
# 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 3 students
  - One submission per group
- Presentations in the last two lectures of the course
- You have 3 choices:
  - pick one of the suggested projects
  - suggest a modifications to a suggested project
  - suggest a different project

## Milestones: March 10th (Before class)

### Between Now and Mar 10th

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

### By March 10th

- Using NYU classes → messages (not forums)
- Send me your selections (one message per group)
- **Message title**: Project Selection
- **Body of the message**
  - Name of group members
  - Your selected project.

# Milestones: April 21st

- **Final report** → 35% of the grade
- No specific length
- Submit through **NYU classes**: one zip-file that contains:
  - report
  - All source code
  - A small text file telling use how to compile/execute your work (i.e. sort of quick user manual)

- **Content:**
  - Abstract
  - Introduction:
  - Literature Survey:
  - Proposed Idea:
  - Experimental Setup
  - Experiments & Analysis
  - Conclusions
  - References

# Milestones: April 28th and May 5th

- **Presentations** → 15% of the grade
- 15 mins per group
  - 10 mins presentation
  - 5 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

# 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 want
  - 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?

# 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 s/he 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!
  - Must better: "x increases with why 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 mention only 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:
  - 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

# Project 1:
# Parallel Computation Model

- Survey of parallel computational models currently available (e.g. PRAM, BSP, logP,…)
- show their strengths and weaknesses
- propose your own (major part of the project)
- compare all of them using a set of programs from benchmarks using a simulator or real machine.

# Project 2:
# Pick an application to parallelize

- 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.

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 set is full?
- Must be cheap in terms of hardware usage.
- Using simulations and benchmark suites
- Measures of success:
  - higher hit rate than competition
  - minimum effect on access latency

# Project 5:
## Bottleneck Analysis of Parallel Programs

- Input: parallel programs
- Output:
  - Bottlenecks of performance
  - Scalability predictions using an analytical model
- Test your results using simulations and benchmark suites

# Project 6:
## Phase Detection of Parallel Programs

- Input: parallel programs
- Output:
  - Online detector of phase change during execution.
  - Compare against other phase detection algorithms
- Test your results using simulations and benchmark suites

# Project 7:

## Performance Prediction of Multithreaded Applications

- Input:
  - Specs of a multithreaded program (What are they 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 8:
# 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:

## Build a visualization tool for OpenMP Programs

- Input: A parallel program written in OpenMP and compiled.
- Output: A visualization tool of the program execution, showing:
  - Number of  threads through time
  - Amount of memory used by the whole program through time
  - How many threads are active/blocked throughout the execution.

- The look of the final visualization is totally up to you to design. But you can learn from other visualization tools.

# 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.

# 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!