Memory Models: Project Description

Fall, 2018

Description

This document describes the motivation, goals, deliverables, and expectations from the CS8803 Memory Models class project.

One of the primary goals of CS8803 - Memory Models is to develop students' ability to do research in the area of Memory Models. The goal of this project is to provide students first-hand experience doing research in the area of memory models, as well as to evaluate their ability to conduct that research.

Memory models are by their nature a full software stack problem, with foundations in architecture, compilers, programming languages, and runtime systems. Students in this course also have a wide array of backgrounds an interests. Consequently, this is an open ended project, in which students are free to choose their own project, so long as they can convincingly argue that the project relates to and/or benefits the broadly defined area of memory models.

Expectations

The goal of this project is to conduct research in the area of memory models. Students will be graded based on their demonstration of their demonstrated aptitude for conducting novel and meaningful research. This includes: understanding related works in their field, describing and motivating the larger impacts of their research, and creating novel ideas and solutions to solve challenging problems.

Logistics

This project will have several deliverables, and will span over the remainder of the semester. Projects will be conducted in **groups of two** students. Each group will submit each of the following project deliverables:

Project Deliverables:

- Project proposal document
- Initial project presentation
- Bi-weekly (every other week) project updates
- Final project presentation
- Final written project report

In addition to the combined deliverables each student will **individually** submit an individual effort report, in which they indicate what portions of their project they were responsible for, and their approximate amount of effort in the project.

Individual Components

The project is split into six primary components. The remainder of this section describes those components, as well as what's expected from each.

Initial Proposal Document

The project proposal document should be a short (no more than 3 pages, 11pt font) document outlining the project students are proposing for this semester. The proposal should describe the project, its goals, and outline a plan for accomplishing those goals.

I understand that research is fluid. Some things will go better than you expect, but more will probably go worse, and you may not wind up doing what you intended to at the start of the project. The proposal is a best effort outline of your intentions for the project.

The proposal will be graded based on the following components:

- **Problem outline** – Describe the specific problem you're hoping to solve, and why others should care. Your work should have a strong motivation, both in a broad sense (why the area is important), as well as a specific sense (why your specific problem is important).
- **Contribution** – Describe your intended contribution. You've outlined a problem, but *how* do you intend to solve it?
- **Novelty** – What are the novel contributions of your work, how does it differ from its most related works.
- **Deliverables** – What are your goals over the rest of the semester, what do you ultimately hope to provide/report in your final report.
- **Timeline** – We will have bi-weekly (every other week) project reports in class. Create an outline of what you hope to have done at each project report point.

Proposal Presentation

Each group should prepare a 5-minute presentation (with slides) outlining their project. The presentation should briefly motivate their specific problem, discuss their proposed solution, its novelty, and outline what you hope to accomplish by the end of the semester. The goal of these discussions is to have a brief discussion of each proposal after the presentation, allowing students a chance to bounce their ideas off of the class, and adjust.

Bi-Weekly (every other week) Updates

Roughly every fourth class period (the 2nd class period every other week) we will have a project discussion class. The goal of these discussions is to allow students the opportunity to update their status, determine if things are going off track, and get feedback on the direction of their project. Students should have a roughly 2-3 minute (1 or 2 slide) presentation ready, to update their status since their last progress report.

Final Presentation

At the end of the semester students will present their work to the professor in a conference-style presentation. The presentation will be 15 minutes in length, and followed by a brief question and answer period. Like conference presentations, the students presentation is expected to outline the contributions and status of their project, with sufficient motivation. Like most conference presentations the project presentation's goal is not to discuss every detail of the work (that will be in the written report), but to give an outline of the significant and novel results of the work and the general methods used. The presentations will be graded based off the following criteria:

- The presentation is well motivated, and convincingly outlines the problem, and high-level goals of their solution.
- Students present good knowledge of their work and how it fits into the larger body of works in the area.
- The presentation their key insights clearly, and discusses their significance.
- Students are able to accurately and articulately answer any questions about their project, and clarify any misunderstandings.

Final Written Report

The final and most substantial deliverable portion of the project is its final written report. The report will be similar to a conference paper submission, constituting 9-10 pages of text, figures, tables, and diagrams, written in 11pt font. A Latex template will be provided, outlining the exact expected formatting of the paper. This document can be viewed as the culmination of the semesters worth of research and should discuss the progress made and lessons learned in the project in detail.

The document will be graded based on the following criteria:

- **Problem outline** – Describe the specific problem you're hoping to solve, and why others should care. Your work should have a strong motivation, both in a broad sense (why the area is important), as well as a specific sense (why your specific problem is important).
- **Contribution** – Describe your specific contribution. You've outlined a problem, but *how* do you intend to solve it?
- **Novelty/Related works** – What are the novel contributions of your work? How does your work fit into the universe of works in this area?
- **Implementation** – How did you construct your idea, what did you build? Were there any cool tricks, or non-obvious implementation details I should be aware of? Is there anything here you're really proud of?
- **Evaluation** – What have you done to evaluate the effectiveness of your ideas? How effective were your results.
- **Lessons Learned/Future Works** – Most research isn't perfect, or in its final completed form (especially for class projects). What is there still to do, or how can you build on your project? If your project didn't work out well, what unexpected thing or things happened, how would you change this going forward?

Much research doesn't work. I don't expect everyones project will function perfectly (or even well), that's why we have bi-weekly check-ins where we can adjust work when things go awry. Additionally, if your project doesn't work out, you wont inherently fail. Please note that in the previous bullets, relatively few of the graded portions are about results, but instead about your understanding of the problem, the research in the area, and the approach you tried to take (successfully or unsuccessfully) to solve the problem. You do have to put significant effort into this project (and if you don't it will be evident), but if you don't get the results you were looking for, it wont negatively impact your grade.

Deliverable Timeline

- **Oct 3** – The project proposal document will be due (midnight), and proposals will be presented and discussed in class.
- **Oct 24, Nov 7, Nov 19, Dec 3** – In-class project reports will be given.
- **Week of Dec 3** – Project final presentations will be given.
- **Dec 5** – Final written reports will be due. Individual contribution reports will be submitted at the same time.

Potential Project Topics

Below I've provided a list of potential topics you could use, adjust, or get inspiration from for your project. You are also strongly encouraged to brainstorm your own ideas (in my opinion idea brainstorming is the most fun part of the project).

1. **Semi-static fences** – "Racer: TSO" correctly points out that a major weakness in the SC-for-DRF memory models is that the static placement of barriers is often overly conservative, placing many barriers where rarely or even never race. Can we instead allow software to place "predicated" barriers, which can be transitioned into noops when certain conditions occur? This would allow powerful and complex static/dynamic runtimes to actively switch on and off barriers, without the complex racer-style hardware race detection – a best of both worlds.

2. **Understanding the behavior of shared memory (2 projects)** – Researchers have frequently pointed out that memory models are only significant for dynamic memory conflicts, time when concurrent accesses to memory are re-ordered. Many papers (e.g. the "SC Compiler" paper) gain great leverage by leveraging the freedom to move non-shared memory accesses. The question this project proposes is, how common are shared accesses? (and, have no doubt, this is a big question). This question can be viewed from two angles (for potentially two potential projects):

   - **How large are our dynamic sharing windows?** – Many researchers argue that SC enforcing hardware must use large amounts of speculation to have great performance. This high-degree of speculation increases complexity, and power consumption, making SC hardware impractical. My suspicion is that SC violating accesses are actually very rare, and a coarse-grained speculation could potentially handle them efficiently by trading of mis-speculation performance for less power consumption. Extension - Can you envision a hardware design that takes advantage of very infrequently shared data to provide SC at reasonable cost?
   - **What's the discrepency between static sharing and dynamic sharing?** – Statically proving two accesses are non-racy shows that they are not shared accesses, but static detectors are far from perfect, and give many false positives. Can you identify the number of sites that are dynamically accessed concurrently and compare that to their static counterparts? How does that number change if you start to include more information (either dynamic, or something like call context)?

3. **Optimistically Removing Fences** – When we insert software fences we do so conservatively, with the user/compiler inserting a fence where an instruction *may* conflict. Can we do better if we add in some dynamic assumptions? If I've never run block of code B, then I don't need a fence here. If thread C has never started, there is no need for a fence there. This project could tie into **semi-static fences**.

4. **From TSO to SC - what do we need?** – The most commonly used desktop platform today (x86) enforces TSO ordering. TSO is relatively strong, and has relatively few cases that require fences, when compared to more relaxed models. Can you show which cases TSO needs fences in order to operate in an SC manner, and potentially determine situations in which a TSO machine does not need to insert fences to create an SC execution?

If none of these appeal to you, you don’t have your own idea, and you’re still looking for one, feel free to contact me, I have several other directions, but these are my favorites.