CS 744: TVM

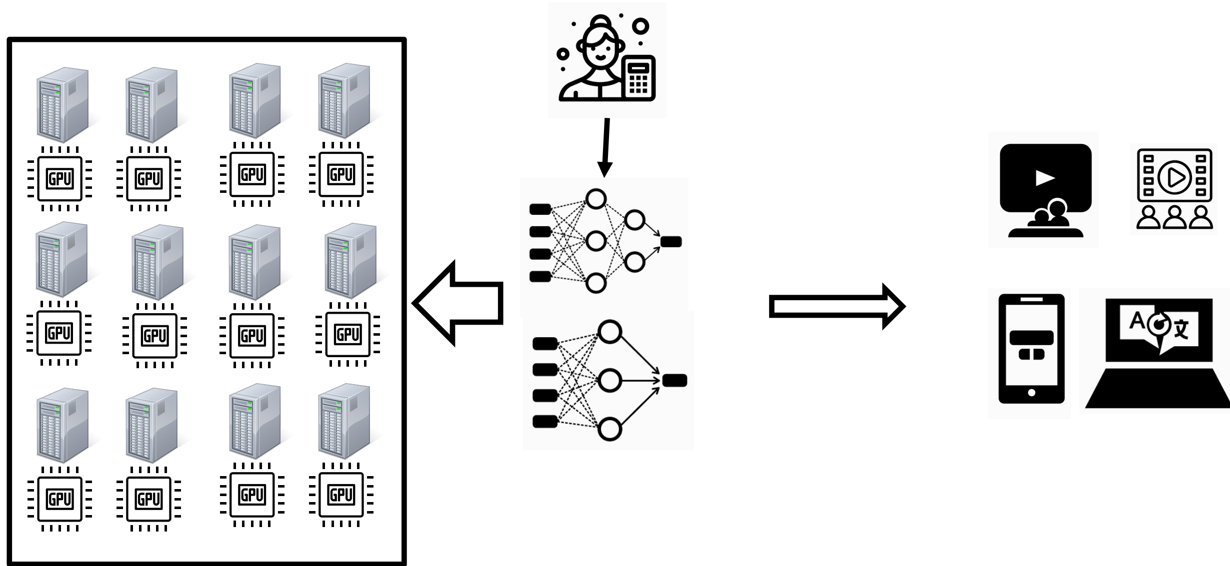
Shivaram Venkataraman

Fall 2020

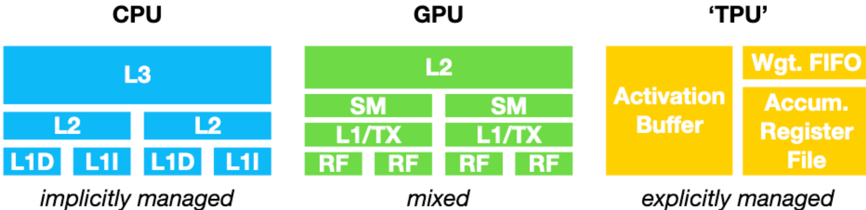
ADMINISTRIVIA

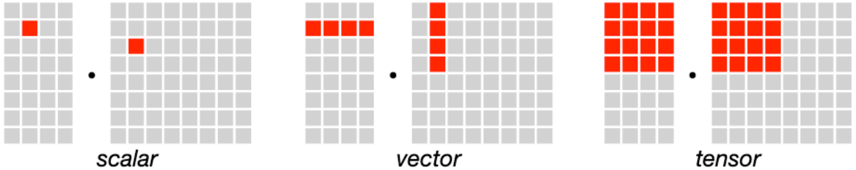
- Course project titles
- Project proposal aka Introduction (10/16)
  - Introduction
  - Related Work
  - Timeline (with eval plan)
- Midterm: Oct 22

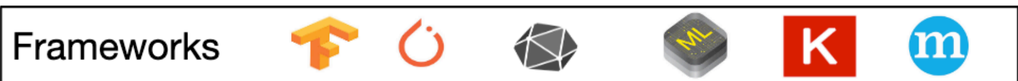
MACHINE LEARNING: STACK

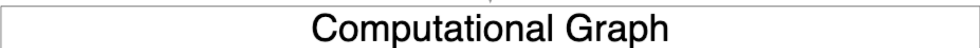


MOTIVATION: PERFORAMNCE PORTABILITY



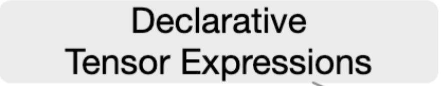



Frameworks
K
m

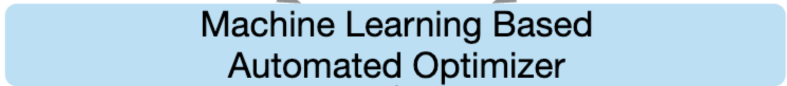

Computational Graph

Section 3
High Level Graph Rewriting

Optimized Computational Graph
Operator-level Optimization and Code Generation


Section 4
Declarative
Tensor Expressions

Hardware-Aware
Optimization Primitives

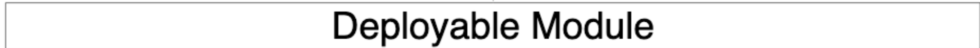

Section 5
Machine Learning Based
Automated Optimizer

Optimized Low Level Loop Program

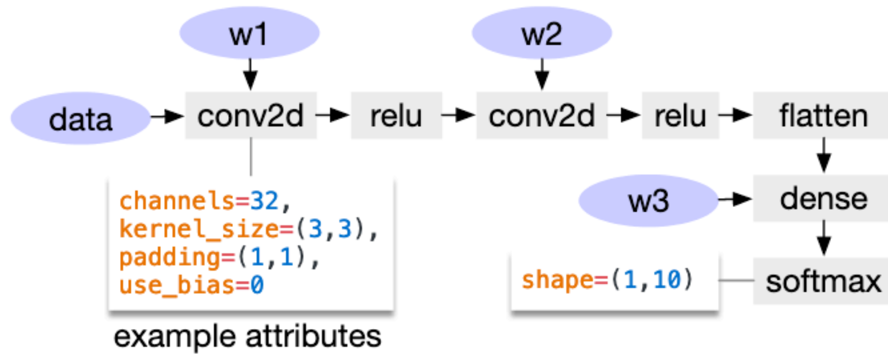
Accelerator Backend

LLVM IR

CUDA/Metal/OpenCL


Deployable Module

OPTIMIZATION COMPUTATION GRAPHS



Operator Fusion

Data layout

TENSOR EXPRESSION LANGUAGE

```
m, n, h = t.var('m'), t.var('n'), t.var('h')
A = t.placeholder((m, h), name='A')
B = t.placeholder((n, h), name='B')
k = t.reduce_axis((0, h), name='k')
C = t.compute((m, n), lambda y, x:
                    t.sum(A[k, y] * B[k, x], axis=k))
```

computing rule

result shape

Common Arithmetic, Math operations

Know the shape of the output and the data accessed

CODE GENERATION

```
for thread_group (by, bx) in cross(64, 64):
  for thread_item (ty, tx) in cross(2, 2):
    local CL[8][8] = 0
    shared AS[2][8], BS[2][8]
    for k in range(1024):
      for i in range(4):
        AS[ty][i*4+tx] = A[k][by*64+ty*8+i*4+tx]
      for each i in 0..4:
        BS[ty][i*4+tx] = B[k][bx*64+ty*8+i*4+tx]
      memory_barrier_among_threads()
```

Nested parallelism

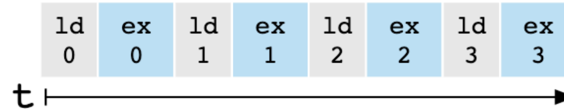
```
def gemm_intrin_lower(inputs, outputs):
   ww_ptr = inputs[0].access_ptr(“r")
   xx_ptr = inputs[1].access_ptr("r")
   zz_ptr = outputs[0].access_ptr("w")
   compute = t.hardware_intrin("gemm8x8", ww_ptr, xx_ptr, zz_ptr)
   reset = t.hardware_intrin("fill_zero", zz_ptr)
   update = t.hardware_intrin("fuse_gemm8x8_add", ww_ptr, xx_ptr, zz_ptr)
   return compute, reset, update

gemm8x8 = t.decl_tensor_intrin(y.op, gemm_intrin_lower)
```

Tensorization

LATENCY HIDING

What is the goal?

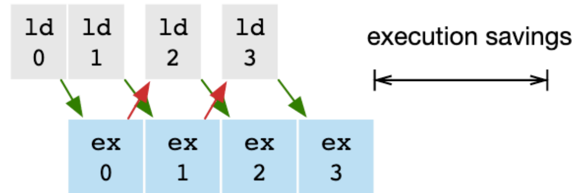
Monolithic Pipeline

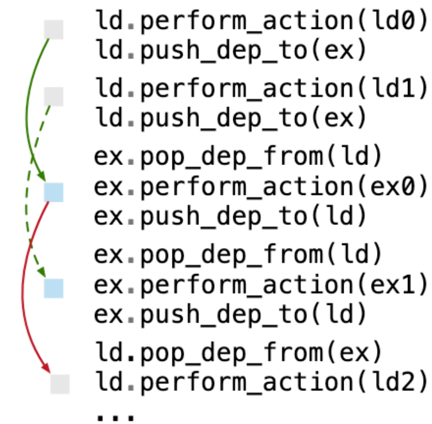


Instruction Stream

```
ld.perform_action(ld0)
ex.perform_action(ex0)
ld.perform_action(ld1)
ex.perform_action(ex1)
...
```

Decoupled Access-Execute Pipeline





read after write (RAW) dependence

write after read (WAR) dependence

AUTOMATING OPTIMIZATION

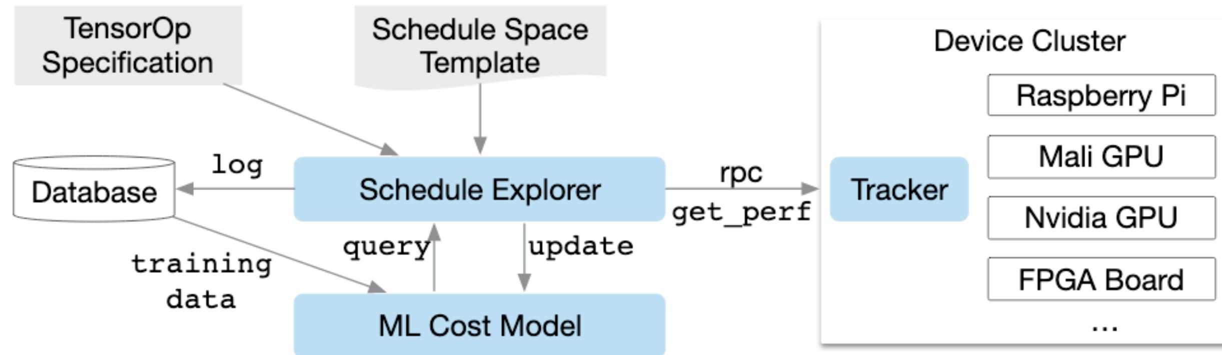
Goal: Create a specialized operator for input shape and layout

Challenge:

Choose appropriate schedule optimizations

Tiling size, loop unrolling

Automate the optimizer!



ML-BASED COST MODEL

Machine Learning Model Design Choices

- Speed: Faster than time it takes to evaluate a config
- Quality: Use a rank objective to predict the relative order of runtime

Gradient tree boosting model

- memory access count
- reuse ratio of each memory buffer at each loop level
- one-hot encoding of loop annotations

ML-BASED COST MODEL

Iteration

- Select a batch of candidates
- Collect data
- Use as training data to update the model

How to select candidates?

Parallel Simulated Annealing

- Start from a random config
- Walk to a nearby config →
  - Successful if cost decreases Else Reject

DISTRIBUTED DEVICE POOL

Pool of devices to speed up profiling

RPC interface to run a trial on device

Share device pools for multiple graphs

SUMMARY

TVM: Compiler for ML inference models

Support high performance for range of models, hardware devices

Key ideas

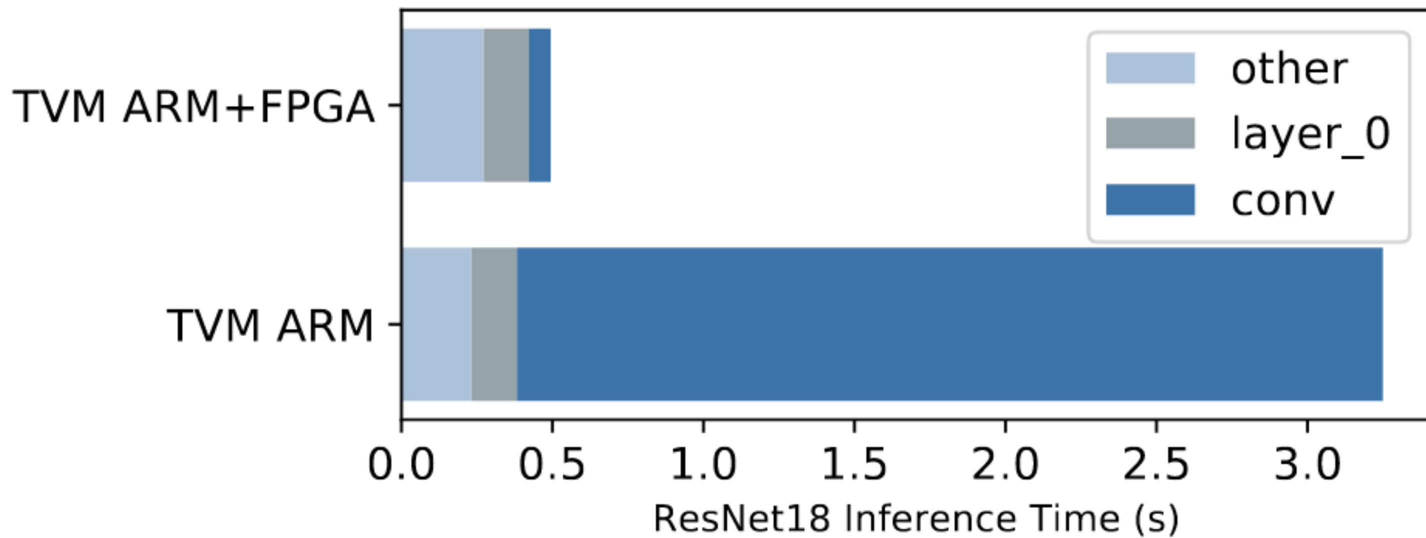
- Graph-level optimizations
- Tensor expression language: Code-gen, Latency hiding etc
- ML based Cost Model for automation

DISCUSSION

https://forms.gle/WiVgJ3abGXXgfBN99

Consider that you are building an optimizer for Spark programs instead of ML inference. What would be some configuration knobs that you could similarly tune? What might be different from the TVM optimizer?

What is your takeaway from the following figure?



NEXT STEPS

Next class: Ray

Course project: Oct 16 (introductions)

Midterm: Oct 22