CIS4930/COP5618 Concurrent Programming Project

Important Dates

- **As soon as possible but no later than March 14:**
  - Submit proposal for approval (submit ONE proposal per group)
- **April 21**
  - Poster session for CISE4930 students during class. Attendance is mandatory for ALL students, both CISE 4930 and COP5618.
  - Source code and electronic version of poster submission for CIS4930 students by 1pm.
- **April 23**
  - Poster session for COP5618 students during class. Attendance is mandatory for ALL students, both CIS 4930 and COP5618.
  - Electronic version of poster submission by 1pm.
- **April 25 at 5pm**
  - Project report due for COP5618 students. Submit ONE copy electronically AND turn in a hard copy. Hard copy of report contains annotated version of source code
  - Electronic submission of source code with instructions for building and running.
  - THIS IS A FIRM DEADLINE

Instructions

- This project may be done in **groups of 1-3 students**. The expectations are the same, regardless of the group size, so it is in your interest to find suitable partners and work together. If a group contains students in both CIS4930 and COP5618, the group will be subject to the requirements, deadlines, and expectations for COP5618.
- **Proposal:** Each project must be approved in advance. Submit a brief (1-2 page) proposal containing
  1. List of members of the group and their email addresses.
  2. Name of one designated member who will submit the project deliverables.
  3. An overview of what you plan to do.
  4. Brief discussion of how the project relates to material covered in the course
  5. Hardware and software that will be used.
- **Poster:** All groups will create a poster describing their project. This will be similar to the posters that researchers do for research projects at conference poster sessions. These will be displayed during the designated class period and evaluated by the instructor, TA, and your peers. If time does

not permit the instructor or TA to discuss each group, an individual session will be arranged later in the week at a mutually acceptable time.

- **Source code:** CIS4930 students must submit complete source code for the project along with a description of how to run it.
- **Written report:** COP5618 students must submit a written report in a format similar to any ACM journal format.
    - The report must include
        - An abstract
        - A description and evaluation of the project. This should include
            - Goal of the project
            - Design of software and/or experiment. Use code snippets as appropriate in the text.
            - Discussion of concurrency in the project
            - Evaluation, including a description of how it was tested.
        - Complete source code. *The hard copy must be annotated and cross referenced with the text so that the reader can easily find relevant pieces of code. (It is OK to do this with a marker.)* Make sure that the source code is formatted in a readable way. (Too many line wraps that destroy the formatting of the source make it unreadable)
        - A description of how the work was divided among group members. Some groups will want to cleanly partition the tasks among members; others may prefer to use pair programming, or some combination of the two approaches. Either is fine, just say what you did.
    - **Do not plagiarize other works.** See http://www.uflib.ufl.edu/msl/07b/studenthonorcode.html
    - **Use Standard English and include in-text citations to other works according to standard academic conventions.** For examples, see any ACM journal or conference proceedings, and look at the instructions at the bottom of the web pages for the latex and word templates. Do not include extraneous information; your goal is a **well-organized and succinct** report that clearly conveys what the project was about and what you learned from doing it.
    - For the electronic submission, submit the text of your report in an uncompressed pdf file. It will be analyzed by the Turnitin service for plagiarism. A separate e-Learning "assignment" will be provided for your source code.
- **Some groups will also require a demo**. This requirement depends on the nature of the project. In most , but not necessarily all cases, the requirement for a demo will be noted with the approval of the proposal. Normally demos will be done during the poster session. (If this is not possible due to the nature of the project, or a demo requirement is added after seeing the poster, a mutually agreeable time will be arranged during the week.)
- **Reports must be submitted electronically via e-learning AND a hard copy must be turned in**. Hard copies should be stapled in the upper left corner or along the left side. Do not use any kind of binder. There are staplers in the CISE office that can handle thick documents. (If you still need to hand in more than one packet that is OK.)

- **Projects closely related to a project done for another class**. Submitting substantially the same project for credit for two courses is a violation of the honor code. However, it may be acceptable to use a project done for another class as a starting point if there will be sufficient new concurrent programming aspects added. This must be addressed in your proposal; a discussion with the instructor beforehand is recommended.

Possible topics

The following list contains some general suggestions. You are not limited to things on the list—any interesting topic that uses or expands the material covered in the course is OK as long as the scope is suitable. However, projects on the following will NOT be approved: matrix multiplication, chat server, any kind of auction, sudoko.

- **Develop, document, test, and demonstrate a non-trivial concurrent/parallel program**. An iphone or android app that uses concurrency in a non-trivial way could be a good project. Particle swarm optimization.
- **Take an existing sequential program and add parallelism to improve its performance**. This could be done to take advantage of multiple cores or utilize a GPU. Determine whether or not the performance actually improved—if not, diagnose why. Your proposal should include a brief discussion of why you believe that this program is amenable to parallelization.
- **Evaluate a tool for testing or profiling concurrent programs**. Obtain the tool, learn how to use it, and try it on suitable examples. Your poster/report would briefly describe how the tool, show how you used the tool on your examples, and your opinion on the usefulness of the tool. Be clear about your criteria for "usefulness". Candidate tools include, but are not limited to
    - MultithreadedTC http://www.cs.umd.edu/projects/PL/multithreadedtc/index.html
        - This tool provides support for unit testing concurrent programs. Your homework assignments (possibly seeded with bugs) could provide some sample programs.
    - Java Pathfinder http://babelfish.arc.nasa.gov/trac/jpf/
        - There are several interesting JPF extensions as well described at http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/start
    - Tau

  If you choose this option, download the tool and make sure you can build it and get it to run on a toy problem before submitting your proposal.

- **Compare two environments for parallel or concurrent programming by implementing the same examples in each**. Possible languages of interest include (but are not limited to) OpenMP, MPI, X10, Titanium, Chapel, Cilk, Scala, Go, CUDA, OpenCL. To make this doable, one language should be one that you already know well. While publishable work is well beyond the scope of a class project, the following two papers that compare concurrent and parallel programming languages illustrate good practices.

1. Ian Karlin, Abhinav Bhatele, Jeff Keasler, Bradford L. Chamberlain, Jonathan Cohen, Zachary Devito, Riyaz Haque, Dan Laney, Edward Luke, Felix Wang, David Richards, Martin Schulz, Charles H. Still, "Exploring Traditional and Emerging Parallel Programming Models Using a Proxy Application," Parallel and Distributed Processing Symposium, International, pp. 919-932, 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013
2. Sebastian Nanz, Faraz Torshizi, Michela Pedroni, Bertrand Meyer, Design of an empirical study for comparing the usability of concurrent programming languages, Information and Software Technology, Volume 55, Issue 7, July 2013, Pages 1304-1315, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2012.08.013. (http://www.sciencedirect.com/science/article/pii/S0950584912001802)

- If you have access to a suitable environment, **implement and systematically test the performance of various lock implementations.**
- **Perform a careful study to determine the relative cost of volatiles, synchronized blocks, and atomic variables** on a particular system at different levels of contention.
- **Enhance aces4.** If you are interested in this, see Dr. Sanders

Hints

- Your overarching goal is to demonstrate through your project that you have learned something this semester and can apply what you learned. Keep this in mind as you select a project and prepare your poster and (if required) report.
- Repeating due to its importance: **Make sure that you do not plagiarize other works**. See http://www.uflib.ufl.edu/msl/07b/studenthonorcode.html
- For reports, the following templates for ACM journals might be useful. Any of the supplied formats are OK. If you scroll down, they give information about the appropriate format for references.
    - latex: http://www.acm.org/publications/submissions/latex_style
    - word: http://www.acm.org/publications/word_style/word-style-toc/

  It is not necessary to exactly follow these formats, and some of the items are clearly not relevant for a class project and should be omitted. (For example, you do not need Categories and Subject Descriptors, General Terms, or Additional Key Words and Phrases) Nevertheless, the overall structure should be emulated.

- You are responsible for obtaining access to whatever systems needed to perform this project. Make sure that your project is suitable for whatever system you use. For example, if you plan to use a dual core PC, then a project that attempts to determine how a spin lock implementation scales with large numbers of threads does not make sense. If you only have access to a large server that is shared by others, do not do a project where a main part of the project requires timing data; you cannot get good data on such a system. All projects should be “demoable” from campus.
- Your project should be well-engineered. Do not gratuitously include concepts from class that do not make sense in the context.

- If you use a swing GUI, make sure to initialize it in a thread-safe way. This was explicitly discussed as an example in class so there is no excuse for getting this wrong.
- Students in the past have not had much success with Hadoop-based projects. Showing speedup typically requires larger systems working on larger data sets than most students have access to.
- Use Wikipedia articles sparingly, if at all. You may find them useful as a starting point, but you need to cite the original references. A good Wikipedia article usually provides these.
- Posters can be printed inexpensively on campus. See http://print.at.ufl.edu/posters.shtml for templates and links to more info.