CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis
Please fill out the class statistics survey

https://forms.gle/7UD9gvXGpFzY6ZYr7
Why ML for Compilers?
Compilers translate high-level languages to low-level machine code

Program

for (i = 0; i < grid_points[0]; i++)
    for (j = 0; j < grid_points[1]; j++)
        for (k = 0; k < grid_points[2]; k++)
            for (m = 0; m < 5; m++)
                add = u[i][j][k][m] - u_exact[m];
                rms[m] = rms[m] + add*add;

Compiler

High-level programming language

Hardware

Low-level assembly language

Finding a semantic preserving (correct) translation that generates fast (optimized) code
Expectations of a compiler

• Produce correct code (correctness)
• Produce fast code (optimization)
• Work for multiple hardware platforms (retargetable)
• Easily maintainable

All of these are getting hard by the day
Back in the day....

Workload
Scientific Computing

Language
Fortran

Compiler
F77

Hardware
IBM 360
New languages were introduced

Workload

Language

Compiler

Hardware

IBM 360
Hardware evolved

Workload

Language

Compiler

Hardware

RISC processors

x86 processors (CISC)
Dennard Scaling (1974)

Dennard Scaling postulated that as transistors get smaller their power density stays constant, so that the power use stays in proportion with area. This allowed CPU manufacturers to raise clock frequencies from one generation to the next without significantly increasing overall circuit power consumption.

[NOTE: URL provided for context or additional reading]

Compiler Winter?

To get performance just stay until the next hardware generation

Workload

Language

Compiler

Hardware

x86 processors (CISC)

RISC processors
End of Dennard Scaling

New Hardware Innovations Needed

Figure 2. Sources of computing performance have been challenged by the end of Dennard scaling in 2004. All additional approaches to further performance improvements end in approximately 2025 due to the end of the roadmap for improvements to semiconductor lithography. Figure from Kunle Olukotun, Lance Hammond, Herb Sutter, Mark Horowitz and extended by John Shalf. (Online version in colour.)
Parallel hardware showed up

Workload

Language

Compiler

Hardware

12
Workloads diversified

Workload
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

Language
- C++
- Java
- F

Compiler

Hardware
- x86 processors (CISC)
- RISC processors
- GPUs
Domain Specific Languages

**Workload**
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

**Language**
- Microsoft FW
- C++
- Java
- Halide
- tvm
- TensorFlow

**Compiler**

**Hardware**
- x86 processors (CISC)
- RISC processors
- GPUs
Domain Specific Architectures

Workload
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

Language
- C++
- Java
- Halide
- tvm
- TensorFlow

Compiler

Hardware
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs
Tensor Processing Units

Figure 2. Floor Plan of TPU die. The shading follows Figure 1. The light (blue) data buffers are 37% of the die, the light (yellow) compute is 30%, the medium (green) I/O is 10%, and the dark (red) control is just 2%. Control is much larger (and much more difficult to design) in a CPU or GPU.
2017 Turing Award
New Golden Age for Computer Architecture

DSLs and DSAs

https://amturing.acm.org/vp/patterson_2316693.cfm

David Patterson

John Hennessy
Compiler Progression

**Workload**
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

**Language**
- C++
- Java
- Halide
- tvm
- TensorFlow

**Compiler**
- GCC
- LLVM
- MLIR

**Hardware**
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs

**DSLs**
- Halide
- tvm

**Scientific Applications**
**Image Processing**
**Deep Neural Networks**
**End-user Applications**

**Slow and Painful Progress!**
2020 Turing Award

For fundamental algorithms and theory underlying programming language implementation and for synthesizing these results and those of others in their highly influential books, which educated generations of computer scientists.

https://amturing.acm.org/byyear.cfm
Significant Manual Effort

• Plenty of Complex Analysis Passes
• Heuristic Optimization Algorithms
  • Loop transformations, vectorization, parallelization, peephole optimizations……
• Analytical Cost Models
  • Tunable Parameters
  • Simplified Machine Models

ACM Software Systems Award 2012

Thousands of contributors and millions of lines of code
(e.g., LLVM: 1,115 contributors and 2.5 million lines)
Meeting Expectations of a Compiler is not easy

Workload
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

Language
- C++
- Java
- Halide
- tf
- TensorFlow

Compiler
- GCC
- LLVM

Hardware
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs
Meeting Expectations of a Compiler is not easy

Workload
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

Language
- C++
- Java
- Halide
- TVM
- TensorFlow

Compiler
- GCC
- LLVM
- MLIR

Hardware
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs
Meeting Expectations of a Compiler is not easy

**Workload**
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

**Language**
- C++
- Deep Neural Networks
- TensorFlow

**Compiler**
- LLVM

**Hardware**
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs
Meeting Expectations of a Compiler is not easy

**Workload**
- Scientific Applications
- End-user Applications
- Image Processing
- Deep Neural Networks

**Language**
- C++
- Java
- Halide
- tvm
- TensorFlow

**Compiler**
- GCC
- LLVM
- Halide
- tvm

**Hardware**
- x86 processors (CISC)
- RISC processors
- GPUs
- TPUs
- GraphCore
- DSPs
- DSAs
Meeting Expectations of a Compiler is not easy

Too many combinations of workloads, languages and hardware!!!!
Significant manual effort

- Plenty of Complex Analysis Passes
- Heuristic Optimization Algorithms
  - Loop transformations, vectorization, parallelization, peephole optimizations…….
- Analytical Cost Models
  - Tunable Parameters
  - Simplified Machine Models

- Are tedious to develop and maintain
- Can easily become stale
- Not adaptive
Let’s **automate** decision making

- Auto-tuning - automatically finding the best optimization strategy
  - Techniques, algorithms - mostly search
  - Frameworks
  - Input sensitivity
  - Heterogeneity
- Learned Optimizations - Machine Learning
  - Generalizable policies
  - Hybrid Learning + Search
- Data-driven Cost Models
- New Program Representations (Program Embeddings)

- State-of-the-art results
- Easier to develop and maintain
- Responsive and adaptive
2021 Turing Award

For his pioneering contributions to numerical algorithms and libraries that enabled high performance computational software to keep pace with exponential hardware improvements for over four decades.

https://amturing.acm.org/byyear.cfm

Jack Dongarra
Designing new hardware with ML

- Design Space Exploration
- Techniques
- Finding better and newer hardware configurations
- Data-driven simulations
- Improved Electronic Design Automation
- Joint placement and routing

Article | Published: 09 June 2021

A graph placement methodology for fast chip design

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter & Jeff Dean
Logistics
Class Structure

• **Time:** Every Tuesday and Thursday: 9.30am - 10.45am

• **Place:** 1043 Sidney Lu Mechanical Engineering Building
  • After the first few lectures class participation is required for paper discussions

• **Instructor:** Charith Mendis  **TA:** Stefanos Baziotis
  • First few weeks: Introductory lectures
  • Rest of the course: Paper reading, reviewing, presentations and project
  • Guest Lecture (may be a departmental colloquium)

• **Website:** https://mlcomp.cs.illinois.edu/fa2023

• **Campuswire:** Please join the class campuswire. Link is on the website.
COVID-19

• Please adhere to guidance given at https://covid19.illinois.edu/on-campuse/on-campus/on-campus-students/

• In-person lectures and discussions; not planning on hybrid lectures
Learning Outcomes

• After completing this course you should be able to
  • Articulate latest research in this area
  • Be prepared to perform original research in this area
  • Critique and evaluate scholarly work in this area
  • Communicate your own research findings with the community

This is a Research Seminar Course

• Please be interactive; the more the better
• Ask questions
• Give constructive feedback to presentations
Grading (Tentative)

Three (or four) components

• Paper reviews 20%
• Paper presentation and discussion lead 40%
• Project 40%
  • May be 1 structured (mini) project
  • May be 1 open-ended project
Paper Reviews

• Each class will have a required reading starting September 12th

• Write a review between 250 - 750 words on
  • Summary and contributions of the paper
  • Strengths and weaknesses
  • How to improve the paper (be vague and adventurous)

• Due on Sunday (Tuesday class) and on Tuesday (Thursday class) midnight

• We will use hotCRP to enter reviews (https://uiuc-cs598mlcomp2023.hotcrp.com)

• 20 main readings chosen by the instructor. If you want a paper to be included, please explain yourself
Paper Presentation

- Choose at least 5 papers that you are willing to present by August 31st

- Submission link is available in the class website

- **Week before**: Meet instructor to discuss the presentation plan (compulsory!)
  - Use this time to ask questions and discuss the outline
  - Presentation slides are due when reviews are due for that class
  - Submission details are in the website

- **During the class**: Be present in class (compulsory!)
  - Deliver a 30 min presentation on the paper
  - Answer questions for the following 20 min
  - Final 25 min for open discussion on the paper (lead by the instructor)
Paper Presentation

• After class: Summarize the discussion of the paper
  • Submit the summary by the start of the next class

• The presentation should include
  • Problem definition
  • Motivation: Why is this an important problem?
  • Outline the high-level solution
  • Illustrate the solution
  • Evaluation: What worked and what didn’t
  • Related Work: Put the solution in context of other research
  • Strengths and weaknesses
  • How would you extend this work?
We have open sourced the XLA TPU compiler timing dataset [1].

Earlier, in our work we proposed a learned cost model with this dataset. This is a reading in this class.

Challenge: Can you beat our technique?

There is a Kaggle competition setup with total prize of $50,000 partly sponsored by Google. Students are encouraged to submit their cost model designs to this competition (when available).

Deliverables:
- Model file with the correct interface
- Grading will be automated; yet to decide whether it would be competitive grading.

[2] A Learned Performance Model for Tensor Processing Units, Kaufmann et. al. MLSys 2021
(Open-ended) Project

• Complete a project by the end of the semester in groups of 2

• Project Proposal
  • 1 page writeup

• Schedule a 10-min meeting with the instructor to discuss the proposal
  • Watch out for a signup sheet
  • Use the feedback to adjust project expectations and directions

• Deliverables:
  • 5 page write up in regular 2-column conference format
  • 7 min presentation per group
(Open-ended) Project

- Details to follow in the next few weeks

- Tentatively we plan on having 3 kinds of projects (subject to change)
  - Surveys of at least 15 papers on a topic
  - Reproducing results of at least 3-4 related research papers and comparisons
  - Research project on a novel direction (ok not to get desired results) encouraged!
Resources

• How to read a research paper: https://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf

• Constructive and Positive Reviewing: https://www.cs.utexas.edu/users/mckinley/notes/reviewing.html

• How to speak by Patrick Winston: https://www.youtube.com/watch?v=Unzc731iCUY
Any Questions?