Learning to Optimize Tensor Programs

Frameworks

High-level data flow graph and optimizations

Hardware
Learning to Optimize Tensor Programs

Frameworks

High-level data flow graph and optimizations

Hardware
Learning to Optimize Tensor Programs

Frameworks

High-level data flow graph and optimizations

Machine Learning based Program Optimizer

Hardware
Learning to Optimize Tensor Programs

Frameworks

High-level data flow graph and optimizations

Machine Learning based Program Optimizer

Learning to generate optimized program for new operator workloads and hardware

Hardware
Search over Possible Program Transformations

Compute Description

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]
Search over Possible Program Transformations

Compute Description

```python
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```
Search over Possible Program Transformations

Compute Description

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

Billions of possible optimization choices

Loop Transformations
Thread Bindings
Cache Locality
Thread Cooperation
Tensorization
Latency Hiding

Hardware
Learning-based Program Optimizer
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program

Runtime Measurements
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program

Runtime Measurements

High experiment cost, each trial costs ~1 second
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program

Cost Model
Learning-based Program Optimizer

Need reliable cost model per hardware
Learning-based Program Optimizer
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program

Training data $\mathcal{D}$
Learning-based Program Optimizer

Program Optimizer → Code Generator → Program

Learning

Statistical Cost Model

Training data
Learning-based Program Optimizer

- Relatively low experiment cost
- Domain-specific problem structure
- Large quantity of similar tasks

Unique Problem Characteristics

1. Program Optimizer
2. Code Generator
3. Statistical Cost Model
4. Training data

Diagram showing the flow of data and components involved in the learning-based program optimizer.
Program-aware Cost Modeling

High-Level Configuration
Program-aware Cost Modeling

High-Level Configuration

```
for y in range(8):
    for x in range(8):
        for k in range(8):
            C[y][x] += A[k][y]*B[k][x]
```

Low-level Abstract Syntax Tree
(shared between tasks)
Program-aware Cost Modeling

High-Level Configuration

```python
for y in range(8):
    for x in range(8):
        C[y][x]=0
    for k in range(8):
        C[y][x]+=A[k][y]*B[k][x]
```
Program-aware Cost Modeling

High-Level Configuration

```
for y in range(8):
    for x in range(8):
        C[y][x]=0
    for k in range(8):
        C[y][x]+=A[k][y]*B[k][x]
```

Low-level Abstract Syntax Tree (shared between tasks)

Boosted Tree Ensembles

```
for context vec of x
for context vec of y
for context vec of k
```

Soft scatter

```
final embedding
```

TreeGRU
Effectiveness of ML based Model
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X

Baseline: CuDNN
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X

Baseline: CuDNN

TVM: Random Search
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
Transfer Learning Among Different Workloads

Historical Optimization Tasks
Transfer Learning Among Different Workloads

Historical Optimization Tasks

Domain Invariant Program Representations
Transfer Learning Among Different Workloads

- Historical Optimization Tasks
- Domain Invariant Program Representations
- Transferable Models to speedup new tasks
Transfer Learning Among Different Workloads

Historical Optimization Tasks

Domain Invariant Program Representations

Transferable Models to speedup new tasks

- GBT on Configuration S
- GBT on Flatten Loop Context x
- GBT on Context Relation R
- GBT No Transfer

TITANX C7 in domain

Number of Trials:

0.0 0.5 1.0

Number of Trials:

0 250 500 750

TITANX C1–C6 -> C7

TITANX C1–C6 -> Matmul-1024

Mali GPU C1–C6 -> A53 CPU C7

Number of Trials:

0 2

Number of Trials:

0 0.005

Number of Trials:

0 50 100 150

Number of Trials:

0 50 100 150
NVIDIA GPU Optimization (GTX 1080 Ti)

Latency (ms)

<table>
<thead>
<tr>
<th>Model</th>
<th>MXNet + TensorRT 4.0</th>
<th>AutoTVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>1.75</td>
<td>3.5</td>
</tr>
<tr>
<td>MobileNet</td>
<td>3.5</td>
<td>5.25</td>
</tr>
<tr>
<td>VGG-19</td>
<td>5.25</td>
<td>7.0</td>
</tr>
<tr>
<td>Inception V3</td>
<td>7.0</td>
<td>5.25</td>
</tr>
<tr>
<td>DenseNet-121</td>
<td>7.0</td>
<td>7.0</td>
</tr>
</tbody>
</table>
Bonus (INT8, GTX 1080)

Latency (ms)

- cuDNN
- AutoTVM

Graph showing latency for different configurations:
- 1-7-512-512-1
- 4-7-512-512-1
- 1-7-512-512-3
- 4-7-512-512-3
- 1-14-256-256-1
- 4-14-256-256-1

Values:
- $1.6E-04$
- $1.2E-04$
- $8E-05$
- $4E-05$
- $0E+00$
High Level: Scaling Automatic Performance Profiling
Low Level: Portable RPC Tracker + Server

Resource Manager (Tracker)

Nvidia GPU Server
- RPC RT
- CUDA tasks

AMD GPU Server
- RPC RT
- ROCm tasks

Zynq FPGA board
- JIT driver
- Hardware bitstream

Android Phone
- RPC RT
- OpenCL tasks

Raspberry Pi
- RPC RT
- ARM tasks

Shared cluster of heterogeneous devices

Optimization Service
- cross compiler
- RPC client

ML-based cost model
- Prioritizer

Running optimization services

Resource Allocation

RPC Session Data Path

Red modules can be reconfigured remotely in each session

Prioritizer

Workload 1
- Workload 2
- Workload 3

Optimization Service

Cross compiler
- RPC client

Workload

Red modules

Prioritizer
RPC Communication Flow

Client → Tracker → Device

- Upload code
- Run code
- Return run time
RPC Communication Flow

Client

Tracker

Device

device free

Client

Device

upload code

run code

return run time
RPC Communication Flow

Client -> Tracker -> Device

- Client: upload code
- Tracker: device free
- Device: run code
- Device: return run time

Client -> Device

- Client: run code
- Device: return run time
RPC Communication Flow

Client

Tracker

Device

request device

device free

upload code

run code

return run time
RPC Communication Flow

Client → Tracker
-request device
Tracker ← Device
-device free

Client → Device
-upload code
Device ← Client
-run code
Device ← Client
-return run time
RPC Communication Flow

Client → Tracker: request device
Tracker → Device: device free

Client → Device: upload code
Device → Client: run code
Device → Client: return run time
RPC Communication Flow

Client → Tracker:
- request device
- return handle

Tracker ← Device:
- device free

Client → Device:
- upload code
- run code
- return run time

Device ← Client:
Model to Tuned Implementation

Model → operator extraction → Bag of Operators → AutoTVM tuning → Tuned Model
Handcrafted Schedule Templates

conv2d, x86

conv2d, GPU, winograd

conv2d, ARM, spatial packing