Multicore Processors: Architecture & Programming

Projects

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

The project part of the course

- 50% of the total grade
- groups of 1 to 4 students
  - One submission per group
- Presentations in the last two lectures of the course
- You have 3 choices:
  - pick one of the suggested projects
  - <u>suggest</u> a modifications to a suggested project
  - <u>suggest</u> a different project

Milestones

Milestone 1: Mar 1st (By midnight)

Between Now and Mar 1st

- You can discuss with me any suggested ideas you may have about the project.

By Mar 1st

- Send me and email with your selections (one email per group not per student)
  - CC all group members
- **Email subject**: Project Selection
- **Body of the message**
  - Name of group members
  - Your selected project title.

Warning!!

- If you don't email me the final group/project by end of Mar 1st

  (-2%) of the total project grade for each late day.
- If you decide for a different project or a modified version of the published projects without discussing it with me first, the project topic may be rejected because it is too hard/simple, and you start getting late penalty.
  - So, don't wait till Mar 1st to discuss your suggested idea with me.

Milestone 2: Apr 19th

- **Final submission** → 40% of the total course grade
- No specific length
- Submit through **Brightspace**: one zip-file that contains:
  - report
  - All source code
  - A small text file, called **readme.txt**, telling us how to compile/execute your work (i.e. sort of quick user manual)
  - VERY IMPORTANT: All projects must work on our CIMS machines and does not use any software/library that does not work or is not present on CIMS. Telling us that it works on your laptop does not count.
- **Report Content:**
  - Abstract
  - Introduction
  - Literature Survey
  - Proposed Idea
  - Experimental Setup
  - Experiments & Analysis
  - Conclusions
  - References

Milestone 3: Apr 26th and May 3rd

- **Presentations** → 10% of the course grade
- 10 mins per group
  - 8 mins presentation
  - 2 mins Q&A
- You present only the main points and conclusions. Details are in the report.
- **7 slides only**: (Figures are better than list of bullets whenever possible)
  - Title Slide (including project title and name of all members)
  - 1 slide problem definition
  - 1 slide survey
  - 1 slide experimental setup
  - 2 slides results and analysis
  - 1 slide conclusions (up to three take aways from the project)

Milestone 3: Apr 26th and May 3rd

- The presentations will be graded based on:
  - How well you presented
  - Whether you followed the rules (timing, slides, etc)
  - How well you answer the question (s)

Regarding the Final Report

Regarding Final Report: Abstract

- The first thing in your report, but the last thing to write.
- 1 paragraph summarizing your problem and why it is important.
- 1-2 lines hinting on your technique
- 1-2 lines summarizing your finding(s)

Regarding Final Report: Introduction

- Expand the abstract to include why the problem you are solving is important.
- Then, give a general idea about your way of approaching the problem.

Regarding Final Report: Literature Review

- Feel free to organize this section in any way you want:
  - Include any subsections you think you need.
  - Draw any Figures you like.
- But you need to discuss *at least* the following items:
  - What did others do regarding the topic at hand? Note: the worst survey ever is when you say "x has done y; z has done k; …". You must put the work of others in taxonomy. That is, "The solution to this falls in x categories: …. . In the first category x has done …".
  - What are the pros and cons of what they are doing?
  - Why their work is not enough, and your proposed project is needed?
  - Important: Literature must not be about papers solving exactly your same problem. But can include papers about different problems but have similar challenges.

Regarding Final Report: Proposed Idea

- This is a major part of this report.
- Describe in great details the solution to the problem at hand.
  - For example, if you are parallelizing something, you must explain how you did that and justify your choices. If you are comparing things, you must specify what is the criteria of comparison is/are and why you picked them; and so on…

Regarding Final Report: Experimental Setup

- State:
  - The simulator you used (if any) or the specs of the machine you used
  - The benchmarks (if any),
  - etc …
- How do you know you have written a good experimental setup? If another researcher reads this section, the researcher must be able to replicate your experiments without asking you any questions.

Regarding Final Report: Results and Analysis

- **Formatting tip**: using tables, bullets, figures, diagrams, … are better than words in presenting results.
- What is/are the measures of success in your work? That is, what do you need to measure and what are you expecting the results to be in order to say that you succeeded?
- What are the experiments that you did in order to get the measurements you mentioned in the above bullet?
- ***Analysis:***
  - **Bad**: "*As we can see x is increasing with y*" ← You won't get any credits for that!
  - **Much better**: "x increases with y because 1, 2, 3, …. And under the conditions …. x may not increase with y. We learn from that …."

Regarding Final Report: Conclusions

- What are the most important findings of your project?
  - A bullets list is the best here.
- No more than three points.

Regarding Final Report: References

- If you only mention the references I provided you, it means you did not do any extra work.
- Websites like wikipedia, stackoverflow, etc are not references, don't use them.
- Don't use urls unless there is no paper about the topic.
- Example of reference format:
  - J. Lee and H. Kim. *Tap: A tlp-aware cache management policy for a cpu-gpu heterogeneous architecture.* In High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on, pages 1–12, 2012.

Suggested Projects

Check the page on the course website whose link is in the "Comments" column of Feb 22nd lecture.

Project 1: Parallel Computational Model

- Survey of parallel computational models currently available (e.g. PRAM, BSP, logP,…).
  - Do not limit to the three models we discussed in class.
- Show their strengths and weaknesses
- Propose your own (major part of the project)
- Compare all of them using a set of benchmark programs.

Project 2: Pick an application to parallelize (YOU must pick the application.)

- Must demonstrate lessons beyond "I implemented X on the multicore using this library/language/thread model, and it runs faster than a sequential version".
- Better compare with another parallel version in addition to the sequential version.
- Must clearly show the challenges facing the parallelization of this application and how you overcame it.

These lessons may include:

- Interesting generalized algorithmic or data structure contributions that can be applied to other applications (and these other applications should be made explicit)
- Interesting optimization techniques that can be applied beyond the application in the paper
- Interesting data management/representation techniques: how to best leverage the memory hierarchy, how to represent data in the most suitable way
- Rigorous comparisons with existing implementations.

Project 3:

Comparing several parallel programming languages

- Comparing programming languages is a valuable activity.
- A good comparison work helps understand the tradeoffs made in the design of a programming language.
- Decide on the different categories of benchmarks that were used and why?
- Criteria includes:
  - programmability
  - runtime overhead
  - performance
  - scalability
- A good comparison research must address 3 points.
  - You must make sure the quality of the benchmark implementations are uniform.
  - You must have an analytic performance target and **detailed profiling** so you can assess the quality of the language implementations.
  - You must assess the suitability of the algorithms relative to the design goals of the language.
- Example: **OpenMP vs Pthreads**

Project 4:

Cache replacement Policy for Shared Cache

- What block to kick when cache set is full?
- Must be cheap in terms of hardware usage.
- Using simulations and benchmark suites
- Pick one of two settings:
  - Multithreaded applications
  - Multiprogramming environment (i.e. more than one application running on the multicore simultaneously)
- Measures of success:
  - higher hit rate than competition
  - minimum effect on access latency

Project 5:
Bottleneck Analysis Parallel Programs

- Input: parallel programs
- Output:
  - Bottlenecks of performance
- Test your results using simulations (or execution on real machines) and benchmark suites

Project 6 – version a:
Performance Prediction of Multithreaded Applications

- Input:
  - Specs of a multithreaded program (What are these specs is part of the project.)
  - Specs of the machine that will be used to run program (What are they is part of the project.)
  - Problem size
  - Speedup on some machines
- Output: Expected speedup, on a different machine, relative to a one thread execution.

Project 6 - version b:
Performance Prediction of Multithreaded Applications

- Input:
  - Specs of a multithreaded program (What are these specs is part of the project.)
  - Specs of the machine that will be used to run program (What are they is part of the project.)
  - Problem size
- Output: Expected speedup relative to a one thread execution.

Project 7: Compare Coherence Protocols

- Exhaustive list of coherence protocols
- Implement them using simulator
- Discuss effect on parallel programs in terms of:
  - Scalability
  - Bandwidth requirement

Project 9:
Evaluation of Memory Allocators in OpenMP Programs

- Input: Several memory allocators implemented for multithreaded applications
- Output:
  - comparison of the different ones in terms of:
    - Speed
    - Scalability
    - False sharing avoidance
    - Low fragmentation
  - Implement your own

Suggested Resources

Simulator: Multi2sim

http://www.multi2sim.org/

- Simulates any number of cores and any cache hierarchy you specify.
- Written in C/C++
- Well documented
- Has many benchmarks already compiled for it.

Simulator: gem5

https://www.gem5.org/

- Almost the de facto simulator for computer systems
- Simulates any number of cores and any cache hierarchy you specify.
- Written in C++
- Well documented

Benchmarks
PARSEC

https://parsec.cs.princeton.edu/

- Good set of representative applications
- Most applications are implemented twice: once in OpenMP and once in PThreads

Have Fun!