FlexTensor:
An Automatic Schedule Exploration and Optimization Framework for Tensor Computation on Heterogeneous System

Size Zheng , Yun Liang , Shuo Wang,

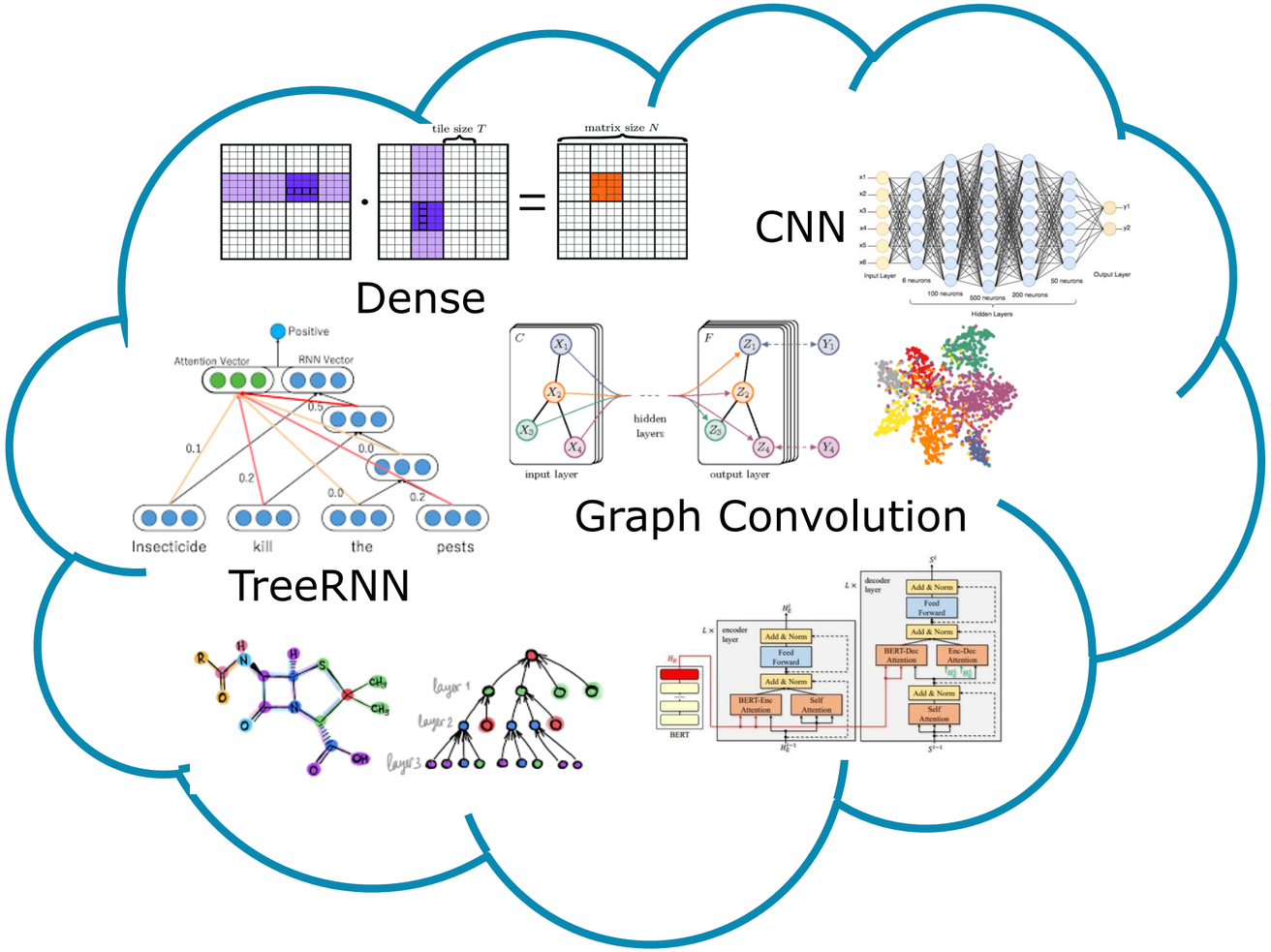
Renze Chen, Kaiwen Sheng

School of Electronics Engineering and Computer Science, Peking University, Beijing, China

Group URL: *http://ceca.pku.edu.cn/*

Email: *zhengsz@pku.edu.cn, ericlyun@pku.edu.cn*

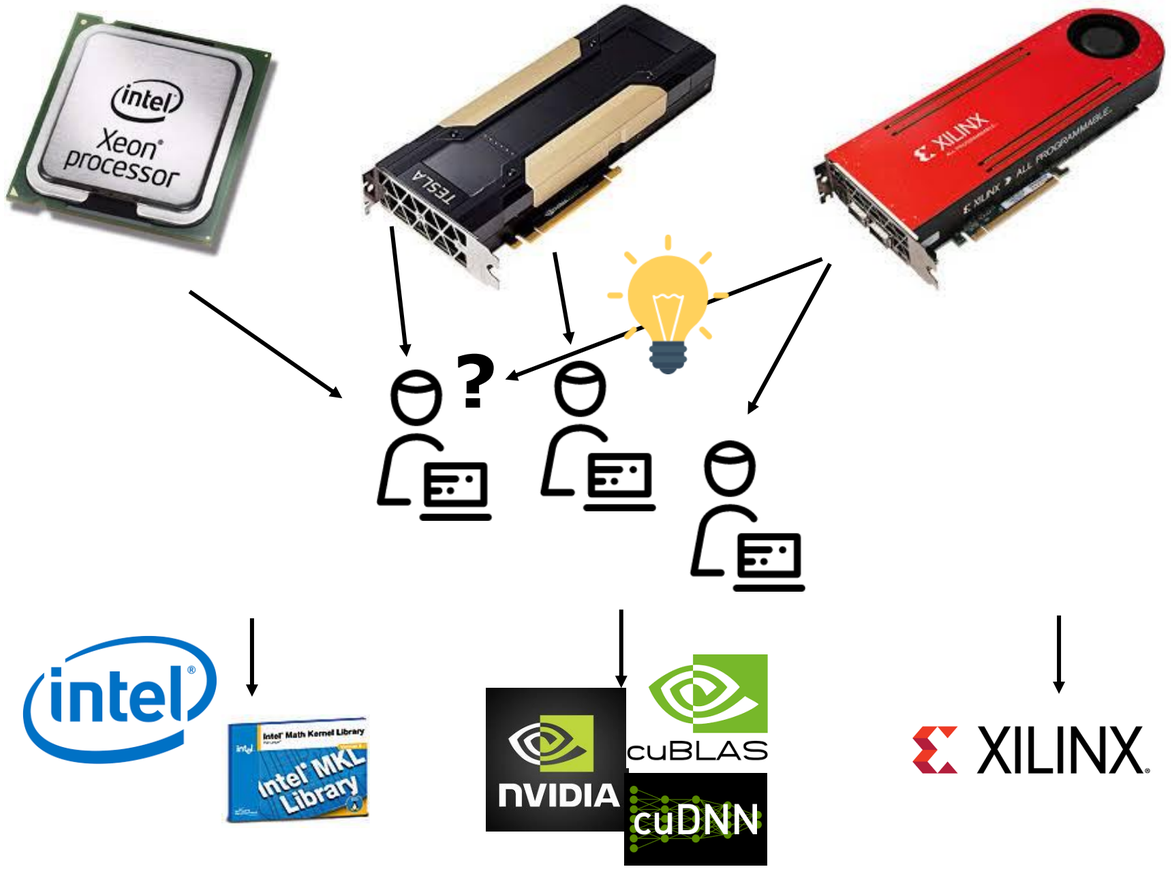
Tensor Computations are Everywhere!



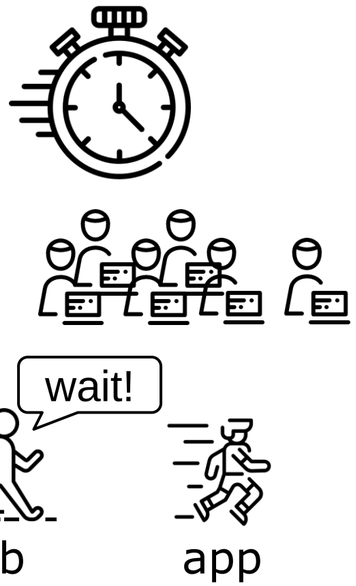
based on various of tensor operators

- GEMM
- Conv2d
- Conv3d
- ⋮
- Pooling

Heterogeneous System — Chance or Challenge?

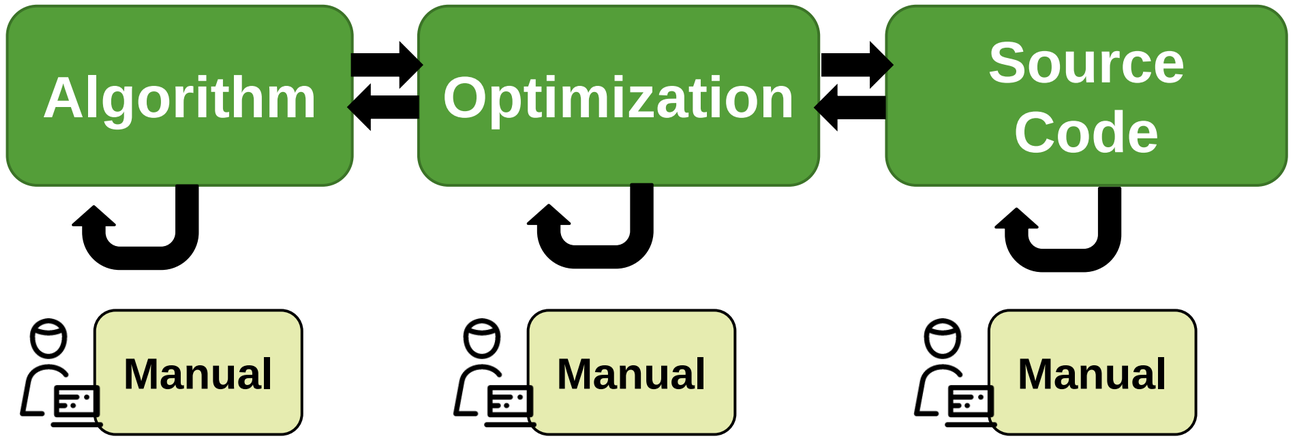


- Long time cost
- Lots of human efforts
- Too slow



manually designed libraries

Hand-Optimized Libraries



- long time cost
- hardware-specific
- expertise in everything

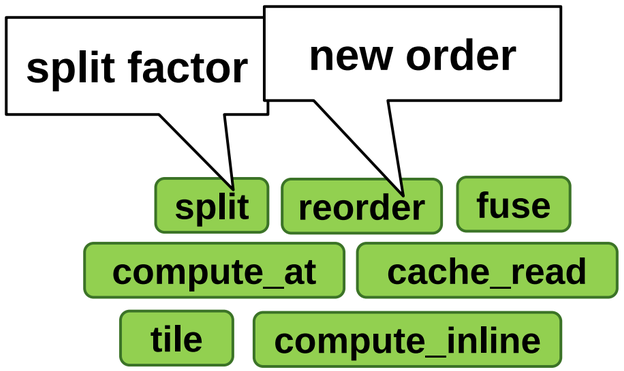
Code-Generation with Scheduling

Compute Description:

- High-level
- Algorithm
- Mathematic expressions

Scheduling:

- Primitives
- Hardware-specific
- Parameters for optimization



compute description

```
def vector_add ( A,  B ):
    C = compute ( ( 16, ),
          lambda i :  A [ i ] + B [ i ]
    return C
```

naïve source code

```
for ( int i=0; i < 16; i=i+1)
{
    C [ i ] = A [ i ] + B [ i ];
}
```

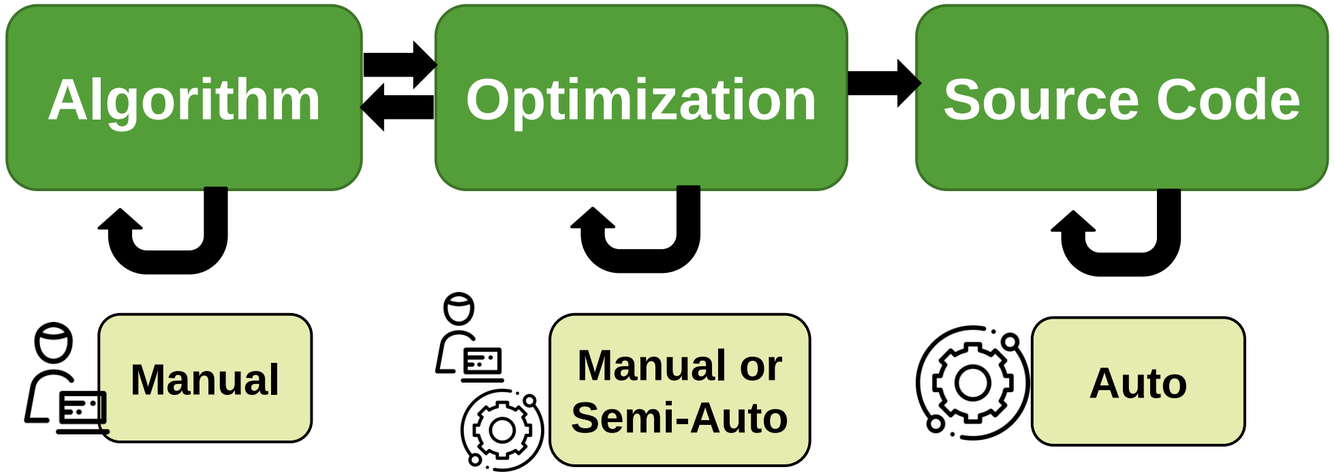
scheduling

```
C = vector_add ( A, B )
s = create_schedule ( C.op )
i = s [C].op.axis [0]
outer, inner =
     s [C].split ( i, factor=4 )
s [ C ].unroll ( inner )
```

optimized code

```
for (int outer=0; outer < 4; outer=outer+1)
{
  C [outer*4+0]= A [outer*4+0]+B [outer*4+0];
  C [outer*4+1]= A [outer*4+1]+B [outer*4+1];
  C [outer*4+2]= A [outer*4+2]+B [outer*4+2];
  C [outer*4+3]= A [outer*4+3]+B [outer*4+3];
}
```

Code-Generation with Scheduling



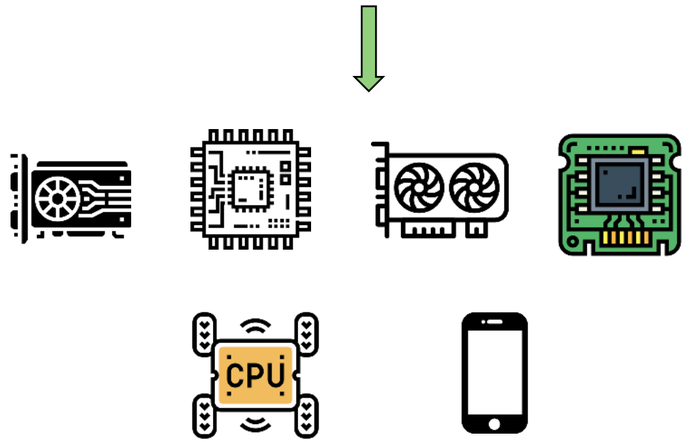
time cost for optimization

hardware-specific

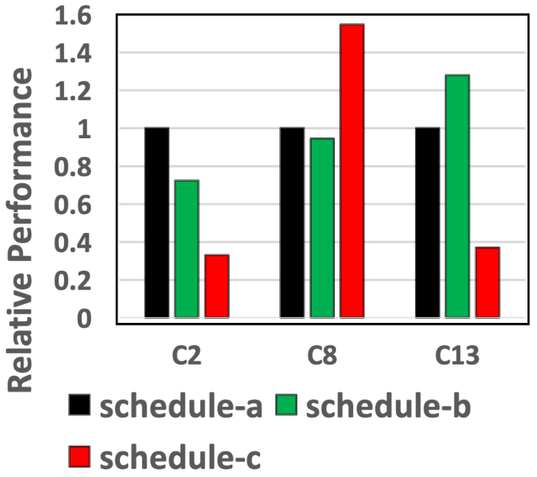
expertise in optimization

Halide

tvm

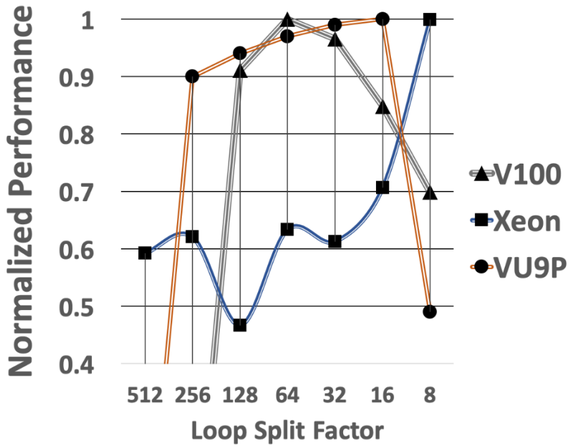


Challenges of Writing Schedules



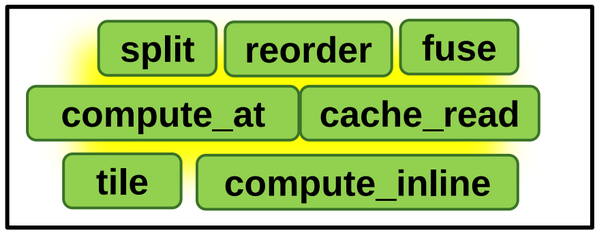
Hard to find optimal combinations

Same Operator:
- different input shapes
- different primitive combinations



Hard to find optimal parameters

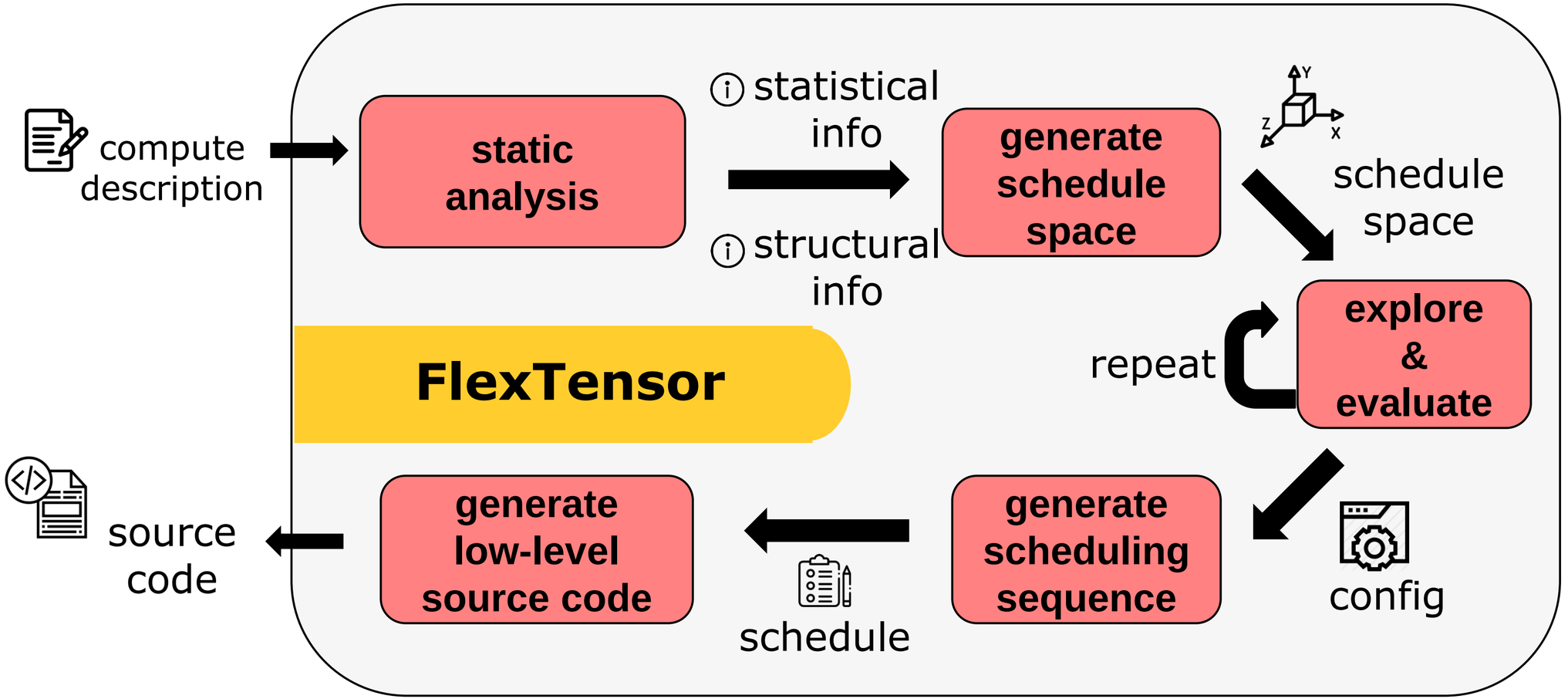
Same primitive combination:
- different devices
- different parameters for primitives



Hard to find construct search space

Large combination space:
- many different kinds of primitives
- parameter space for each primitive

FlexTensor


compute description
static analysis
statistical info
structural info
generate schedule space
schedule space
explore & evaluate
repeat
FlexTensor
generate scheduling sequence
config
schedule
generate low-level source code
source code

Static Analysis

```
C = compute ( ( 16, ),
    lambda i :  A [ i ] + B [ i ] )
```

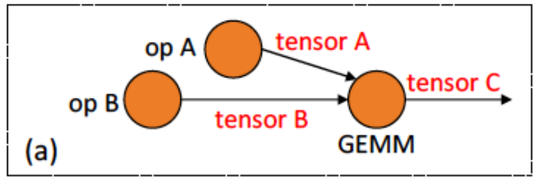
compute description

statistical information

- loop trip counts
- number of loops
- loop order

structural information

- computation graph structure
- producer consumer relationship




```
for i  in range ( 1024 ):
   for j  in range ( 1024 ):
      C [ i, j ] = 0  # initialize C
      for k  in range ( 1024 ):
         C [ i, j ] += A [ i , k ] * B [ k, j ]
```

(b)

(c)

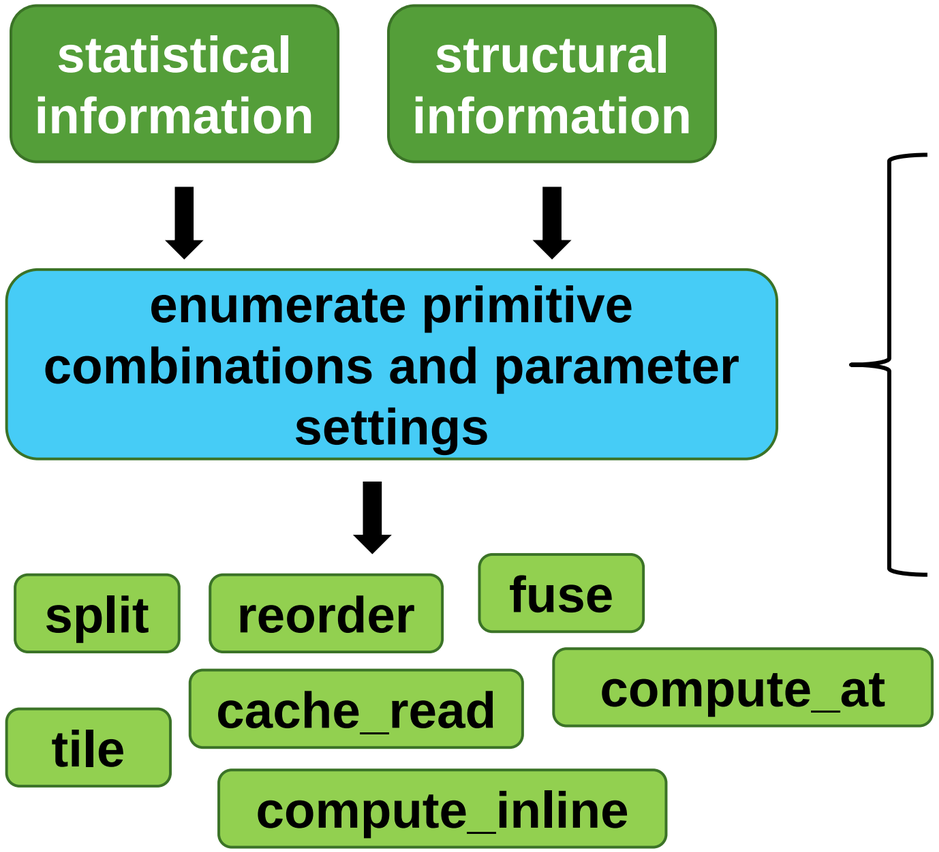
statistical	
# sl	2
# rl	1
stc	[1024, 1024]
rtc	[1024]
order	[i, j, k]

structural	
# node	3
# in	2
# out	1
# cs	0

An example of GEMM

- (a) GEMM mini-graph
- (b) GEMM high-level code
- (c) Statistical & structural info. from code (b)

Schedule Space Generation

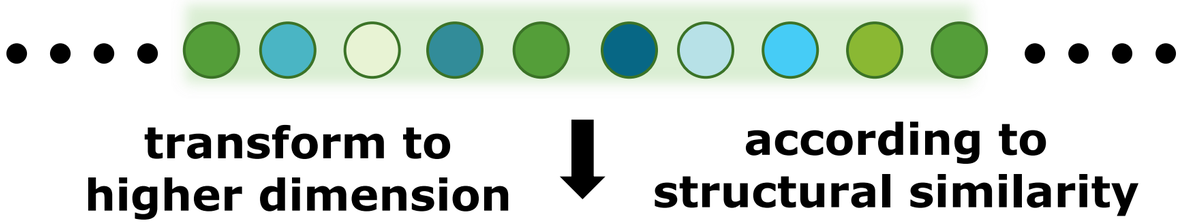


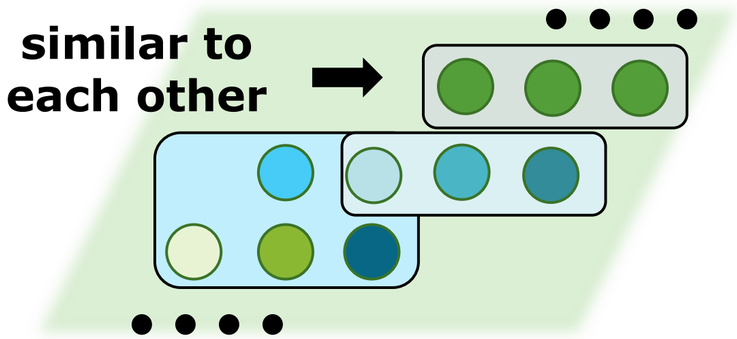
Principles

- limit the depth of primitives combination
- prune the parameter space
- pre-determine certain decisions for different hardware

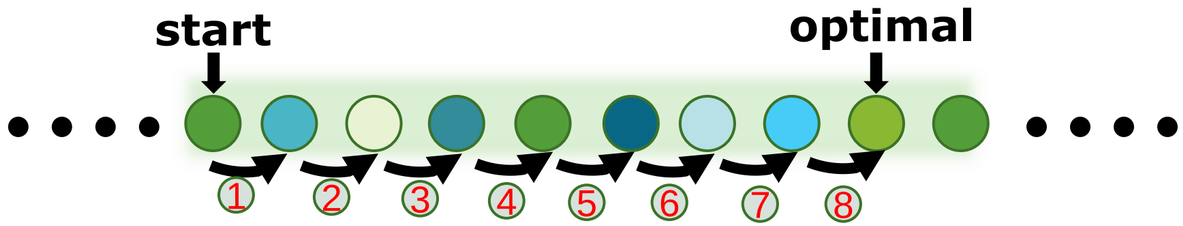
Schedule Space Generation

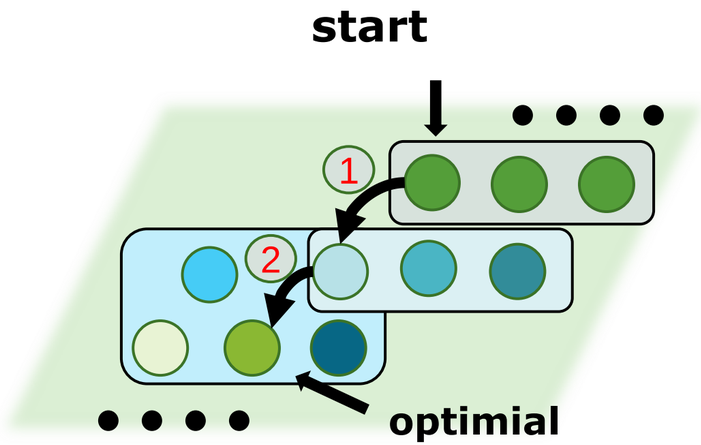
Exploit structural similarity





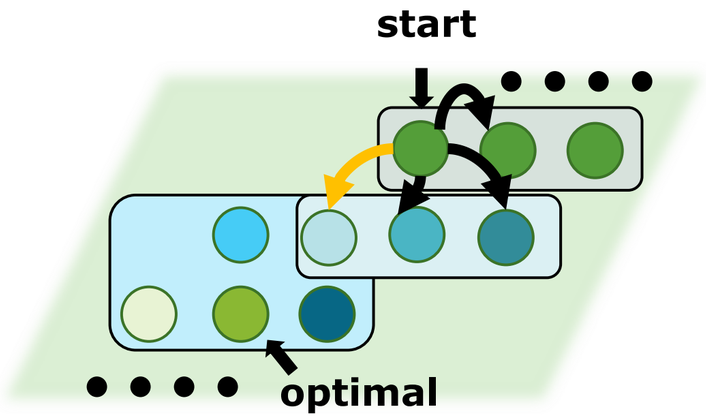
Rearrange the search space





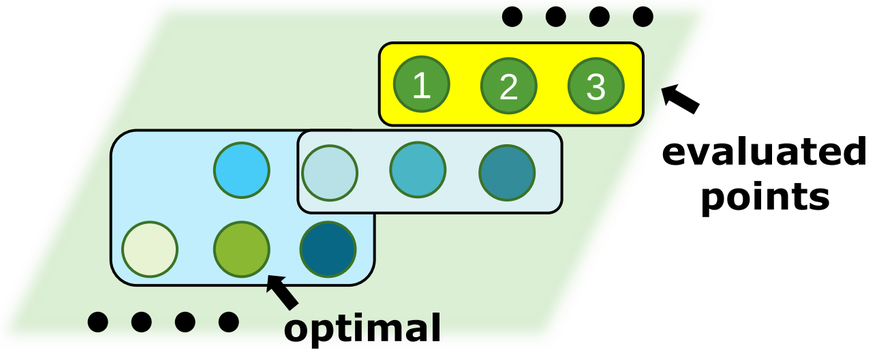
shorten the path from starting point to the optimal point

Efficient Exploration



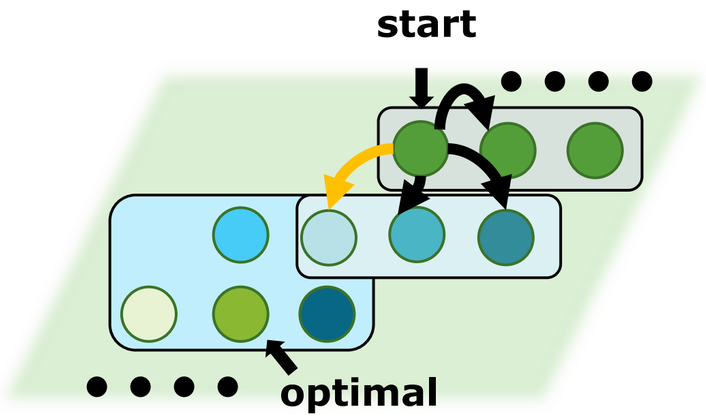
Which point to start with?
—**Heuristics: simulated annealing**
Which direction to search along?
—**Machine Learning: Q-Learning**
How to evaluate each point?
—**Run on target device**
—**Cost model**

Select the starting points



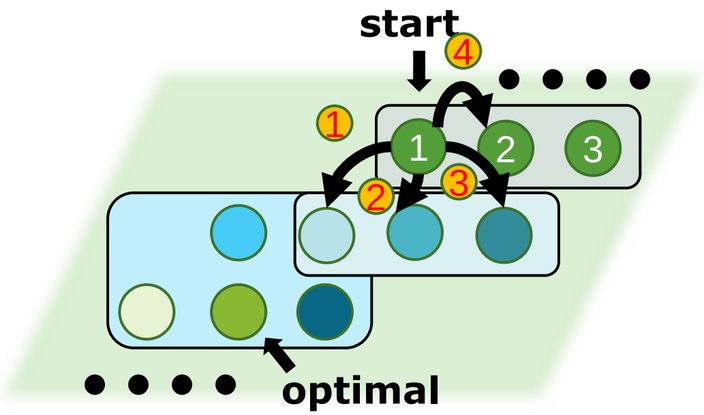
choose from 1, 2, and 3
known value: v^1, v^2, v^3
the best one known: v^*
choose according to possibility:
$e^{-\gamma\frac{(v^*-v^i)}{v^*}}, i = 1,2,3$
allow choosing multiple points

Efficient Exploration



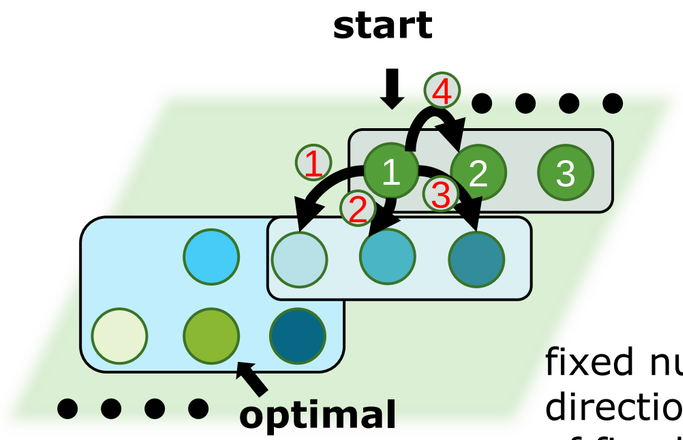
Which point to start with?
—**Heuristics: simulated annealing**
Which direction to search along?
—**Machine Learning: Q-Learning**
How to evaluate each point?
—**Run on target device**
—**Cost model**

Select the searching direction

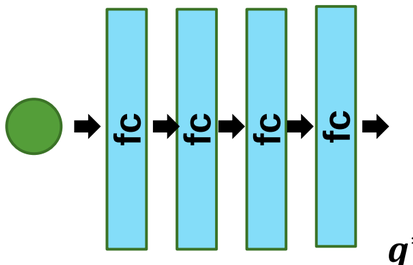


1. **keep record of visited points: discard** 4
2. **use DQN algorithm to predict Q-value of each direction:** q^1, q^2, q^3
3. **choose the largest one:** $q^* = max(q^i), i = 1, 2, 3$

The DQN Algorithm



fixed number of directions (because of fixed dimension)

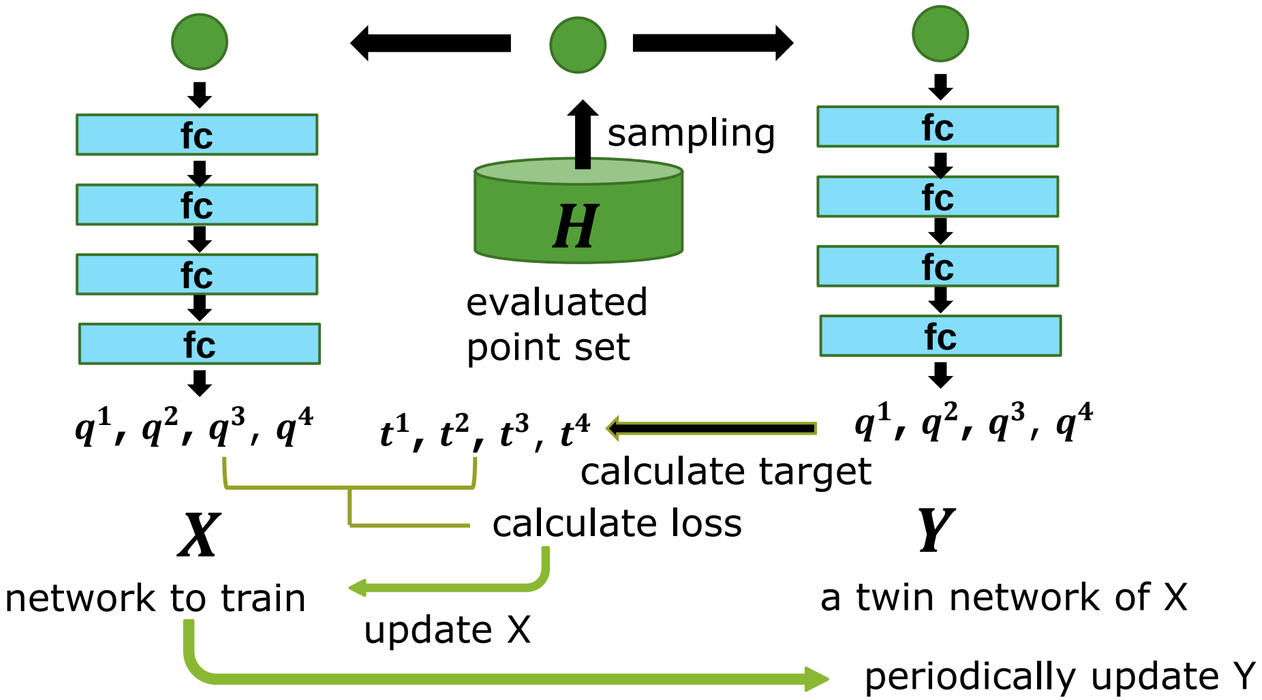


q^1, q^2, q^3, q^4

discard q^4

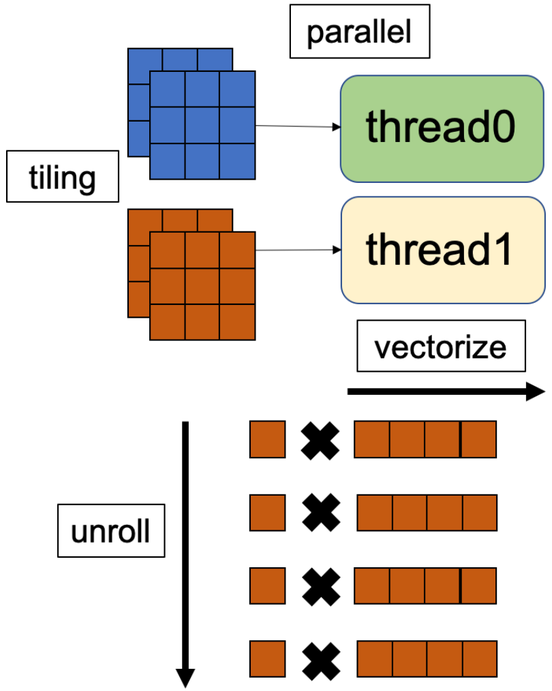
$q^* = max(q^i), i = 1, 2, 3$

Model Architecture

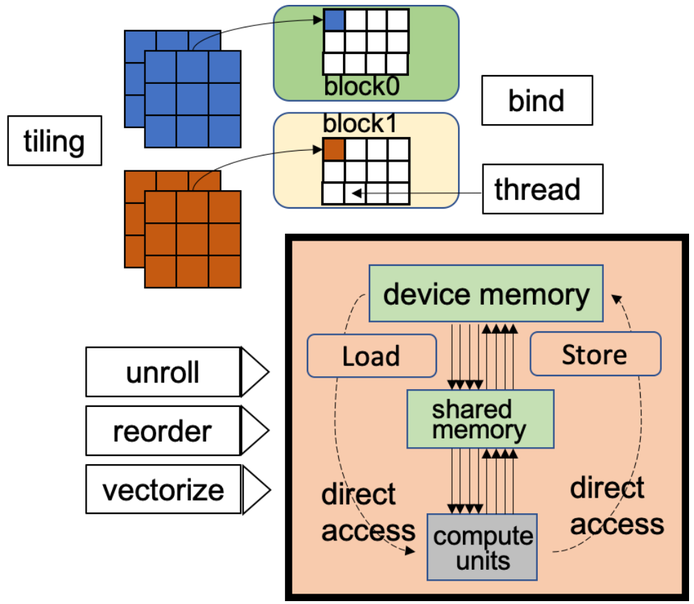


Training Method

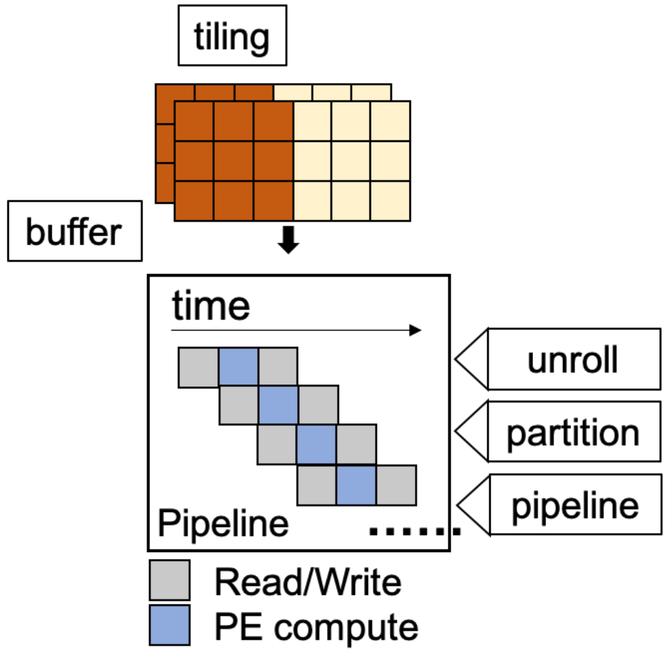
Schedule Generation



CPU

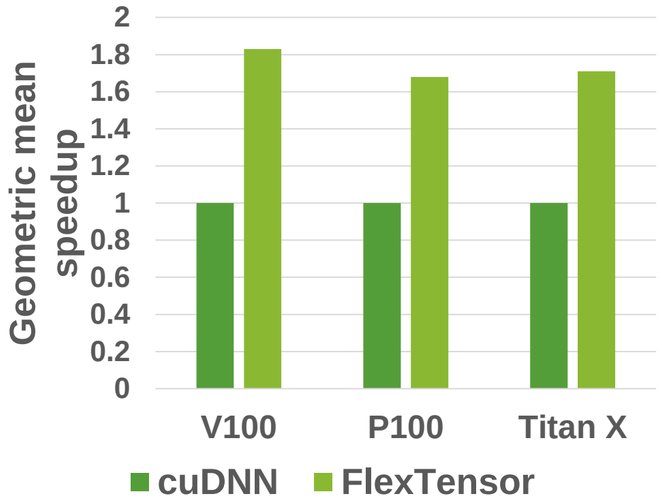


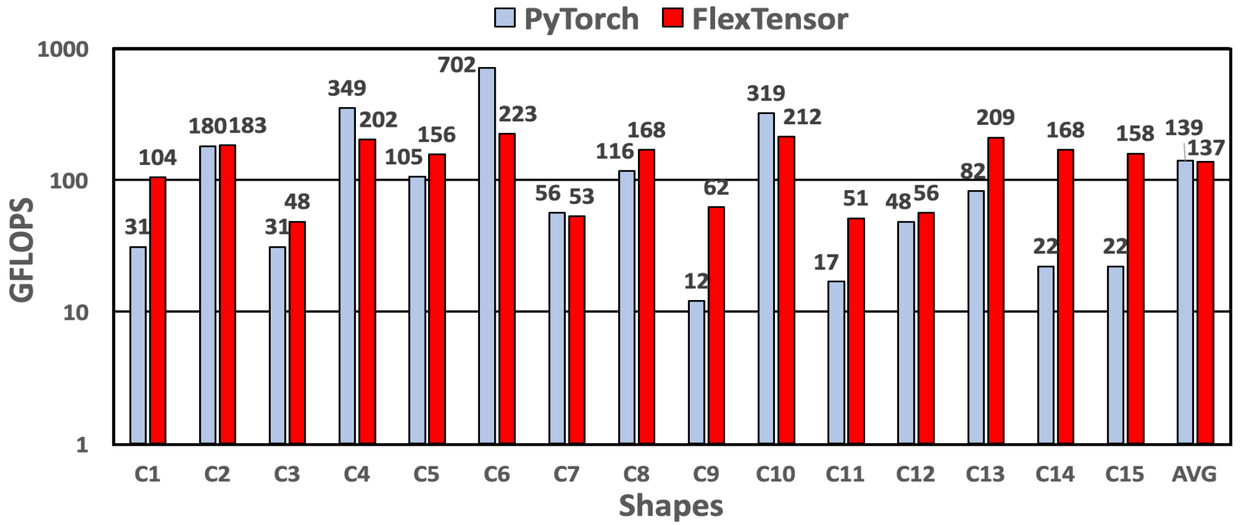
GPU



FPGA

Evaluation



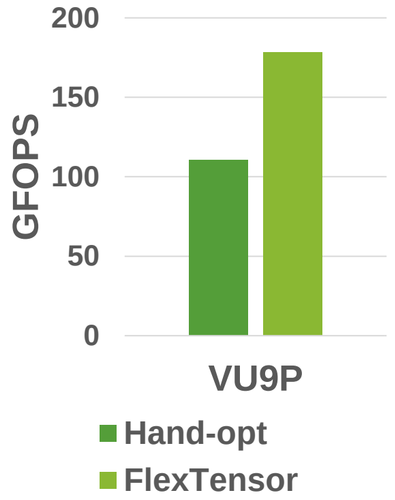


- **Three GPU platforms**
- **Speedup over cuDNN**
  - 1.83x on V100
  - 1.68x on P100
  - 1.71x on Titan X

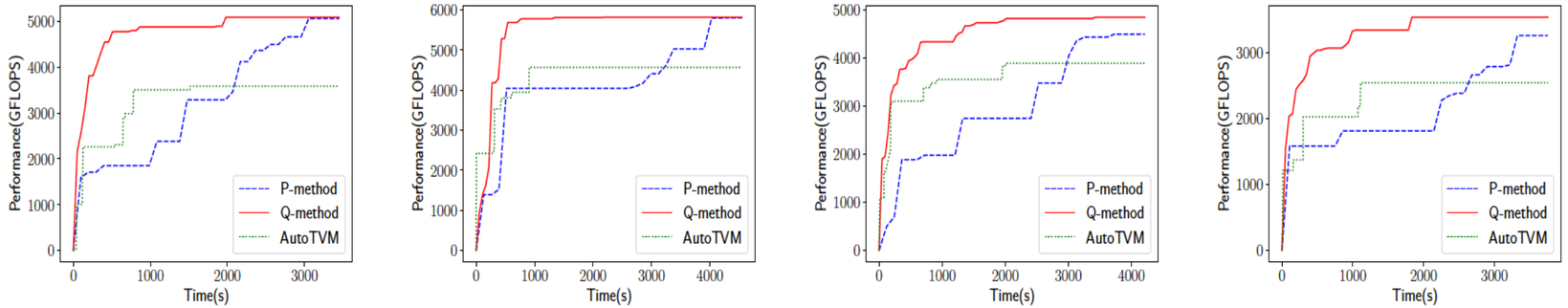
Speedup (Geometric mean):
- **1.72x** to MKL-DNN

Absolute performance (average):
- **MKL-DNN:** 139.49 GFLOPS
- **FlexTensor:** 136.91 GFLOPS

Evaluation



Exploration time VS. Performance



(a) C1 (b) C6 (c) C8 (d) C9

Speedup (Geometric mean):
- **1.5x** speedup over hand-opt

Absolute performance (average):
- **Hand-opt:** 110.42 GFLOPS
- **FlexTensor:** 178.16 GFLOPS

Q-method (red): our method
P-method (blue): ours without Q-learning
Baseline (green): AutoTVM

Q-method only use **27.6%** the time of AutoTVM.

Download and Install FlexTensor

Download

https://github.com/KnowingNothing/FlexTensor-Micro

Dependencies

python >= v3.5
TVM (a modified version) https://github.com/KireinaHoro/tvm.git
PyTorch

Install Steps

1. download and install tvm
2. pip install torch
3. setup the environments as showed in https://github.com/KnowingNothing/FlexTensor-Micro/blob/micro-tutorial/README.md

Docker Image

Refer to https://pku-ahs.github.io/tutorial/en/master/steps.html

Run FlexTensor

Off-the-shelf scripts

flextensor/optimize/optimize_*.py

Tutorial scripts

flextensor/tutorial/*

Source code structure

- flextensor/*.py **the main scheduler, task, and space**
- flextensor/baselines/*.py **baselines such as PyTorch and AutoTVM**
- flextensor/configs/*.py **the workload input shapes**
- flextensor/nn/ **the operator descriptions**

Optimize Conv2d on CPU

CPU platform

Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz

Optimization script

flextensor/tutorial/conv2d_llvm/optimize_conv2d.py

Command line arguments

python optimize_conv2d.py --help

--shapes: the convolution input shapes
(the name of networks)
--target: the target platform **llvm**
--log: the name of log file
(store the schedule configurations)
--parallel: multiprocessing in tuning

```
(sccc) [zchnc@scccc conv2d_llvm]$ python optimize_conv2d.py --help
usage: optimize_conv2d.py [-h] [-s SHAPES] [-f FROM_] [-t TO] [-l LOG]
                          [--test TEST] [--trials TRIALS] [--target TARGET]
                          [--device DEVICE] [--timeout TIMEOUT]
                          [--parallel PARALLEL] [--use_model]
                          [--method METHOD] [--host HOST]
                          [--target_host TARGET_HOST] [--port PORT]
                          [--force_inline] [--use_rpc]

optional arguments:
  -h, --help            show this help message and exit
  -s SHAPES, --shapes SHAPES
                        Use which shapes [yolo, google, res, squeeze, vgg-16,
                        vgg-19]
  -f FROM_, --from_ FROM_
                        From which shape
  -t TO, --to TO        To which shape
  -l LOG, --log LOG     Log file name
  --test TEST           test file name
  --trials TRIALS       number of trials for op
  --target TARGET       target device type
  --device DEVICE       target device number
  --timeout TIMEOUT     timeout
  --parallel PARALLEL   parallel
  --use_model           use performance model
  --method METHOD       how to schedule
  --host HOST
  --target_host TARGET_HOST
  --port PORT
  --force_inline
  --use_rpc
```

Run Optimized Conv2d on CPU

Command

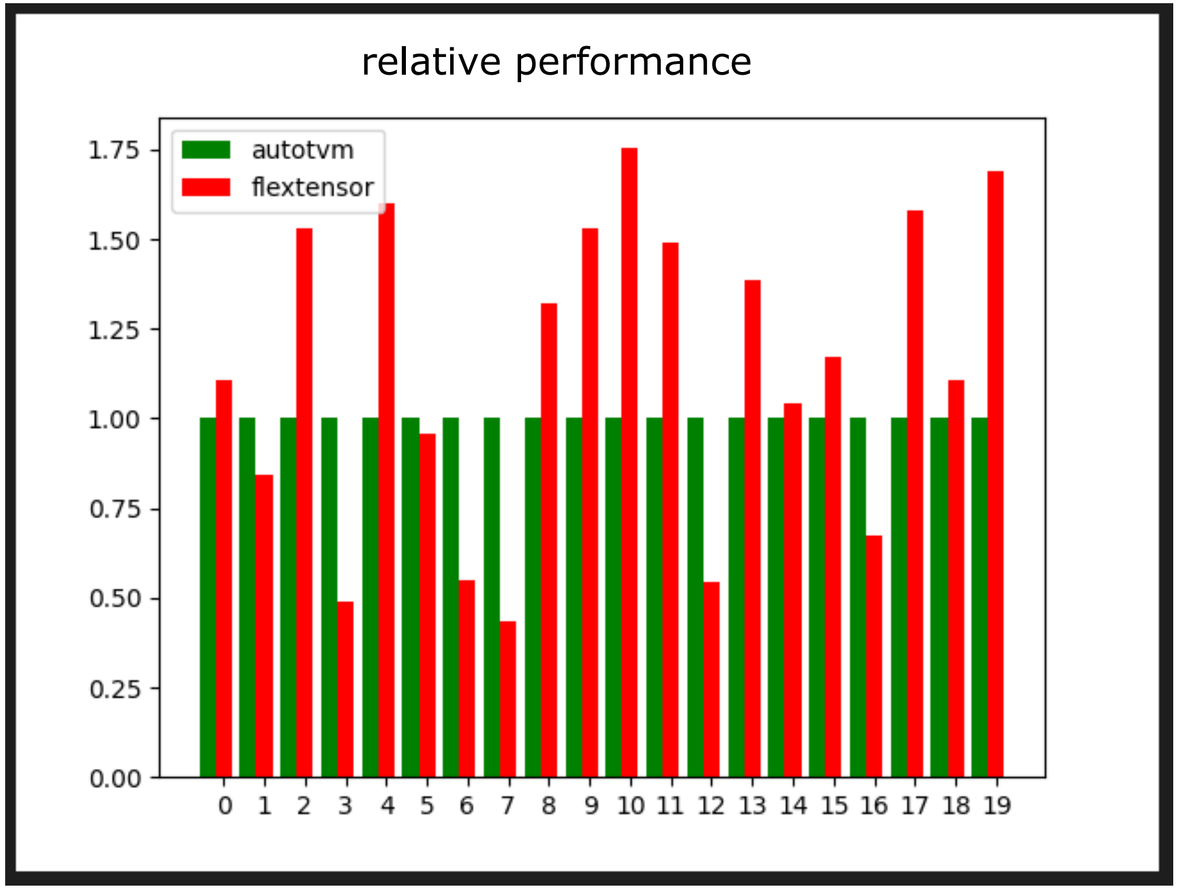
python optimize_conv2d.py --test <name of log file>

Plot the results

python plot.py

Geomean Speedup

1.04X



Optimize Conv2d on GPU

GPU platform

Nvidia V100 GPU

Optimization script

flextensor/tutorial/conv2d_cuda/optimize_conv2d.py

Command line arguments

python optimize_conv2d.py --help

--shapes: the convolution input shapes
(the name of networks)
--target: the target platform **cuda**
--log: the name of log file
(store the schedule configurations)
--parallel: multiprocessing in tuning

```
(sccc) [zchnc@scccc conv2d_llvm]$ python optimize_conv2d.py --help
usage: optimize_conv2d.py [-h] [-s SHAPES] [-f FROM_] [-t TO] [-l LOG]
                          [--test TEST] [--trials TRIALS] [--target TARGET]
                          [--device DEVICE] [--timeout TIMEOUT]
                          [--parallel PARALLEL] [--use_model]
                          [--method METHOD] [--host HOST]
                          [--target_host TARGET_HOST] [--port PORT]
                          [--force_inline] [--use_rpc]

optional arguments:
  -h, --help            show this help message and exit
  -s SHAPES, --shapes SHAPES
                        Use which shapes [yolo, google, res, squeeze, vgg-16,
                        vgg-19]
  -f FROM_, --from_ FROM_
                        From which shape
  -t TO, --to TO        To which shape
  -l LOG, --log LOG     Log file name
  --test TEST           test file name
  --trials TRIALS       number of trials for op
  --target TARGET       target device type
  --device DEVICE       target device number
  --timeout TIMEOUT     timeout
  --parallel PARALLEL   parallel
  --use_model           use performance model
  --method METHOD       how to schedule
  --host HOST
  --target_host TARGET_HOST
  --port PORT
  --force_inline
  --use_rpc
```

Run Optimized Conv2d on GPU

Command

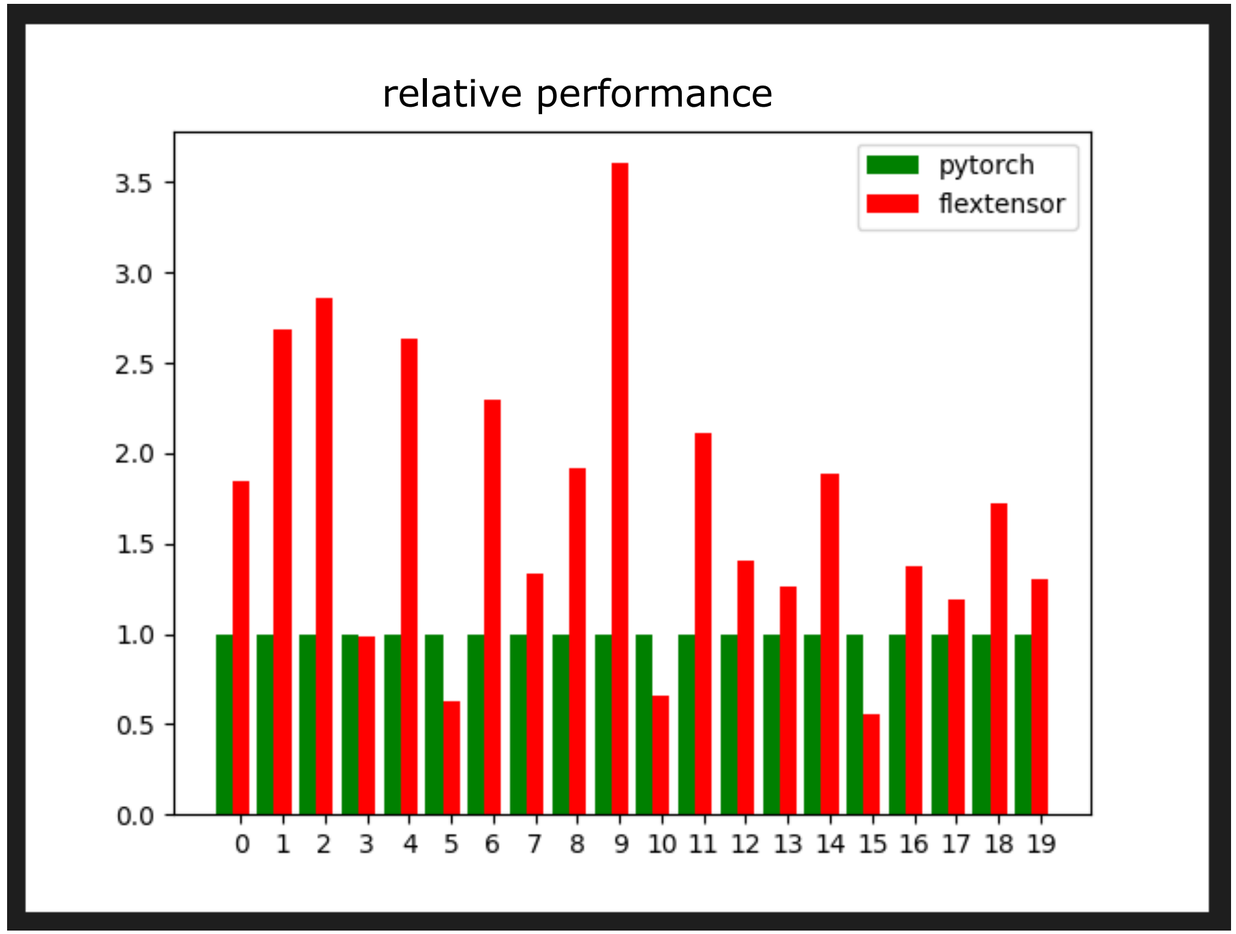
python optimize_conv2d.py --test <name of log file>

Plot the results

python plot.py

Geomean Speedup

1.52X



Adding New Optimization Tasks to FlexTensor

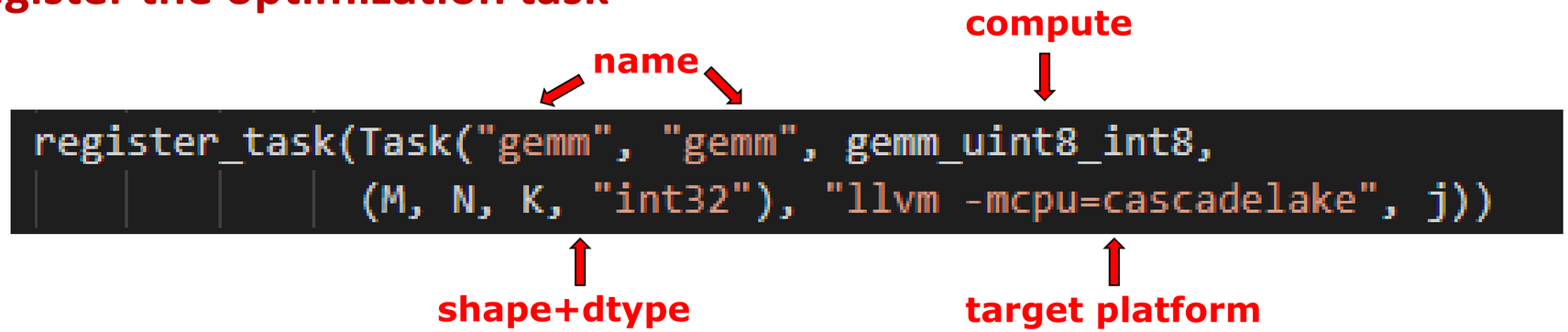
File to change

flextensor/task.py

Describe a new compute

```
def gemm_uint8_int8(i, j, k, dtype="int32"):
    a = tvm.placeholder((i, k, 16, 4), name='a', dtype="int8")
    b = tvm.placeholder((k, j, 4), name='b', dtype="uint8")
    kk = tvm.reduce_axis((0, k), name='k')
    kki = tvm.reduce_axis((0, 4), name="ki")
    c = tvm.compute((i, j, 16), lambda ii, jj, iii:
                    tvm.sum(b[kk, jj, kki].astype(dtype) * a[ii, kk, iii, kki].astype(dtype),
                            axis=[kk, kki]), name='c')
    return [c.op], [b, a, c]
```

Register the optimization task



Adding One New Scheduler to FlexTensor

File to change

flextensor/scheduler.py

Design a schedule generator

get parameters from **config**

use **config** to determine
schedule primitive combinations

```
def _vnni_schedule_simple(s, op, op_state):
    # prepare extents
    sp_extents = [to_int(x.dom.extent) for x in op.axis]
    if hasattr(op, "reduce_axis"):
        re_extents = [to_int(x.dom.extent) for x in op.reduce_axis]
    else:
        re_extents = []

    if "intrin" in config:
        target, ind, slist, rlist = config["intrin"][0]
        intrin = INTRIN_TABLE[target][ind]
    else:
        intrin = None
        s_list = []
        r_list = []

    sp_factors = []
    re_factors = []
    # spatial split

    if "spatial" in config:···
    else:···

    # reduce split
    if "reduce" in config and hasattr(op, "reduce_axis"):···
    elif hasattr(op, "reduce_axis"):···
    else:
        sub_re_axis_list = []
    # match intrinsic
    def rearrange(lst):
        return list(zip(*lst))

    sub_sp_axis_list = rearrange(sub_sp_axis_list)
    sub_re_axis_list = rearrange(sub_re_axis_list)

    num_sp = len(sub_sp_axis_list) - 1
    num_re = len(sub_re_axis_list) - 1
    # inner-most
    inner_most = [sub_sp_axis_list[num_sp]]
    if num_re >= 0:
        inner_most.append(sub_re_axis_list[num_re])
    # do intrinsic
    if intrin is not None:···
    else:···
```

Advanced Optimization: Intrinsic Generation

Intrinsic Files

flextensor/intrinsic.py
flextensor/space.py

Add new intrinsic

Describe the intrinsic compute and IR generation

```
def intrinsic_gemv_uint8_int8_compute(M, K):
    int32_lanes = M  # 16 int32 lanes in AVX512
    num_int8_elements = K  # 4 int8 elements in int32
    data = tvm.placeholder((num_int8_elements,), dtype="uint8", name="data")
    kernel = tvm.placeholder(
        (int32_lanes, num_int8_elements), dtype="int8", name="kernel")
    k = tvm.reduce_axis((0, num_int8_elements), name="k")
    C = tvm.compute(
        (int32_lanes,),
        lambda i: tvm.sum(data[k].astype("int32") *
                          kernel[i, k].astype("int32"), axis=k),
        name="C",
    )
    return C, [data, kernel, C]
```

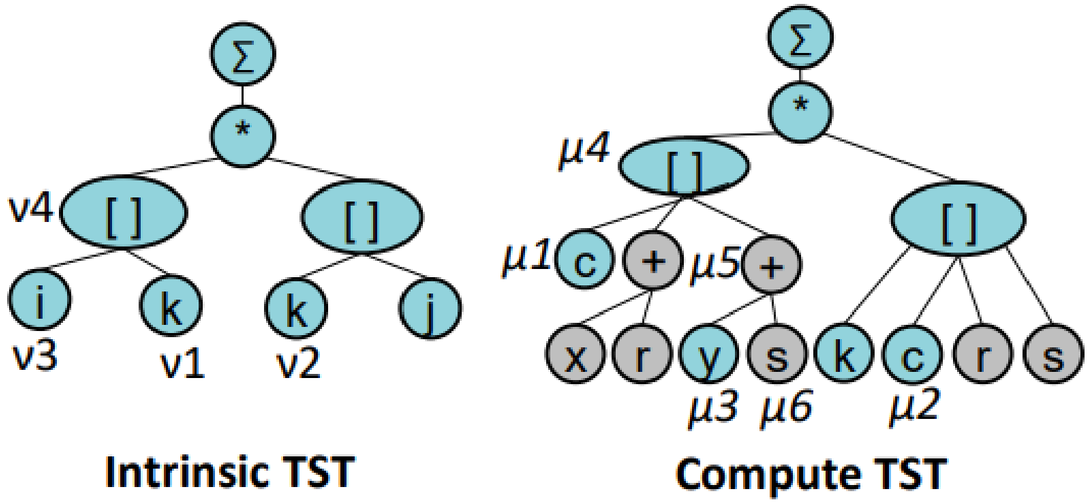
```
def intrinsic_gemv_uint8_int8_skylake(M, K):
    _, (data, kernel, C) = intrinsic_gemv_uint8_int8_compute(M, K)

    a_buffer = tvm.decl_buffer(data.shape, dtype='uint8', name="a_buffer",
                               offset_factor=1,
                               strides=[1])
    b_buffer = tvm.decl_buffer(kernel.shape, dtype='int8', name="b_buffer",
                               offset_factor=1,
                               strides=[tvm.var('ldw'), 1])

    def _intrin_func(ins, outs):…

    with tvm.build_config(offset_factor=1, partition_const_loop=True):
        return tvm.decl_tensor_intrin(C.op, _intrin_func, binds={data: a_buffer, kernel: b_buffer})
```

How FlexTensor Performs Pattern Matching?



AST Matching

O[n, p, **q, k**] += A[n, p+r, **q**+s, **c**] * B[r, s, **k, c**]

A GEMM within a Conv2d

O[i, j, **x**] += A[i, k, **z**] * B[j, k, **x, z**]

A GEMV within a Tensor-Tensor Multiplication

Optimize GEMM with VNNI instruction in FlexTensor

CPU platform

Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz

Optimization script

flextensor/tutorial/conv2d_vnni/optimize_gemm.py

Command line arguments

python optimize_gemm.py --help

--shapes: the convolution input shapes
(the name of networks)
--target: the target platform **llvm –mcpu=cascadelake**
--log: the name of log file
(store the schedule configurations)
--parallel: multiprocessing in tuning
--dtype: data type **int32**
--target_host: the host platform for compilation
llvm –mcpu=cascadelake

```
(sccc) [zchno@scccc conv2d_llvm]$ python optimize_conv2d.py --help
usage: optimize_conv2d.py [-h] [-s SHAPES] [-f FROM_] [-t TO] [-l LOG]
                          [--test TEST] [--trials TRIALS] [--target TARGET]
                          [--device DEVICE] [--timeout TIMEOUT]
                          [--parallel PARALLEL] [--use_model]
                          [--method METHOD] [--host HOST]
                          [--target_host TARGET_HOST] [--port PORT]
                          [--force_inline] [--use_rpc]

optional arguments:
  -h, --help            show this help message and exit
  -s SHAPES, --shapes SHAPES
                        Use which shapes [yolo, google, res, squeeze, vgg-16,
                        vgg-19]
  -f FROM_, --from_ FROM_
                        From which shape
  -t TO, --to TO        To which shape
  -l LOG, --log LOG     Log file name
  --test TEST           test file name
  --trials TRIALS       number of trials for op
  --target TARGET       target device type
  --device DEVICE       target device number
  --timeout TIMEOUT     timeout
  --parallel PARALLEL   parallel
  --use_model           use performance model
  --method METHOD       how to schedule
  --host HOST
  --target_host TARGET_HOST
  --port PORT
  --force_inline
  --use_rpc
```

Run Optimized GEMM with VNNI instruction in FlexTensor

Command

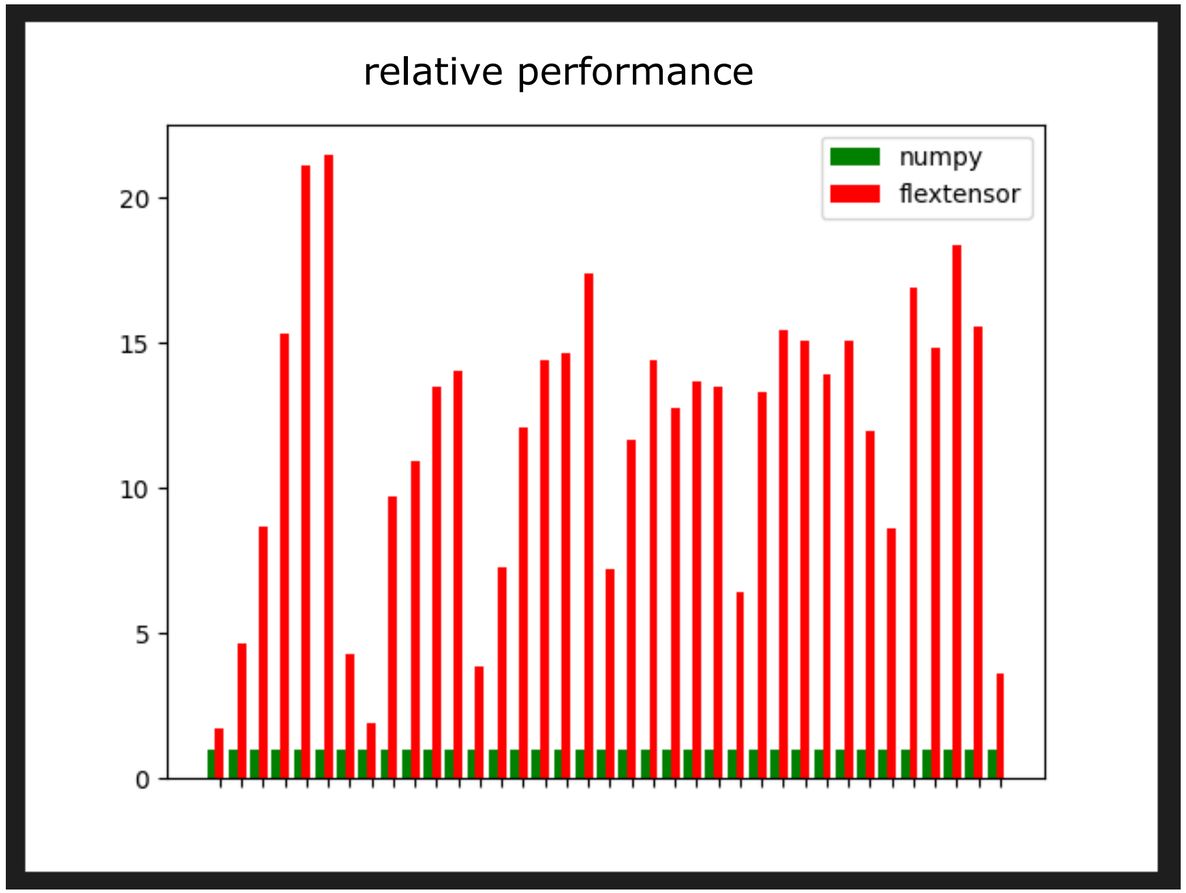
python optimize_gemm.py --test <name of log file>

Plot the results

python plot.py

Geomean Speedup

10.31X



Inspect the generated IR

Command

python optimize_gemm.py --test <name of log file> --dump_ir

```
produce c {
  parallel (ii.outer.jj.outer.fused, 0, 128) {
    for (ii.inner.init, 0, 16) {
      c[ramp((((floordiv(ii.outer.jj.outer.fused, 32)*8192) + (ii.inner.init*512)) + (floormod(ii.outer.jj.outer.fused, 32)*16
)), 1, 16)] = x16(0)
    }
    for (k.outer, 0, 2) {
      for (ii.inner, 0, 16) {
        for (k.inner, 0, 16) {
          c[ramp((((floordiv(ii.outer.jj.outer.fused, 32)*8192) + (ii.inner*512)) + (floormod(ii.outer.jj.outer.fused, 32)*16)
), 1, 16)] = (llvm_intrin((uint32)6507, (uint32)0, x16(0), x16(reinterpret(b[ramp((((k.outer*2048) + (k.inner*128)) + (floormo
d(ii.outer.jj.outer.fused, 32)*4)), 1, 4)])), reinterpret(a[ramp(((((floordiv(ii.outer.jj.outer.fused, 32)*32768) + (ii.inner*
2048)) + (k.outer*1024)) + (k.inner*64)), 1, 64)])) + c[ramp((((floordiv(ii.outer.jj.outer.fused, 32)*8192) + (ii.inner*512))
+ (floormod(ii.outer.jj.outer.fused, 32)*16)), 1, 16)])
        }
      }
    }
  }
}
```

Resources

Our Group

http://ceca.pku.edu.cn/

FlexTensor (original repo)

https://github.com/KnowingNothing/FlexTensor

FlexTensor for MICRO tutorial

https://github.com/KnowingNothing/FlexTensor-Micro

Publications

- *HASCO: Towards Agile HArdware and Software CO-design for Tensor Computation* **– ISCA'21**
- *FlexTensor: An Automatic Schedule Exploration and Optimization Framework for Tensor Computation on Heterogeneous System* **– ASPLOS'20**