. The report is a short paper (8–15 pages) describing the results of the activity and should reflect the substantial amount of work that you put into the activity. The paper should make clear what is the problem, the state-of-the art in this research area, a discussion of related work, a clear definition of the contributions of your paper, a detailed discussion of your algorithms and approach and their application, and performed experimental results. The report will be graded for writing quality as well as for technical content, so attention should be paid to organization, grammar, spelling, and scholarly style. All references used, both text and code, must be properly cited in the bibliography.

3. Source code: typically in the course of your research you will develop source code for your algorithms and experimental results. Please include with your report:
a) the source code of your algorithms along with a brief README file
b) the source code of your experiments along with a brief README file
c) the source code of your paper (LaTeX preferred).

Submission:
All materials should be submitted online using Webcourses
(https://webcourses.ucf.edu/) as a single zip or rar archive.

Format:
All papers should follow either the ACM or the IEEE Manuscript Format.
A sample IEEE template and further formatting instructions are available here:
http://www.ieee.org/conferences_events/conferences/publishing/templates.html

III. The Report

a) What Makes a Good Essay?

Past experience has shown that many UCF students do not know how to write an essay or scholarly paper. I can't really tell you how to write a paper in this brief handout, but the section title is intended to remind you that a project report is an essay of sorts, and the things you have learned in other courses about logical organization, writing style, use of proper grammar and spelling, and proper methods of citing other people's work all apply here.

Like an essay, the project report should have some ideas of your own to report. It should reach some conclusion, and it should give logical arguments and relevant data to support that conclusion. What I do not want in a paper is a simple paraphrasing of somebody else's work. Quoting other people's work as a way to make a point is perfectly acceptable, if properly attributed; simply copying their work, attributed or not, is not acceptable. I want to read about your ideas, your code, and your conclusions, not somebody else's. But of course you will rely on other people's work to support your arguments.

b) Required Format of the Report

Your paper should follow accepted guidelines for scholarly work in computer science. As already stated, all papers should follow the ACM or IEEE Manuscript Format. The paper should begin with title, author, and abstract. The body of the paper should be divided into logical sections appropriate to the structure of the material. Each section should have a numbered section heading. Numbered and unnumbered subheadings should be used where appropriate. Figures and tables should have captions and should be referenced by number. Related work should be cited in the text, and full reference information should appear at the end in a bibliography. Any material copied verbatim should be enclosed in quotation marks as well as being properly cited. The bibliography should contain full citation information, including author, title, year, and publication data. If the publication is a conference proceedings, then the proceedings title, editor, organization or publisher, etc. should be included. If it's a book, then the publisher should be

included. If it's a web page, then the URL and sponsoring organization should be mentioned.

c) Writing Tools

Most research papers in computer science are prepared using the LATEX typesetting system. Because this is a computer science course, and because it is educational for you to learn to use the tools of the field, I am requesting that you too prepare your paper using LATEX and that you prepare your references using the companion tool, BibTEX. Exceptions require the instructor's prior approval.

Compared to papers produced on a typical word processor, the results from LATEX are much more professional looking. LATEX automatically numbers pages and sections, automatically produces a table of contents and list of figures (if desired) and generates page headers and cross-references. BibTEX takes information that you put into a bibliographic database and formats it according to commonly-accepted styles. You don't have to remember whether the article title or journal title should be italicized, or where commas and periods are needed—it does all that for you. But the big win comes when typesetting equations and other mathematical notation. LATEX produces professional-quality typeset equations and formulas rather painlessly. Attempting to do the same in a word processor is clumsy at best, and the results are generally disappointing.

d) Hints

Start now! Locate your resources. Many but not all papers are available on the web. Make sure your project is doable in the amount of time available. Your proposal should identify the resources necessary to carry out your project.

IV. Project Suggestions and Hints

Almost anything related to the course material is acceptable as a project topic. If you are completely out of any ideas and need someplace to start, here are a few project suggestions or simply hints for brainstorming:

1) The implementation of lock-free highly concurrent data structures and algorithms such as lists, queues, stacks, vectors, maps, etc. As a start, you can take a look at the available data structures and algorithms in the C++ Standard Template Library (STL).

2) Find an algorithm or application that would benefit from the use of a multicore nonblocking data structure (hash table, queue, vector): such algorithms could be: parallel graph traversal and algorithms, branch-and-bound search, discrete event simulation, data mining algorithms, computer vision algorithms, scientific computation codes, large graph analysis.

3) The application of program analysis for automatic parallelization, such as the use of an open compiler infrastructure such as ROSE (http://www.rosecompiler.org/) or POET (http://bigbend.cs.utsa.edu/poet/index.php) to translate old code relying on mutual exclusion to lock-free code.

4) The use of a compiler optimization scripting tool such as POET to perform multicore-specific program optimizations.

5) The use of formal verification and validation tools such as model checkers and theorem provers (such as the SPIN Model Checker, Alloy Anayzer, TLC, and many more) to validate key properties of concurrent algorithms.

6) The use of static and dynamic analysis platforms and performance modeling tools to validate concurrent algorithms. Examples of such tools are Valgrind and Intel Pin. For instance, one possible project is to use Intel Pin to analyze the occurrence of the ABA problem in certain lock-free data structures.

7) A visualization tool or an enhanced IDE for concurrent code and data structures that helps for the development and debugging of parallel code.

8) The use of event driven simulators to study the scalability of a certain parallel algorithm. Such algorithms are useful when we solve large graph problems that arise from analysis of large data sets, e.g web search, facebook, cloud computing, etc.

9) Exploring the programming language evolution with respect to the multicore computer architecture revolution. You can study the new concurrency features in C++0x, perform a comparison study between languages, analyze the use of a new language feature with respect to a particular application, explore some new programming languages and their use (such as GO).

10) Explore the problems of memory management and allocation in a concurrent environment.

For more ideas, you can read some papers from the following research groups:
http://research.microsoft.com/en-us/um/people/tharris/
http://www.cs.rochester.edu/~scott/
http://www2.research.att.com/~bs/papers.html
http://habanero.rice.edu/Habanero_Home.html
http://srl.cs.berkeley.edu/~ksen/doku.php?id=projects
http://labs.oracle.com/people/moir/
http://www.research.ibm.com/people/m/michael/pubs.htm
http://www.rosecompiler.org/ROSE_HTML_Reference/ProjectPublications.html