Optimizing DNN Algorithms on Heterogenous Computing

Supervisors: Professor Jia, Jiaya & Professor Yu, Bei

Mentor: Mr. Qi, Sun

LJ2001

Department of Computer Science and Engineering

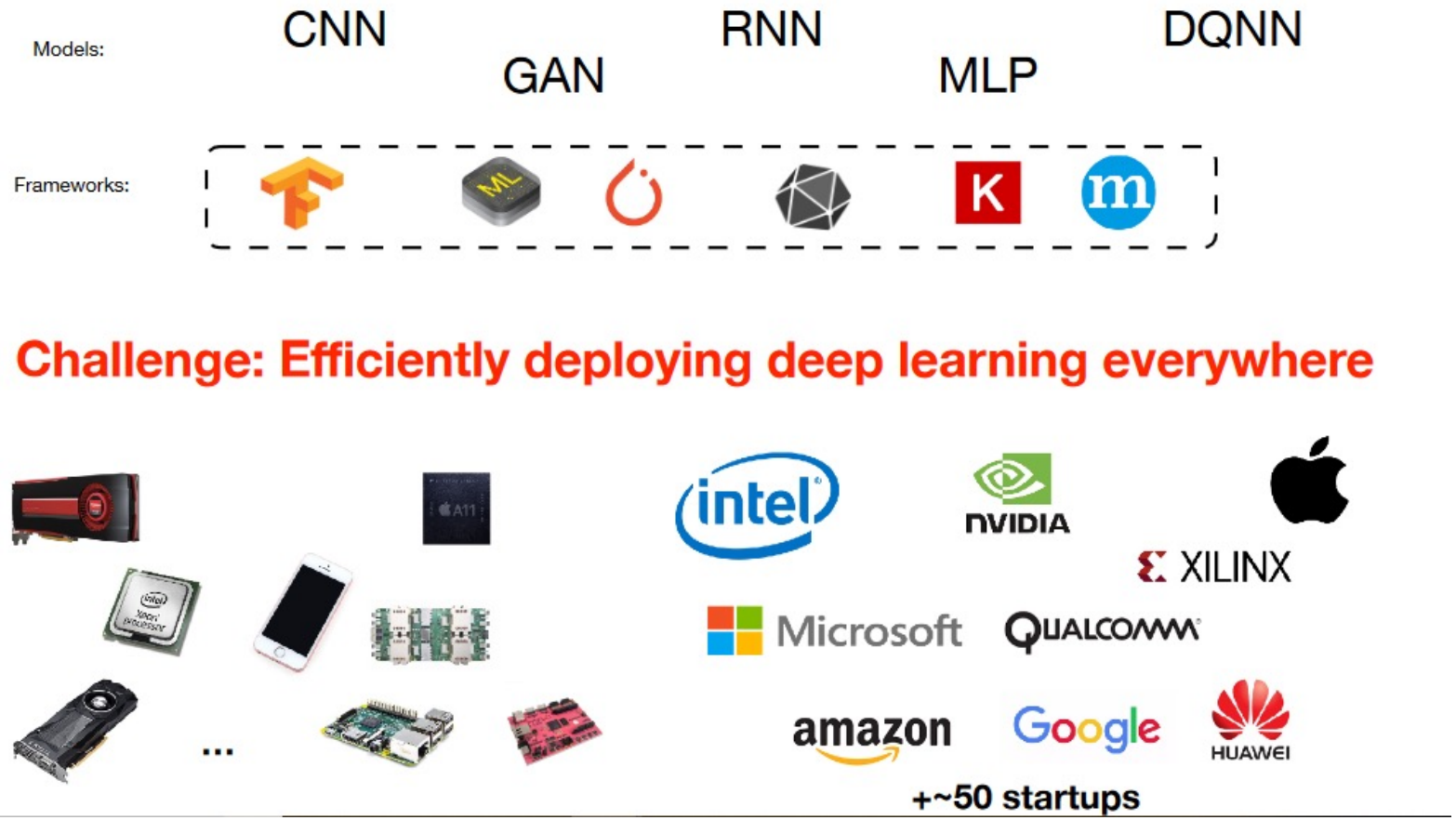
April 26, 2021

Presented by Choi, Jang Hyeon

Outline

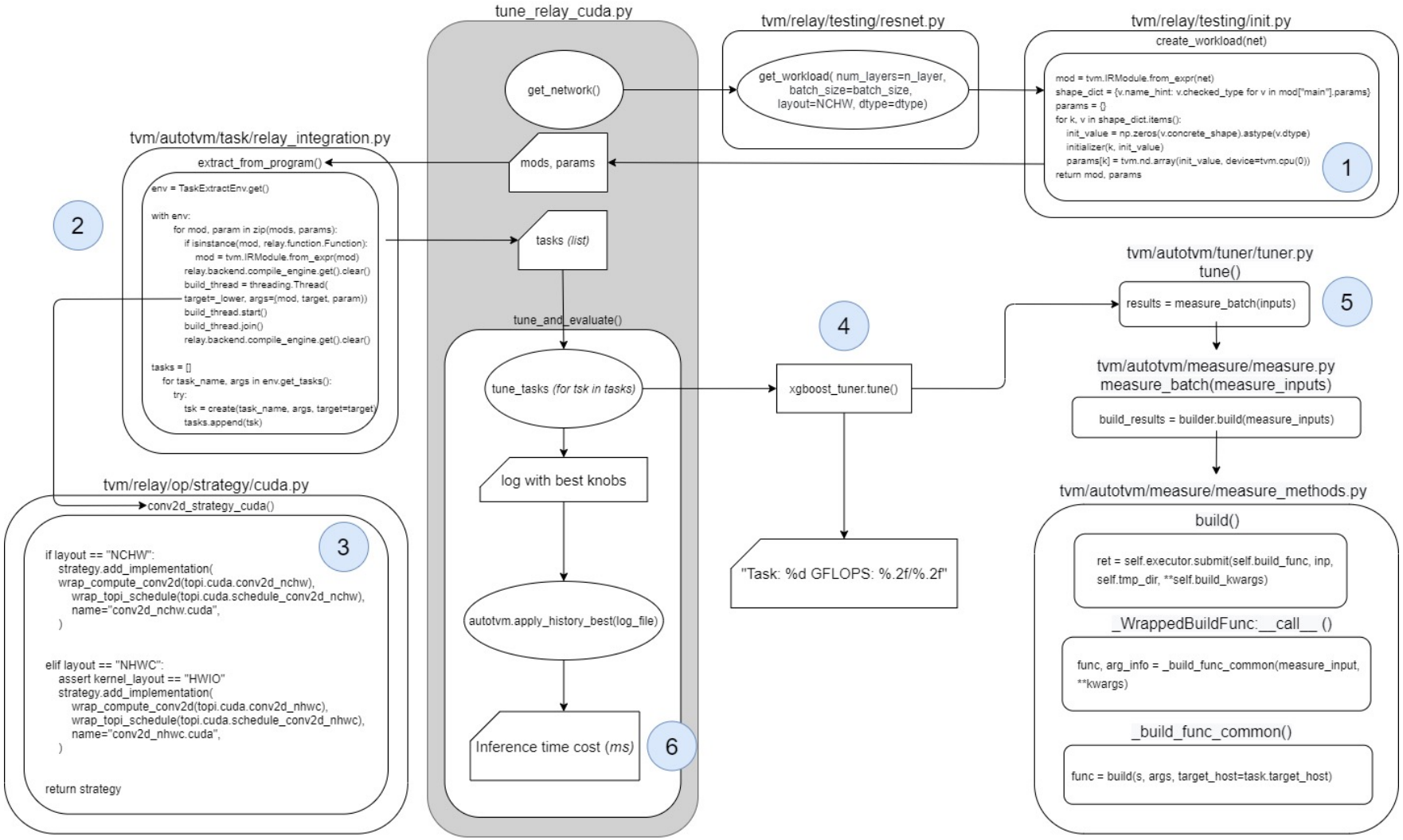
- Enlarging the configuration space
  - Adding parameter candidates.
  - Introducing new knobs into a design space.
- Attempts on merging the configuration space

Brief Recap



https://sampl.cs.washington.edu/tvmconf/#about-tvmconf

General Workflow of AutoTVM



https://github.com/apache/tvm/tree/main/python/tvm

https://tvm.apache.org/docs/tutorials/autotvm/tune_relay_cuda.html

IR Module

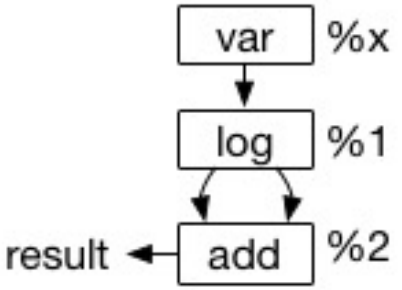
Python Code

```
x = relay.var("x")
v1 = relay.log(x)
v2 = relay.add(v1, v1)
f = relay.Function([x], v2)
```

Text Form

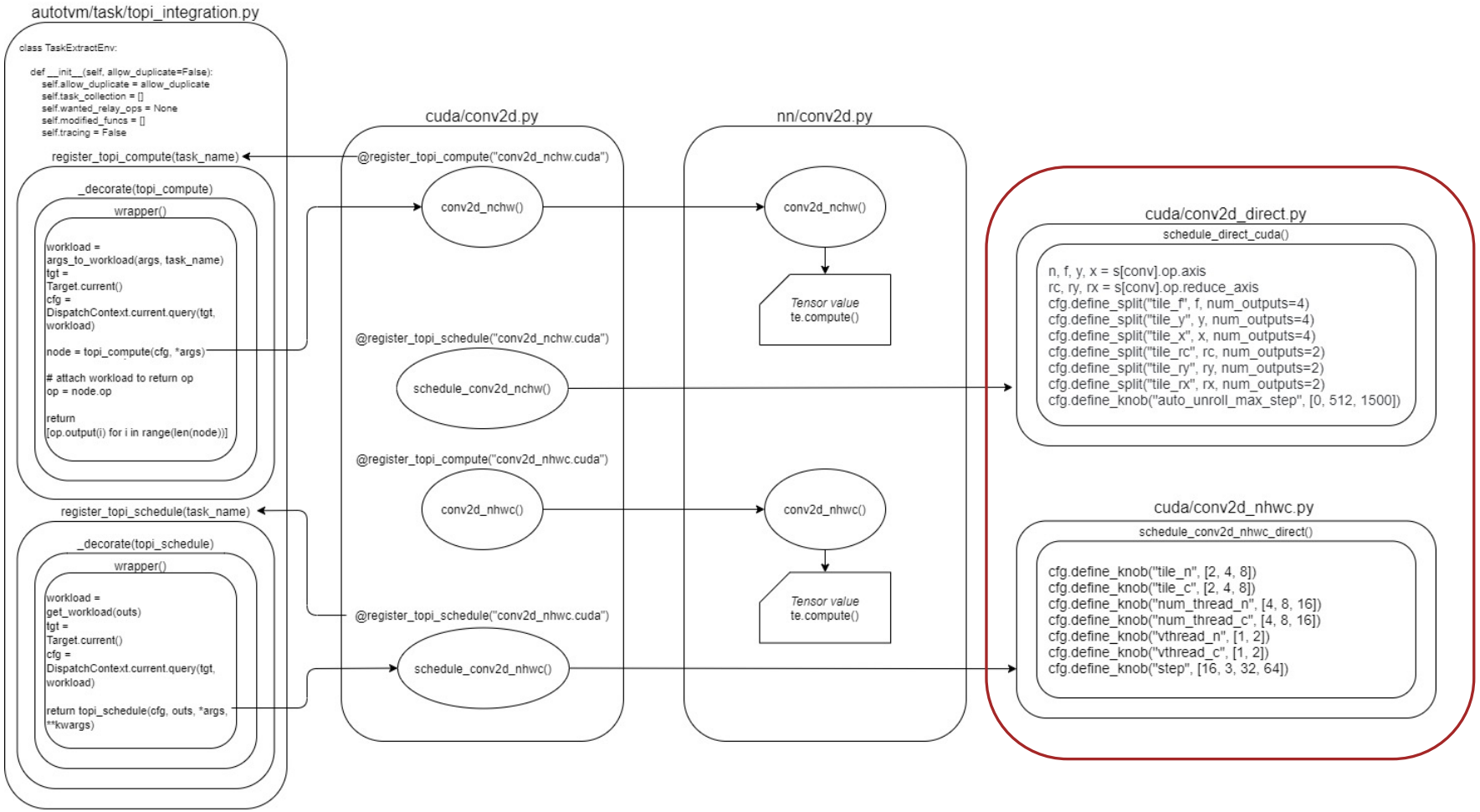
```
fn (%x) {
  %1 = log(%x)
  %2 = add(%1, %1)
  %2
}
```

AST Structure



https://tvm.apache.org/docs/dev/relay_intro.html

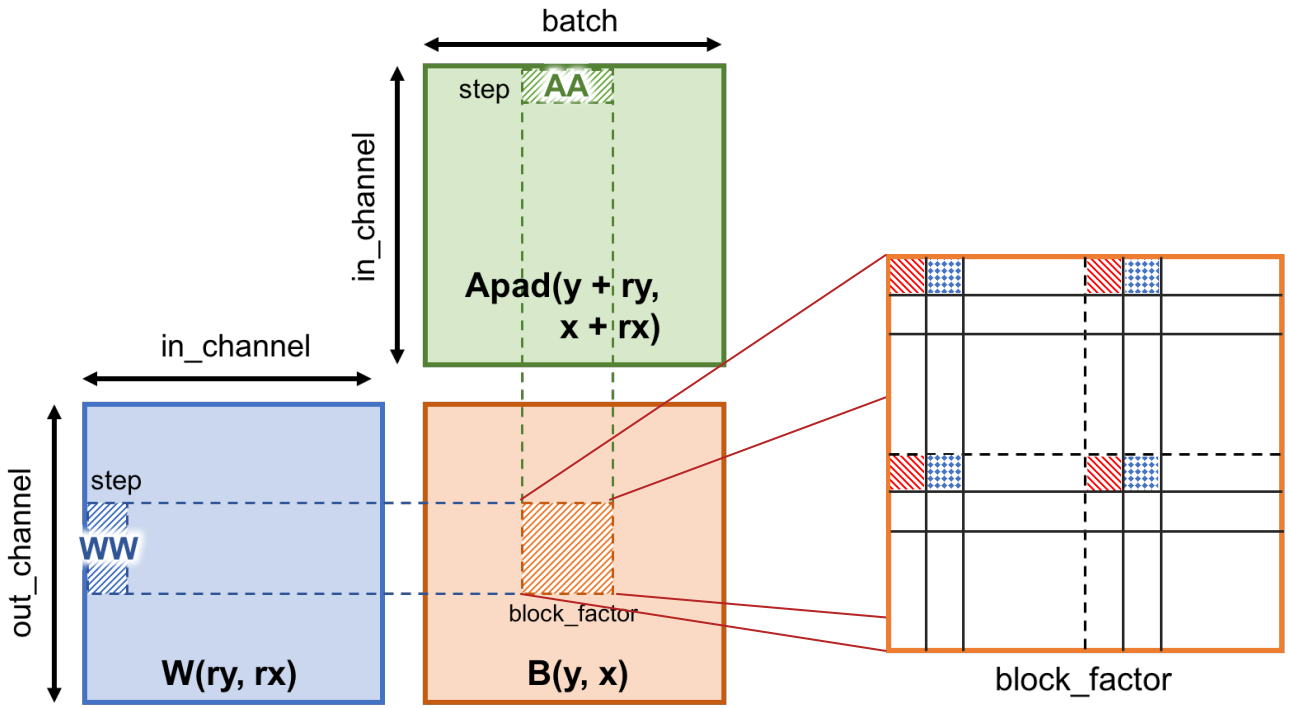
Design Space Templates



https://github.com/apache/tvm/blob/main/python/tvm/topi/cuda/conv2d_direct.py

https://github.com/apache/tvm/blob/main/python/tvm/topi/cuda/conv2d_nhwc.py

Defining Splits vs. Knobs




```
A = t.placeholder((1024, 1024))
B = t.placeholder((1024, 1024))
k = t.reduce_axis((0, 1024))
C = t.compute((1024, 1024), lambda y, x:
                t.sum(A[k, y] * B[k, x], axis=k))
s = t.create_schedule(C.op)
```

```
for y in range(1024):
  for x in range(1024):
    C[y][x] = 0
    for k in range(1024):
      C[y][x] += A[k][y] * B[k][x]
```

+ Loop Tiling

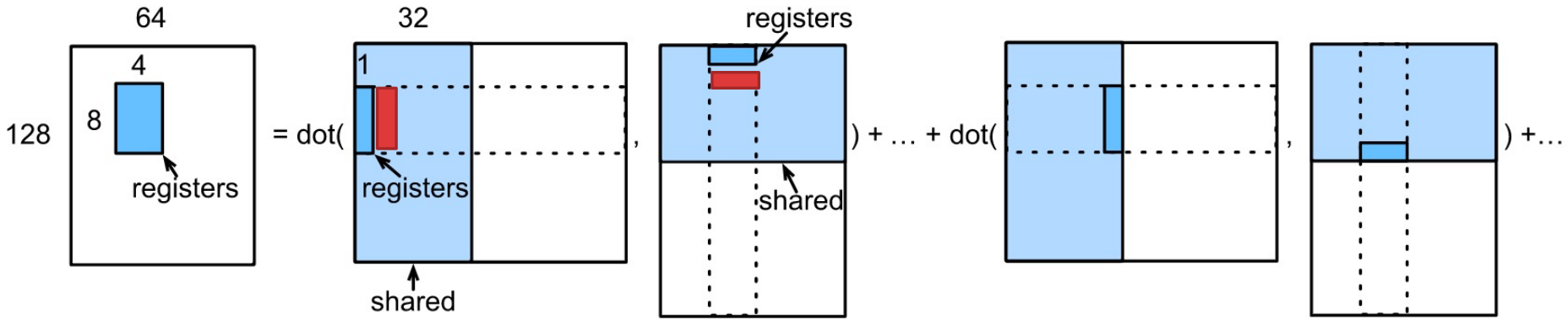
```
yo, xo, ko, yi, xi, ki = s[C].tile(y, x, k, 8, 8, 8)
```

```
for yo in range(128):
  for xo in range(128):
    C[yo*8:yo*8+8][xo*8:xo*8+8] = 0
    for ko in range(128):
      for yi in range(8):
        for xi in range(8):
          for ki in range(8):
            C[yo*8+yi][xo*8+xi] +=
              A[ko*8+ki][yo*8+yi] * B[ko*8+ki][xo*8+xi]
```

https://tvm.apache.org/docs/tutorials/optimize/opt_conv_cuda.html

https://www.programmersought.com/article/38296967626/

Matrix Computation



https://tvm.d2l.ai/chapter_gpu_schedules/matmul.html

Enlarging the Configuration Space

cuda/conv2d_direct.py

schedule_direct_cuda()

```
n, f, y, x = s[conv].op.axis
rc, ry, rx = s[conv].op.reduce_axis
cfg.define_split("tile_f", f, num_outputs=4)
cfg.define_split("tile_y", y, num_outputs=4)
cfg.define_split("tile_x", x, num_outputs=4)
cfg.define_split("tile_rc", rc, num_outputs=2)
cfg.define_split("tile_ry", ry, num_outputs=2)
cfg.define_split("tile_rx", rx, num_outputs=2)
cfg.define_knob("auto_unroll_max_step", [0, 512, 1500])
```

cuda/conv2d_direct.py

schedule_direct_cuda()

```
n, f, y, x = s[conv].op.axis
rc, ry, rx = s[conv].op.reduce_axis
cfg.define_split("tile_f", f, num_outputs=4)
cfg.define_split("tile_y", y, num_outputs=4)
cfg.define_split("tile_x", x, num_outputs=4)
cfg.define_split("tile_rc", rc, num_outputs=2)
cfg.define_split("tile_ry", ry, num_outputs=2)
cfg.define_split("tile_rx", rx, num_outputs=2)
cfg.define_knob("auto_unroll_max_step", [0, 512, 1024, 1500])
```

cuda/conv2d_nhwc.py

schedule_conv2d_nhwc_direct()

```
cfg.define_knob("tile_n", [2, 4, 8])
cfg.define_knob("tile_c", [2, 4, 8])
cfg.define_knob("num_thread_n", [4, 8, 16])
cfg.define_knob("num_thread_c", [4, 8, 16])
cfg.define_knob("vthread_n", [1, 2])
cfg.define_knob("vthread_c", [1, 2])
cfg.define_knob("step", [16, 3, 32, 64])
```

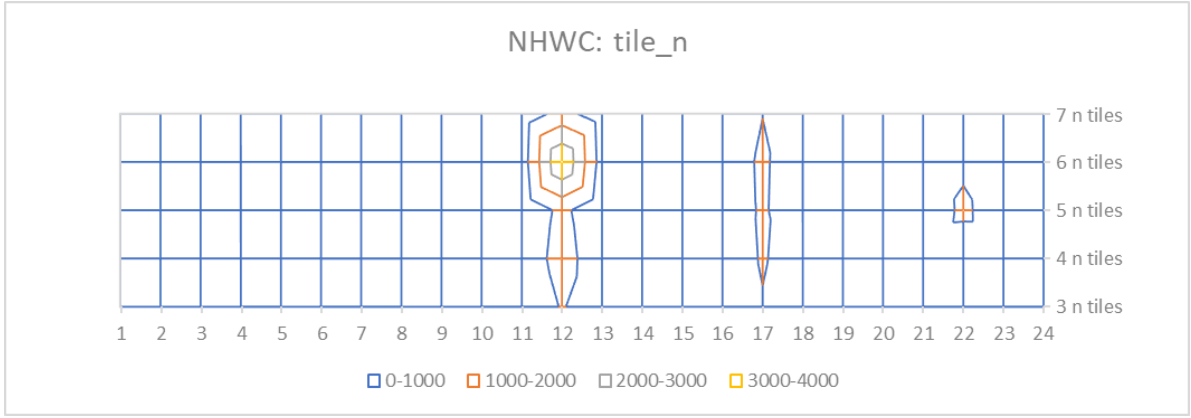
cuda/conv2d_nhwc.py

schedule_conv2d_nhwc_direct()

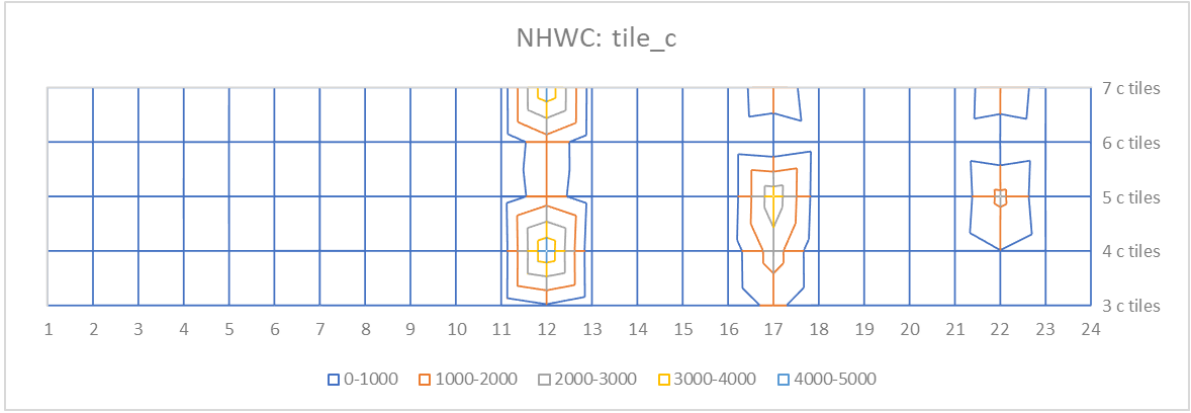
```
cfg.define_knob("tile_n", [2, 4, 8, 16, 32, 64, 128])
cfg.define_knob("tile_c", [2, 4, 8, 16, 32, 64, 128])
cfg.define_knob("num_thread_n", [4, 8, 16])
cfg.define_knob("num_thread_c", [4, 8, 16])
cfg.define_knob("vthread_n", [1, 2])
cfg.define_knob("vthread_c", [1, 2])
cfg.define_knob("step", [16, 3, 32, 64])
```

Enlarging the Configuration Space: Results

NHWC: tile_n

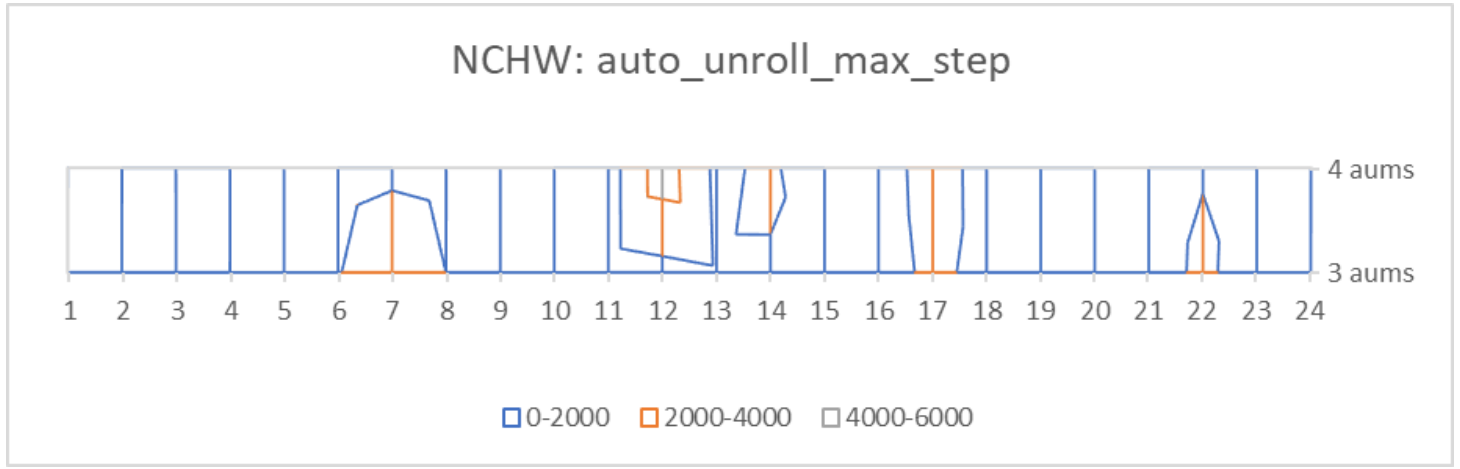


NHWC: tile_c

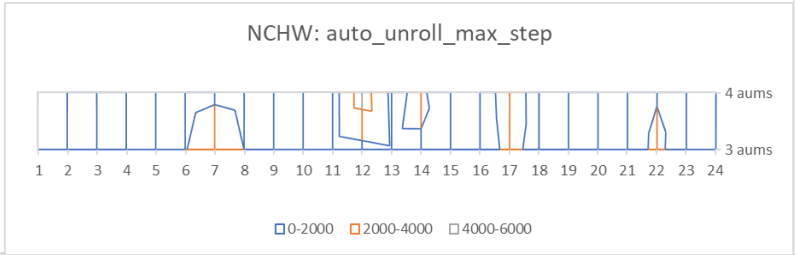


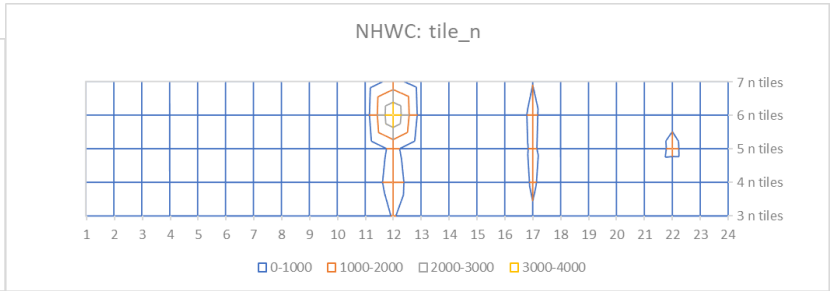
Enlarging the Configuration Space: Results

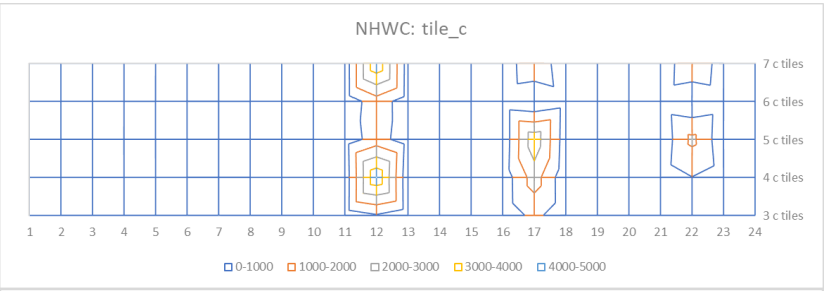
NCHW: auto_unroll_max_step

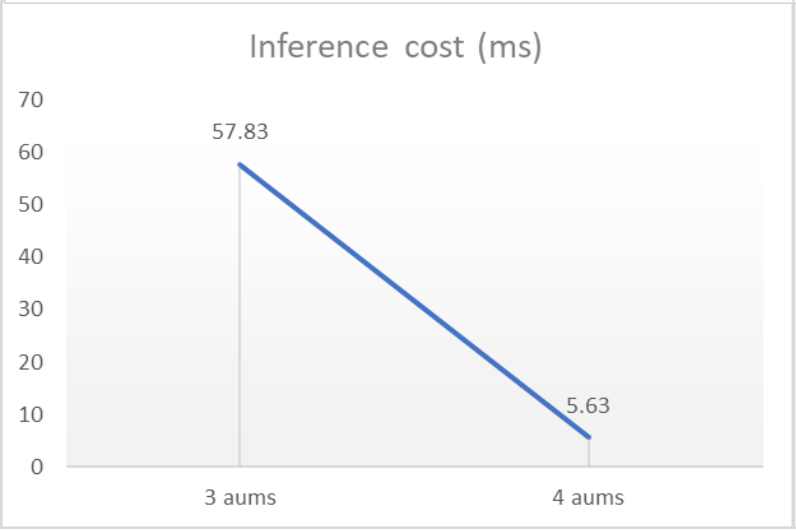


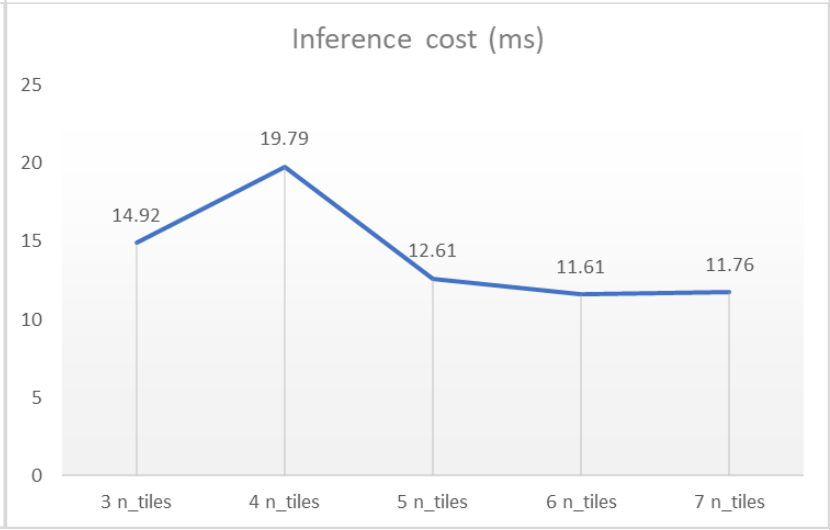
Enlarging the Configuration Space: Results

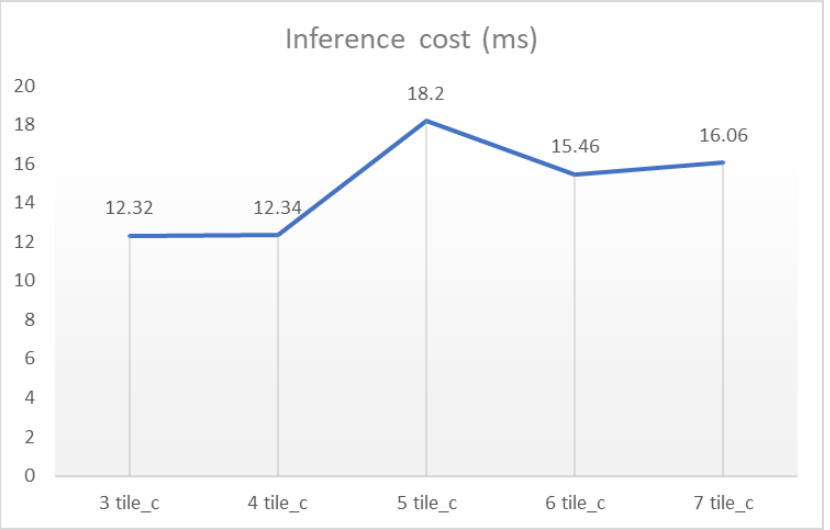




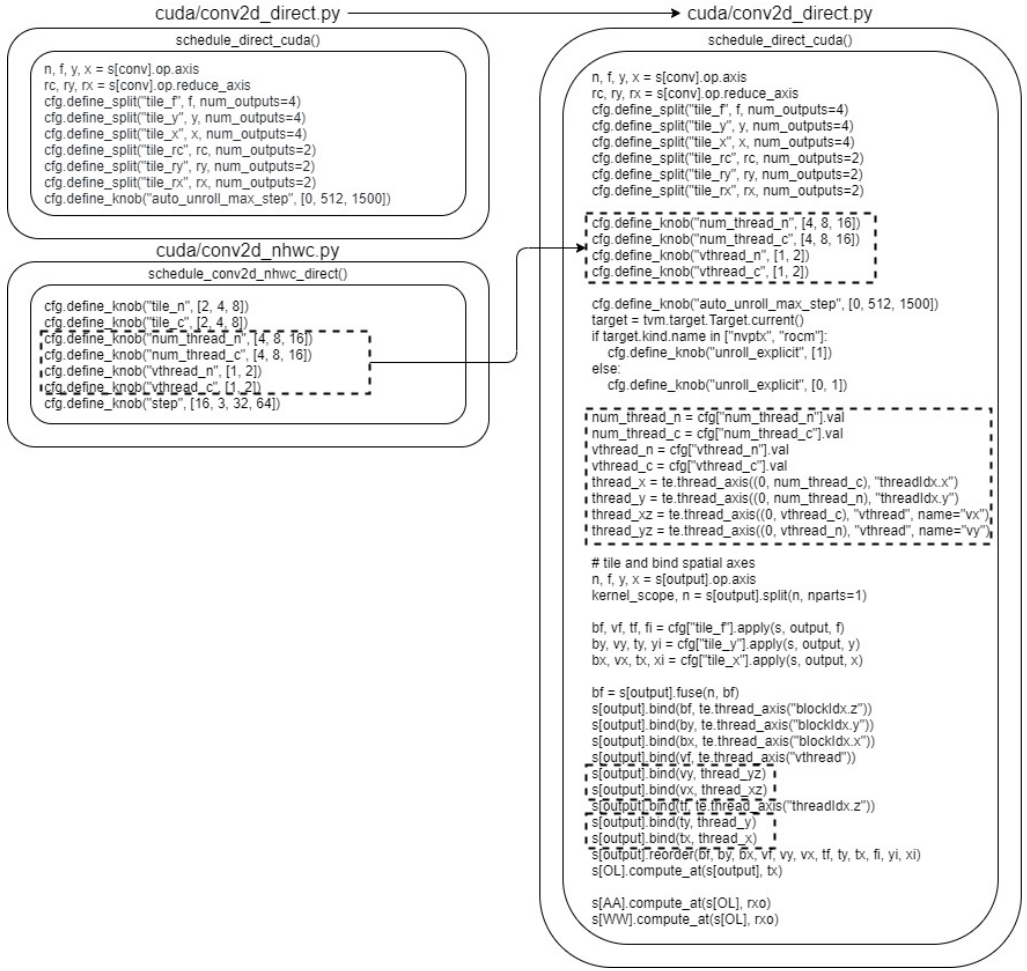




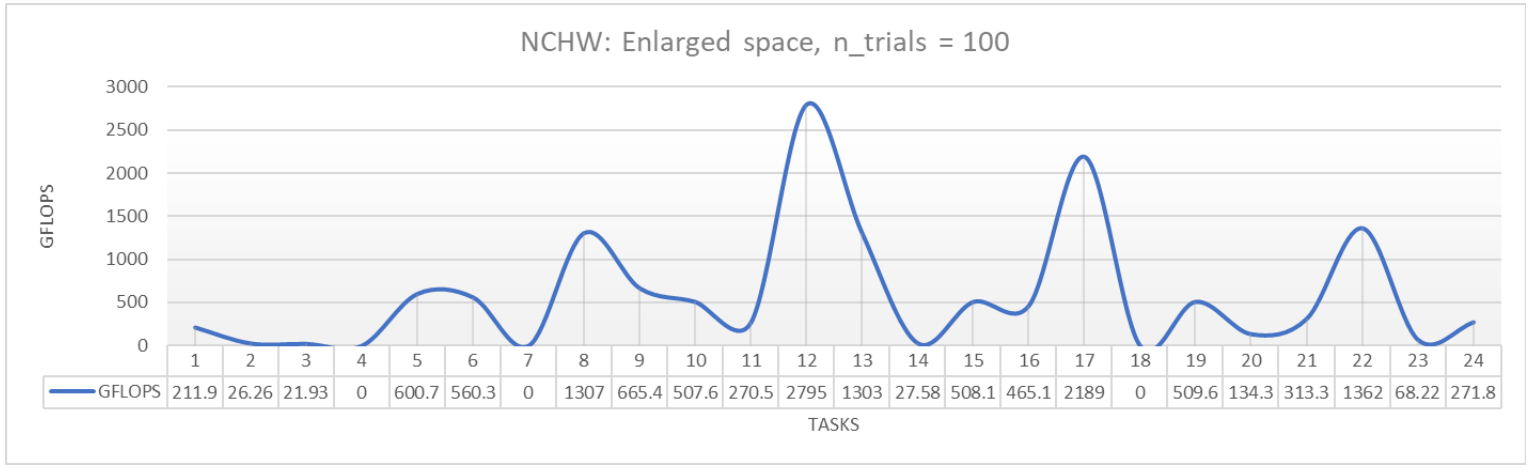




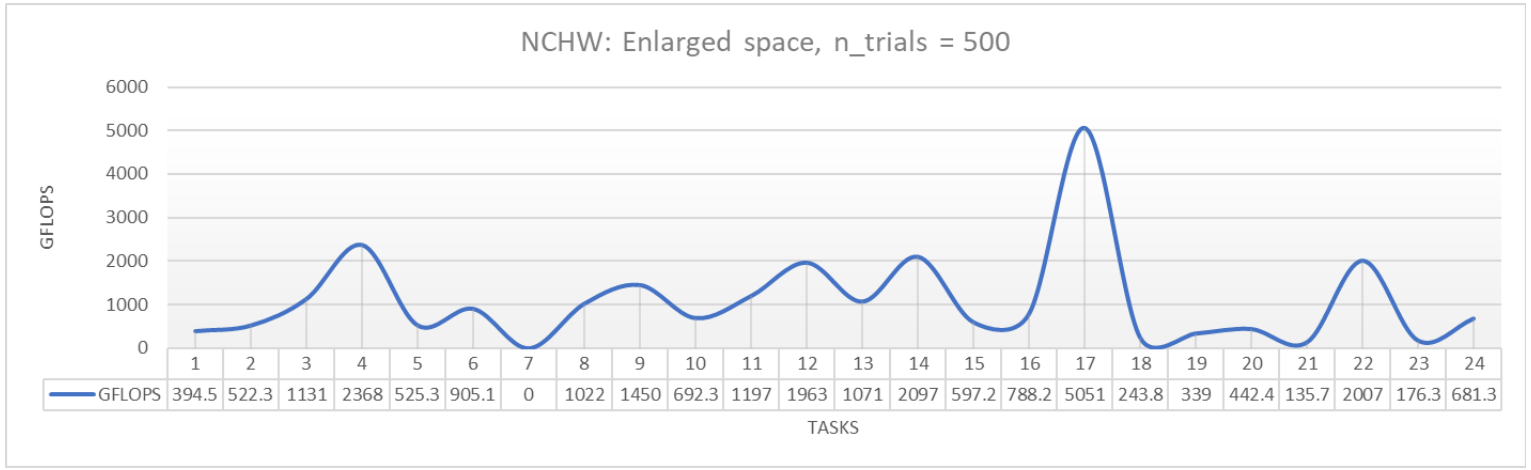
Enlarging the Configuration Space II



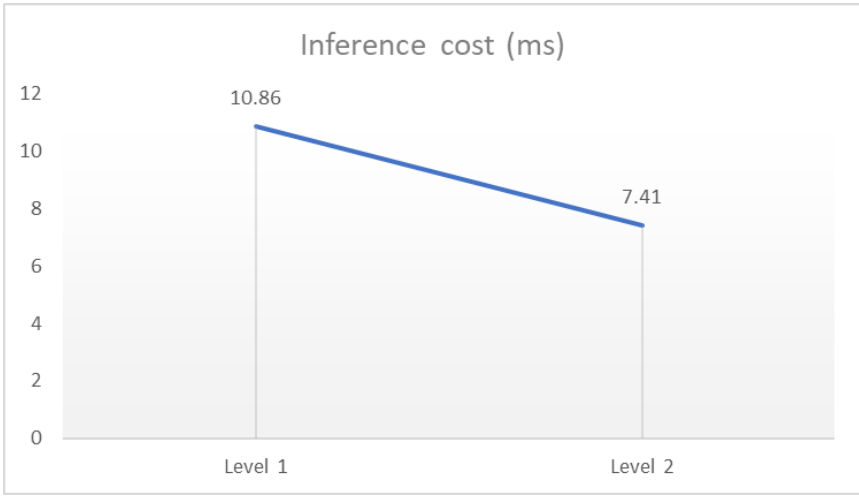
Enlarging the Configuration Space II: Results



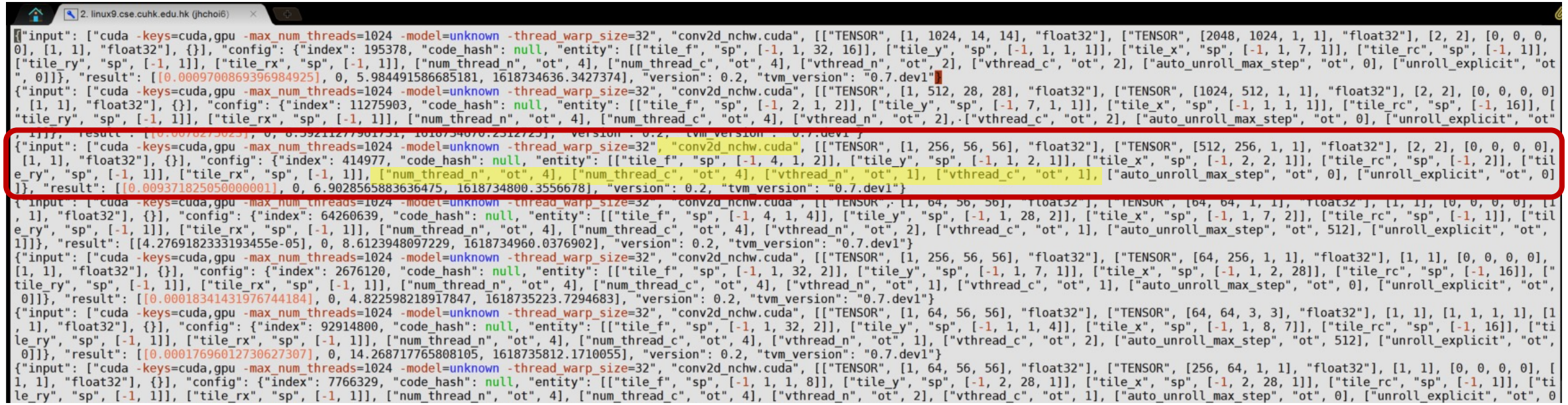
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
GFLOPS	211.9	26.26	21.93	0	600.7	560.3	0	1307	665.4	507.6	270.5	2795	1303	27.58	508.1	465.1	2189	0	509.6	134.3	313.3	1362	68.22	271.8



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
GFLOPS	394.5	522.3	1131	2368	525.3	905.1	0	1022	1450	692.3	1197	1963	1071	2097	597.2	788.2	5051	243.8	339	442.4	135.7	2007	176.3	681.3



Enlarging the Configuration Space II: Results



Attempts to Merge the Configuration Space

- Creating two IR modules from ResNet models of NCHW and NHWC layouts.
  - Formatting the log file.
- Including NHWC tasks and schedule when tuning a NCHW module.

Attempt 1: Creating two IR modules

tvm/relay/testing/resnet.py

```
get_workload( num_layers=n_layer,
batch_size=batch_size,
layout=NCHW, dtype=dtype)
```

tvm/relay/testing/init.py

```
create_workload(net)

mod = tvm.IRModule.from_expr(net)
shape_dict = {v.name_hint: v.checked_type for v in mod["main"].params}
params = {}
for k, v in shape_dict.items():
    init_value = np.zeros(v.concrete_shape).astype(v.dtype)
    initializer(k, init_value)
    params[k] = tvm.nd.array(init_value, device=tvm.cpu(0))

mod2 = tvm.IRModule.from_expr(net2)
shape_dict2 = {v.name_hint: v.checked_type for v in mod2["main"].params}
params2 = {}
for k, v in shape_dict2.items():
    init_value = np.zeros(v.concrete_shape).astype(v.dtype)
    initializer(k, init_value)
    params2[k] = tvm.nd.array(init_value, device=tvm.cpu(0))

return mod, params, mod2, params2
```

1

2

tvm/autotvm/task/relay_integration.py

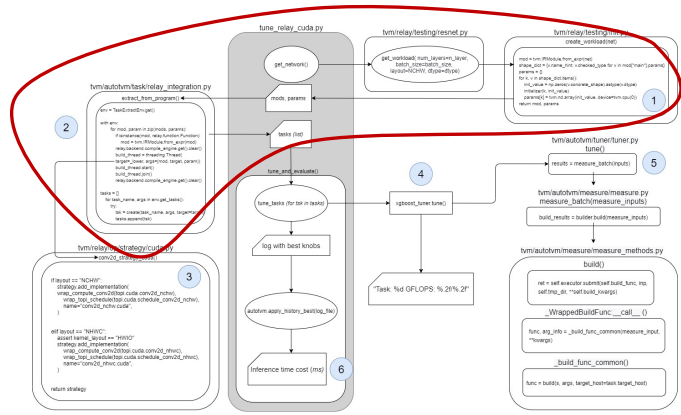
```
extract_from_program()

env = TaskExtractEnv.get()

with env:
    for mod, param in zip(mods, params):
        if isinstance(mod, relay.function.Function):
            mod = tvm.IRModule.from_expr(mod)
        relay.backend.compile_engine.get().clear()
        build_thread = threading.Thread(
        target=_lower, args=(mod, target, param))
        build_thread.start()
        build_thread.join()
        relay.backend.compile_engine.get().clear()

    for mod, param in zip(mods2, params2):
        if isinstance(mod, relay.function.Function):
            mod = tvm.IRModule.from_expr(mod)
        relay.backend.compile_engine.get().clear()
        build_thread = threading.Thread(
        target=_lower, args=(mod, target, param))
        build_thread.start()
        build_thread.join()
        relay.backend.compile_engine.get().clear()

tasks = []
    for task_name, args in env.get_tasks():
        try:
            tsk = create(task_name, args, target=target)
            tasks.append(tsk)
```



Attempt 1: Formatting the log file

```python
import json

res = []
with open('resnet-50_NCHW.log', 'r') as f:
    lines = f.readlines()
    for line in lines:
        dic = json.loads(line)
        keylist = dic.get("input")
        if keylist[1] in ["conv2d_nhwc.cuda", "conv2d_nhwc_winograd_direct.cuda"]:
            if keylist[1] == "conv2d_nhwc.cuda":
                keylist[1] = "conv2d_nchw.cuda"
            elif keylist[1] == "conv2d_nhwc_winograd_direct.cuda":
                keylist[1] = "conv2d_nchw_winograd.cuda"

            a_list = keylist[2][0][1] #target tensor to rearrange
            order = [0, 3, 1, 2]  #nhwc to nchw order
            a_list = [a_list[i] for i in order]
            keylist[2][0][1] = a_list

            keylist[2][1][1] = keylist[2][1][1][::-1] #reversed TENSOR

        dic["input"] = keylist
        res.append(dic)
f.close()

with open("resnet-50_NCHWnew.log", "a") as w:
    for item in res:
        w.write(json.dumps(item))
        w.write('₩n')

w.close()
```

Attempt 2: Binding two types of tasks

tvm/relay/op/strategy/cuda.py

conv2d_strategy_cuda()

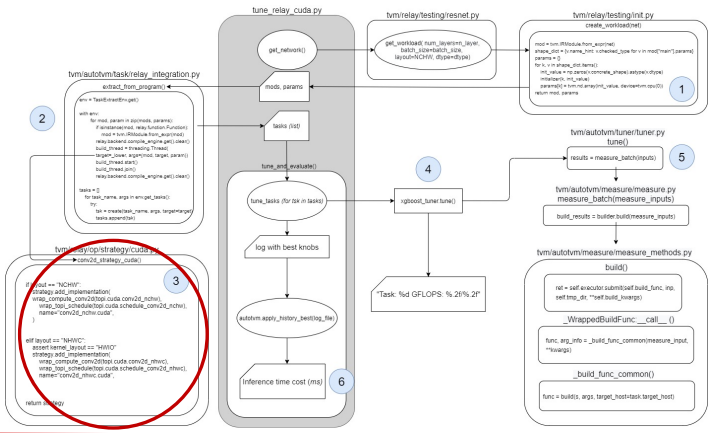
3

```
if layout == "NCHW":
    strategy.add_implementation(
    wrap_compute_conv2d(topi.cuda.conv2d_nchw),
        wrap_topi_schedule(topi.cuda.schedule_conv2d_nchw),
        name="conv2d_nchw.cuda",
    )

    strategy.add_implementation(
    wrap_compute_conv2d(topi.cuda.conv2d_nchw),
        wrap_topi_schedule(topi.cuda.schedule_conv2d_nhwc),
        name="conv2d_nhwc.cuda",
    )

elif layout == "NHWC":
    assert kernel_layout == "HWIO"
    strategy.add_implementation(
        wrap_compute_conv2d(topi.cuda.conv2d_nhwc),
        wrap_topi_schedule(topi.cuda.schedule_conv2d_nhwc),
        name="conv2d_nhwc.cuda",
    )

return strategy
```



If there was a next attempt…

- Differentiate between “conv2d_nchw.cuda” and “conv2d_nhwc.cuda” tasks.
- Devise a method to format tensors and other parameters according to the tasks and schedule of the other memory layout.

```
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw_winograd.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw_winograd.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw_winograd.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw_winograd.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
task name conv2d_nchw.cuda
```

```
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc_winograd_direct.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc_winograd_direct.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc_winograd_direct.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc_winograd_direct.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
task name conv2d_nhwc.cuda
```

```
kernel shape:  2048 512 1 1
kernel shape:  512 512 3 3
kernel shape:  512 2048 1 1
kernel shape:  2048 512 1 1
kernel shape:  512 1024 1 1
kernel shape:  1024 256 1 1
kernel shape:  256 256 3 3
kernel shape:  256 1024 1 1
kernel shape:  1024 256 1 1
kernel shape:  256 512 1 1
kernel shape:  512 128 1 1
kernel shape:  128 128 3 3
kernel shape:  128 512 1 1
kernel shape:  512 128 1 1
kernel shape:  128 256 1 1
kernel shape:  256 64 1 1
kernel shape:  64 64 3 3
kernel shape:  64 256 1 1
kernel shape:  256 64 1 1
kernel shape:  64 64 1 1
kernel shape:  64 3 7 7
kernel shape:  256 64 1 1
kernel shape:  512 256 1 1
kernel shape:  1024 512 1 1
kernel shape:  2048 1024 1 1
```

```
kernel shape:  1 1 512 2048
kernel shape:  3 3 512 512
kernel shape:  1 1 2048 512
kernel shape:  1 1 512 2048
kernel shape:  1 1 1024 512
kernel shape:  1 1 256 1024
kernel shape:  3 3 256 256
kernel shape:  1 1 1024 256
kernel shape:  1 1 256 1024
kernel shape:  1 1 512 256
kernel shape:  1 1 128 512
kernel shape:  3 3 128 128
kernel shape:  1 1 512 128
kernel shape:  1 1 128 512
kernel shape:  1 1 256 128
kernel shape:  1 1 64 256
kernel shape:  3 3 64 64
kernel shape:  1 1 256 64
kernel shape:  1 1 64 256
kernel shape:  1 1 64 64
kernel shape:  7 7 3 64
kernel shape:  1 1 64 256
kernel shape:  1 1 256 512
kernel shape:  1 1 512 1024
kernel shape:  1 1 1024 2048
```

Conclusion

- Enlarged the configuration space
  - Expands the exploration area of the search algorithm
  - A more comprehensive log file can be formed
  - Low inference costs and a higher peak GFLOPS could be achieved.
- Merging the design spaces requires more work
  - Cross-compatibility of the tensor data

Auto-scheduler: Ansor

- AutoTVM = template-based search (manual)
- Autoscheduler = automatic space construction
  - Less search time + higher performances.

	AutoTVM Workflow	Auto-scheduler Workflow

Step 1: Write a compute definition
 	
(relatively easy part)
	`# Matrix multiply`
	
	`C = te.compute((M, N), lambda x, y:`
	`  te.sum(A[x, k] * B[k, y], axis=k))`
		`# The same`

Step 2: Write a schedule template
 	
(difficult part)
	`# 20-100 lines of tricky DSL code`
	
	`# Define search space`
	`cfg.define_split("tile_x", batch, num_outputs=4)`
	`cfg.define_split("tile_y", out_dim, num_outputs=4)`
	`…`
	
	`# Apply config into the template`
	`bx, txz, tx, xi = cfg["tile_x"].apply(s, C, C.op.axis[0])`
	`by, tyz, ty, yi = cfg["tile_y"].apply(s, C, C.op.axis[1])`
	`s[C].reorder(by, bx, tyz, txz, ty, tx, yi, xi)`
	`s[CC].compute_at(s[C], tx)`
	`…`
		`# Not required`

Step 3: Run auto-tuning (automatic search)	`tuner.tune(…)`	`task.tune(…)`

https://tvm.apache.org/2021/03/03/intro-auto-scheduler

Thank you!

References

1. Chen, T., Moreau, T., Jiang, Z., Zheng, L., & Yan, E. (2018). TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation* (pp. 579-594). Carlsbad: USENIX Association. Retrieved from https://www.usenix.org/system/files/osdi18-chen.pdf
2. Common architectures in convolutional neural networks. (2020). Retrieved 26 November 2020, from https://www.jeremyjordan.me/convnet-architectures/#resnet
3. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Microsoft*, 1-12. Retrieved from https://arxiv.org/pdf/1512.03385.pdf\
4. NVIDIA TITAN Xp Graphics Card with Pascal Architecture. (2020). Retrieved 25 November 2020, from https://www.nvidia.com/en-us/titan/titan-xp/\
5. ResNet-50 convolutional neural network - MATLAB resnet50. (2020). Retrieved 24 November 2020, from https://www.mathworks.com/help/deeplearning/ref/resnet50.html
6. Zheng, L., & Yan, E. (n.d.). Auto-tuning a convolutional network for NVIDIA GPU. Retrieved September 18, 2020, from https://tvm.apache.org/docs/tutorials/autotvm/tune_relay_cuda.html
7. https://github.com/apache/tvm/tree/main/python/tvm
8. https://tvm.apache.org/2021/03/03/intro-auto-scheduler