CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis
• **Reading List:** Live on the website!

• **Paper Selections:** Due on August 31st

• **Paper Reviews:** We will use hotCRP to facilitate review writing. We will change the fields of the reviews.

• **Resources and tutorials:** Towards the bottom of the website
Recap

• Compiler Stages
  • Lexer => Parser => Sema => Optimization => Code Generation

• Two types of compiler optimizations

• Phase ordering problem
Lecture 3:
Compiler Optimizations

Optimizations + DSLs
Anatomy of an Optimization Pass

Objective (f)

Input code (I) → Step 1 → Step 2 → … → Step n → Output code (O)
Anatomy of an Optimization Pass

Objective (f)

Input code (I) → Decide what and how to Optimize → Transform Code → Output code (O)
Anatomy of an Optimization Pass

Input code (I) → Decide what and how to Optimize → Transform Code → Output code (O)

- Objective (f)
- Optimization Decision Making
- Transformation Machinery
Robot Analogy

Task: Move from A to B cheaply
Robot Analogy

Task: Move from A to B cheaply

1. Plan
Robot Analogy

Task: Move from A to B cheaply

1. Plan
2. Execute
Robot Analogy

Task: Move from A to B cheaply

1. Plan
2. Execute
Anatomy of an Optimization Pass

Input code (I) → Decide what and how to Optimize → Transform Code → Output code (O)

1. Optimization Decision Making
2. Transformation Machinery
Optimization Decision Making

Faster and Correct Output IR
Optimization Decision Making

Faster and Correct Output IR

Transformation Space
Optimization Decision Making

semantically equivalent transformations

Transformation Space

Faster and Correct Output IR

Input IR \rightarrow \text{Opt} \rightarrow \text{Output IR}
Optimization Decision Making

semantically equivalent transformations

Subspace

Faster and Correct Output IR

Input IR → Opt → Output IR

Transformation Space
Optimization Decision Making

semantically equivalent transformations

Faster and Correct Output IR

Transformation Space

Cost Model

Charith Mendis
04/02/2020

Input IR → Opt → Output IR
Optimization Decision Making

- Subspace
- semantically equivalent transformations

Faster and Correct Output IR

Input IR → Opt → Output IR

Transformation Space → Optimization Strategy → Cost Model
Optimization Decision Making

- Semantically equivalent transformations
- Subspace
- Faster and Correct Output IR

OPTIMIZATION

- Transformation Space
- Optimization Strategy
- Cost Model
Optimization Decision Making

semantically equivalent transformations

Faster and Correct Output IR

Transformation Space  Optimization Strategy  Cost Model
Optimization Decision Making

semantically equivalent transformations

Faster and Correct Output IR

Input IR → Opt → Output IR

Transformation Space → Optimization Strategy

Cost Model

Ideal

Legal Transformations

Optimal

Ground Truth Runtime
Robot Analogy

Task: Move from A to B cheaply

1. Plan
2. Execute

Transformation Space
Optimization Strategy
Cost Model
Robot Analogy

Task: Move from A to B cheaply

1. Plan

2. Execute

Transformation Space Optimization Strategy Cost Model
Robot Analogy

Task: Move from A to B cheaply

1. Plan
2. Execute

Cost: 7
Robot Analogy

Task: Move from A to B cheaply

1. Plan
2. Execute

Cost: 5
Independent and Isomorphic statements can be vectorized

Scalar Code

\[
\begin{align*}
  a[0] &= b[0] + c[0] \\
\end{align*}
\]

Vector Code

Single Instruction Multiple Data (SIMD)

\[
\{a[0], a[1]\} = \{b[0], b[1]\} + \{c[0], c[1]\}
\]

Larsen & Amarasinghe “Exploiting Superword Level Parallelism with Multimedia Instruction Sets” [PLDI'00]
• Find **independent** and **isomorphic** statements

• Not all vector packs can exist with each other

• Need to select the most profitable packing strategy

\[ \begin{align*}
S4 : A4 &= L[1] - A2 \\
\end{align*} \]

\{S1,S2\} \quad \{S4,S5\} \\
\{S2,S3\} \quad \{S5,S6\} \\
\{S1,S3\} \quad \{S4,S6\}

**Transformation Space**
Statement packing strategy 1

Scalar code

S4 : A4 = L[1] - A2

Vector code

S4 : A4 = L[1] - A2

Instruction Breakdown
0 vector
0 packing
0 unpacking
There are costs associated with vectorization

Scalar code

S4 : A4 = L[1] - A2

Vector code

SV1 : \{A1, A2\} = \{L[5], L[6]\} / \{L[2], L[3]\}


SV2 : \{A5, A6\} = \{L[2], L[3]\} - \{A3, A1\}

Non-isomorphic

4 vector
0 packing
0 unpacking
There are costs associated with vectorization

Scalar code

- S4: A4 = L[1] - A2

Vector code

- SV1: \{A1,A2\} = \{L[5],L[6]\} / \{L[2],L[3]\}
- SU1: A1 = unpack(SV1,1)
- S4: A4 = L[1] - A2
- SV2: \{A5,A6\} = \{L[2],L[3]\} - \{A3,A1\}

Instruction Breakdown

- 4 vector
- 0 packing
- 1 unpacking
There are costs associated with vectorization

Scalar code

S4 : A4 = L[1] - A2

Vector code

SV1 : {A1,A2} = {L[5],L[6]} / {L[2],L[3]}
SU1 : A1 = unpack(SV1,1)

SP1 : {A3,A1} = pack(A3,A1)
S4 : A4 = L[1] - A2
SV2 : {A5,A6} = {L[2],L[3]} - {A3,A1}

Instruction Breakdown

4 vector
1 packing
1 unpacking
There are costs associated with vectorization

Scalar code

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S4</td>
<td>A4 = L[1] - A2</td>
</tr>
</tbody>
</table>

Vector code

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SV1</td>
<td>{A1,A2} = {L[5],L[6]} / {L[2],L[3]}</td>
</tr>
<tr>
<td>SU1</td>
<td>A1 = unpack(SV1,1)</td>
</tr>
<tr>
<td>SU2</td>
<td>A2 = unpack(SV1,2)</td>
</tr>
<tr>
<td>SP1</td>
<td>{A3,A1} = pack(A3,A1)</td>
</tr>
<tr>
<td>S4</td>
<td>A4 = L[1] - A2</td>
</tr>
<tr>
<td>SV2</td>
<td>{A5,A6} = {L[2],L[3]} - {A3,A1}</td>
</tr>
</tbody>
</table>

Instruction Breakdown

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4 vector</td>
<td></td>
</tr>
<tr>
<td>1 packing</td>
<td></td>
</tr>
<tr>
<td>2 unpacking</td>
<td></td>
</tr>
</tbody>
</table>
Statement packing strategy 2

Scalar code

<table>
<thead>
<tr>
<th>Statement</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>S4:</td>
<td>A4 = L[1] - A2</td>
</tr>
</tbody>
</table>

Vector code

<table>
<thead>
<tr>
<th>Statement</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV1:</td>
<td>{A2, A3} = {L[6], L[7]} / {L[3], L[4]}</td>
</tr>
<tr>
<td>SU1:</td>
<td>L[2] = unpack(SLV1, 2)</td>
</tr>
<tr>
<td>SU2:</td>
<td>L[3] = unpack(SLV2, 1)</td>
</tr>
<tr>
<td>SV2:</td>
<td>{A4, A5} = {L[1], L[2]} - {A2, A3}</td>
</tr>
</tbody>
</table>

Instruction Breakdown

- Vector: 5
- Packing: 0
- Unpacking: 2
Different vectorization schemes have different profitability

**Strategy 1**

Liu et. al [PLDI’12]

- **SV1** : \{A1,A2\} = \{L[5],L[6]\} / \{L[2],L[3]\}
- **SU1** : A1 = unpack(SV1,1)
- **SU2** : A2 = unpack(SV1,2)
- **SP1** : \{A3,A1\} = pack(A3,A1)
- **S4** : A4 = L[1] - A2
- **SV2** : \{A5,A6\} = \{L[2],L[3]\} - \{A3,A1\}

4 vector
1 packing
2 unpacking

**Strategy 2**

- **SV1** : \{A2,A3\} = \{L[6],L[7]\} / \{L[3],L[4]\}
- **SU1** : L[2] = unpack(SL1V1,2)
- **SU2** : L[3] = unpack(SLV2,1)
- **SV2** : \{A4,A5\} = \{L[1],L[2]\} - \{A2,A3\}

5 vector
0 packing
2 unpacking
# Machine Learning Influence

## Transformation Space
- **Manual**

## Optimization Strategy
- **Greedy / Heuristic**
- **Integer Linear Programming**
- **Dynamic Programming**

## Cost Model
- **Analytical**
  - Linear
  - Non-linear

### Traditional solutions
- **Hand-written**

### Automated solutions
- **Program Logics**
  - **Data-driven**
    - Imitation Learning
    - LSTM based Cost Model
  - **10/19**
  - **10/05**
  - (related reading)
Domain Specific Languages

- Programming model specific to one domain
  - Image / Array Processing - Halide, MATLAB
  - Sparse Tensor Computations - TACO
  - Tensor Algebra - Tensorflow, Pytorch (frameworks)
  - Graphs - GraphIt, Gunrock
  - Genomic Computations - Seq

- Usually comes with a set of domain specific optimizations
Halide

• **Main idea:** Separate algorithm specification from optimizations (schedules)

• Halide Video

  • [https://www.youtube.com/watch?v=3uiEyEKji0M&t=3s](https://www.youtube.com/watch?v=3uiEyEKji0M&t=3s)

• **Optimization objective:** find the best schedule or optimization sequence for a given Halide algorithm
Tensorflow

• Model tensor manipulating programs

• Uses the XLA compiler to target GPUs, TPUs and CPUs

• Main abstraction: Computational Graphs

IR: High Level Operations (HLO)
• (Most) optimizations can be expressed as computational graph rewrites

(a) Associativity of matrix multiplication.

(b) Fusing two matrix multiplications using concatenation and split.

TASO [SOSP’19]

Machine Learning Influence

- **Transformation Space**
  - Automated solutions
  - Program Logics
  - 10/26: Tree Search (Halide)
  - 10/31: Gradient-based Methods (TVM)

- **Optimization Strategy**
  - Data-driven
  - 10/10: GNN based Cost Model (XLA)

- **Cost Model**
  - Data-driven
XLA Compiler

- (Most) optimizations can be expressed as computational graph rewrites

- What if there are 2 or more rewrites that can be performed at the same time?

(a) Associativity of matrix multiplication.

(b) Fusing two matrix multiplications using concatenation and split.
Halide

- **Optimization objective:** find the best schedule or optimization sequence for a given Halide algorithm
Paper Presentation

- Paper presentations assigned on September 4th

- 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
  - Submit using the hotCRP system

- 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

- First presentation on **September 12th**
  - Whaley and Dongarra, “Automatically Tuned Linear Algebra Software” (SC 1998)
  - 30 min presentation
  - [https://amturing.acm.org/award_winners/dongarra_3406337.cfm](https://amturing.acm.org/award_winners/dongarra_3406337.cfm)
Any Questions?