Brief Announcements

• Recordings and Zoom

• **Reading List:** Live on the website!

• **Paper Selections:** Due on **September 7th**

• **Paper Reviews:** We will use hotCRP to facilitate review writing. Instructions to come soon!

• **Resources and tutorials:** Will be up by Friday
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

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

Objective \((f)\)

- Input code \((I)\)
  - Decide what and how to Optimize
  - Transform Code
- Optimization Decision Making
- Transformation Machinery

Output code \((O)\)
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

1. Optimization Decision Making
2. Transformation Machinery

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

Faster and Correct Output IR
Optimization Decision Making

semantically equivalent transformations

Faster and Correct Output IR

Transformation Space

Opt

Input IR → Opt → Output IR
Optimization Decision Making

Semantically equivalent transformations

Faster and Correct Output IR

Transformation Space
Optimization Decision Making

- Transformation Space
- Cost Model
- Faster and Correct Output IR

Input IR -> Opt -> Output IR

Semantically equivalent transformations

Charith Mendis
04/02/2020
Optimization Decision Making

Faster and Correct Output IR

Input IR → Opt → Output IR

Semantically equivalent transformations

Subspace

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 $\rightarrow$ Opt $\rightarrow$ Output IR

Transformation Space $\xrightarrow{\text{Optimization Strategy}}$ Cost Model
Optimization Decision Making

- Subspace
- semantically equivalent transformations

Faster and Correct Output IR

Optimization Decision Making

Transformation Space

Optimization Strategy

Cost Model

Legal Transformations

Ideal

Optimal

Ground Truth Runtime

Ideal
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
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)

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

Vector Packs

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

\[
\begin{align*}
S1 &: A1 = \frac{L[5]}{L[2]} \\
S2 &: A2 = \frac{L[6]}{L[3]} \\
S3 &: A3 = \frac{L[7]}{L[4]} \\
S4 &: A4 = L[1] - A2 \\
S5 &: A5 = L[2] - A3 \\
\end{align*}
\]

\[
\begin{align*}
\{S1,S2\} & \quad \{S4,S5\} \\
\{S2,S3\} & \quad \{S5,S6\} \\
\{S1,S3\} & \quad \{S4,S6\}
\end{align*}
\]

**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\}

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

Non-isomorphic

Instruction Breakdown

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**

- S1: $A_1 = L[5] \div L[2]$
- S4: $A_4 = L[1] - A_2$
- S6: $A_6 = L[3] - A_1$

**Vector code**

- SV1: $\{A_1, A_2\} = \{L[5], L[6]\} / \{L[2], L[3]\}$
- SU1: $A_1 = \text{unpack}(SV1, 1)$
- SU2: $A_2 = \text{unpack}(SV1, 2)$
- SP1: $\{A_3, A_1\} = \text{pack}(A_3, A_1)$
- S4: $A_4 = L[1] - A_2$
- SV2: $\{A_5, A_6\} = \{L[2], L[3]\} - \{A_3, A_1\}$

**Instruction Breakdown**

- 4 vector
- 1 packing
- 2 unpacking
Statement packing strategy 2

Scalar code

| S4  | A4 = L[1] - A2   |

Vector code

| SV1 | {A2,A3} = {L[6],L[7]} / {L[3],L[4]} |
| SU1 | L[2] = unpack(SLV1,2) |
| SU2 | L[3] = unpack(SLV2,1) |
| SV2 | {A4,A5} = {L[1],L[2]} - {A2,A3} |

Instruction Breakdown

- 5 vector
- 0 packing
- 2 unpacking
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

Optimal

SV1 : \{A2,A3\} = \{L[6],L[7]\} / \{L[3],L[4]\}
SU1 : L[2] = unpack(SLV1,2)
SU2 : L[3] = unpack(SLV2,1)
SV2 : \{A4,A5\} = \{L[1],L[2]\} - \{A2,A3\}

5 vector
0 packing
2 unpacking
Machine Learning Influence

<table>
<thead>
<tr>
<th>Transformation Space</th>
<th>Optimization Strategy</th>
<th>Cost Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traditional solutions</strong></td>
<td>Hand-written</td>
<td>• Greedy / Heuristic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Integer Linear Programming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dynamic Programming</td>
</tr>
<tr>
<td><strong>Automated solutions</strong></td>
<td>Program Logics</td>
<td>• Analytical Linear Non-linear</td>
</tr>
</tbody>
</table>

- **Data-driven**
  - Imitation Learning 10/21
  - LSTM based Cost Model 10/07
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)
XLA Compiler

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

![Diagram of computational graph rewrites](https://cs.stanford.edu/~padon/taso-sosp19.pdf)

TASO [SOSP’19]

Machine Learning Influence

Transformation Space | Optimization Strategy | Cost Model

Automated solutions | Program Logics | Data-driven | Data-driven

10/28: Tree Search (Halide) | 10/12: GNN based Cost Model (XLA)
11/02: Reinforcement Learning (Halide)
Paper Presentation

• Paper presentations assigned on September 8th

• **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 20-30 min presentation on the paper
  • Answer questions for the following 15 min
  • Final 30-40 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 (hotCRP)

• First presentation on **September 14th**
  • Matteo Frigo, “A Fast Fourier transform Compiler” (PLDI 1999)
  • 15 min presentation
  • Meet instructor on Thursday 9th to discuss the outline
Any Questions?