CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis
Please fill out the class statistics survey

https://forms.gle/jAnohznDSEBvyPgr8
Why ML for Compilers?
Compilers translate high-level languages to low-level machine code

Program:

```c
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:

Low-level assembly language:

```assembly
addq %rcx, %rax
movq %rax, %rcx
salq $6, %rcx
addq %rcx, %rax
imulq $21125, %rdi, %rcx
addq %rax, %rcx
movq %rdx, %rax
salq $2, %rax
addq %rsi, %rax
```

High-level programming language

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: F, C++, Java
- Compiler
- Hardware: IBM 360
Hardware evolved

Workload

Language

Compiler

Hardware

x86 processors (CISC)

RISC processors
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.


Compiler Winter?

Workload

Language

Compiler

Hardware

x86 processors (CISC)

RISC processors

To get performance just stay until the next hardware generation
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

- x86 processors (CISC)
- RISC processors
- GPUs

Multi-Cores
Multi-Processor
GPUs
Workloads diversified

Workload
- Scientific Applications
- End-user Applications
- Graphs
- Deep Neural Networks

Language
- C++
- F
- Java

Compiler

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

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

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

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
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/bbyyear.cfm

Alfred Aho

Jeffrey Ullman
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

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
Meeting Expectations of a Compiler is not easy

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

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

**Compiler**
- GCC
- LLVM
- MLIR

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

<table>
<thead>
<tr>
<th>Workload</th>
<th>Language</th>
<th>Compiler</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific</td>
<td>C++</td>
<td>LLVM</td>
<td>x86 processors (CISC)</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
<td></td>
<td>RISC processors</td>
</tr>
<tr>
<td>End-user</td>
<td>Java</td>
<td></td>
<td>GPUs</td>
</tr>
<tr>
<td>Applications</td>
<td>Halide</td>
<td></td>
<td>TPUs</td>
</tr>
<tr>
<td>Image Processing</td>
<td>tvm</td>
<td></td>
<td>GraphCore</td>
</tr>
<tr>
<td>Graphs</td>
<td>TensorFlow</td>
<td></td>
<td>DSPs</td>
</tr>
<tr>
<td>Deep Neural</td>
<td></td>
<td></td>
<td>DSAs</td>
</tr>
<tr>
<td>Networks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The diagram illustrates the process of compiler development across various types of applications, languages, and hardware. Scientific applications, end-user applications, and image processing are examples of workloads that a compiler may need to optimize for. Languages such as C++, Java, Halide, TensorFlow, and DSLs can be compiled using tools like GCC and LLVM. Hardware platforms including x86 processors, RISC processors, GPUs, TPUs, GraphCore, and DSPs create a heterogenous system that compilers must be able to handle effectively.
Meeting Expectations of a Compiler is not easy

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

Language
- Microsoft
- C++
- Java
- Halide
- TensorFlow
- DSLs

Compiler
- GCC
- LLVM
- Halide
- tvm
- MLIR

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

<table>
<thead>
<tr>
<th>Workload</th>
<th>Language</th>
<th>Compiler</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific Applications</td>
<td>Deep Neural Networks</td>
<td>LLVM</td>
<td>x86 processors (CISC)</td>
</tr>
<tr>
<td>Image Processing</td>
<td></td>
<td></td>
<td>RISC processors</td>
</tr>
<tr>
<td>Graphs</td>
<td></td>
<td></td>
<td>GPUs</td>
</tr>
<tr>
<td>Deep Neural Networks</td>
<td></td>
<td></td>
<td>TPUs</td>
</tr>
<tr>
<td>DSPs</td>
<td></td>
<td></td>
<td>GraphCore</td>
</tr>
<tr>
<td>DSAs</td>
<td></td>
<td></td>
<td>DSPs</td>
</tr>
</tbody>
</table>

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
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

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:** **1105** Siebel Center
  - After the first few lectures class participation is required for paper discussions

• **Instructor:** Charith Mendis
  - First few weeks: Introductory lectures
  - Rest of the course: Paper reading, reviewing, presentations and project
  - Guest Lecture from Google (tentative)

• **Website:** https://charithm.web.illinois.edu/cs598cm/fa2022/

• **Piazza:** Please join the class piazza. Link is on the website.
COVID-19

- Please adhere to guidance given at https://covid19.illinois.edu/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

Four components

- Mini- quizzes 10%
- Paper reviews 15%
- Paper presentation and discussion lead 25%
- Project 50%
Mini-Quizzes

• A simple question about that day’s assigned main reading paper
• This can be a MCQ, short answer or an opinion based question
• Questions will be administered through Piazza. Please sign up!
• Each question is worth 0.5% (total 10% for 20 papers)
Paper Reviews

• Each class will have a required reading starting September 13th

• 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://cs598cm-mlcomp-fall2022.hotcrp.com)

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

• Choose at least 5 papers that you are willing to present by September 1st

• 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?
Project

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

- Project Proposal: Due October 6th midnight
  - 1 page writeup

- Schedule a 10-min meeting with the instructor to discuss the proposal during the week of October 10th-15th.
  - 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 due on December 6th
  - 7 min presentation per group on December 6th (last day of classes)
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?