

# AI编译器-后端优化

# Auto Tuning



# ZOMI

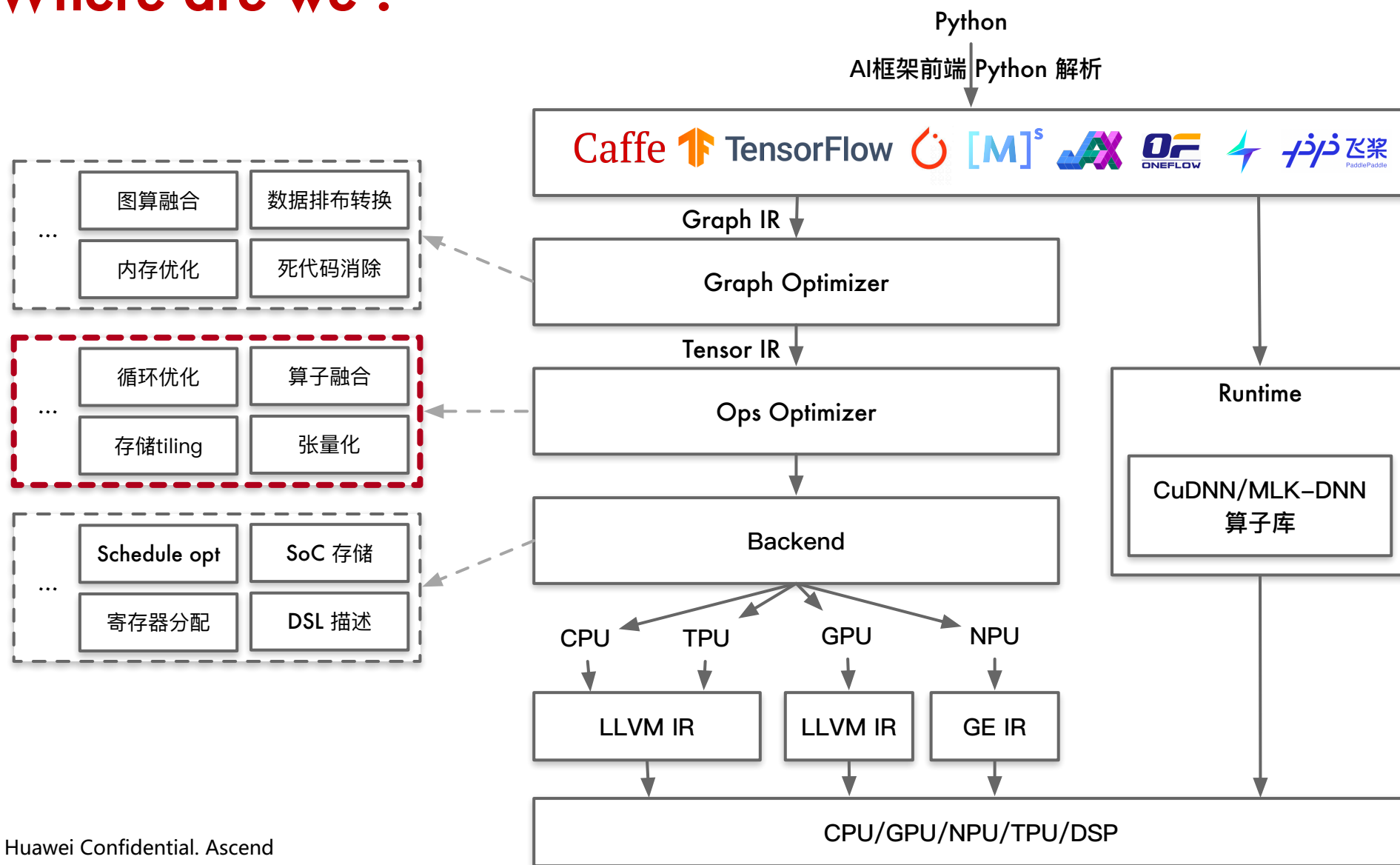


# Talk Overview

## I. AI 编译器后端优化

- 后端优化概念
- 算子计算与调度
- 算子调度优化
- **Auto-Tuning**
- Polyhedral

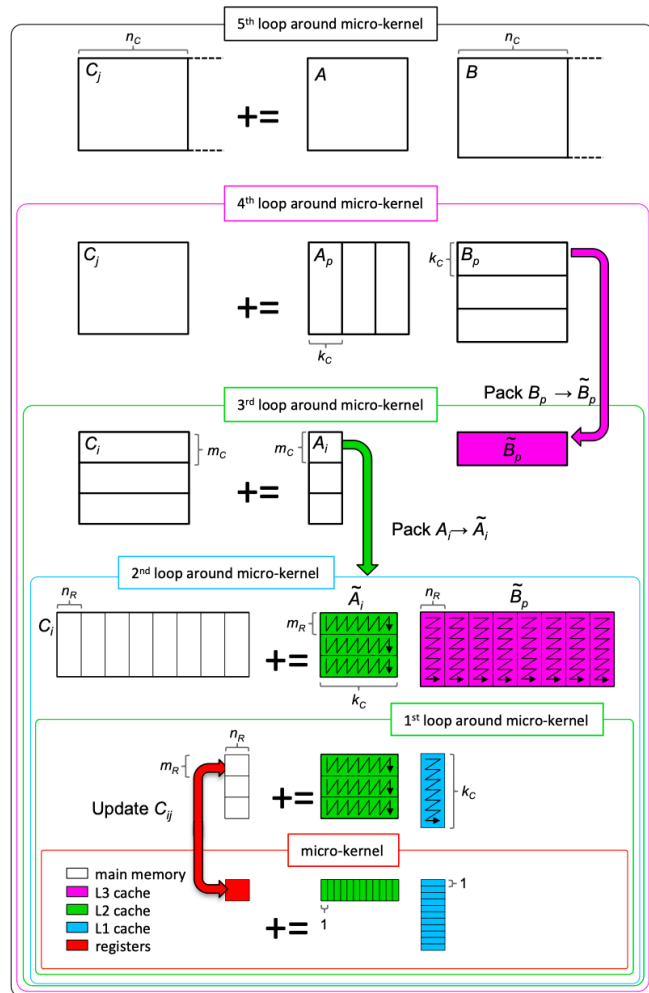
# Where are we ?



# Auto Tuning

- 对于给定的程序和目标架构，找到最优的编译优化方法。使用哪些优化方法？选择什么参数集？用什么顺序应用优化方法以达到最佳性能？

# BLISlab: A Sandbox for Optimizing GEMM



```

Loop 5  for  $j_c = 0 : n - 1$  steps of  $n_c$ 
         $\mathcal{J}_c = j_c : j_c + n_c - 1$ 
Loop 4  for  $p_c = 0 : k - 1$  steps of  $k_c$ 
         $\mathcal{P}_c = p_c : p_c + k_c - 1$ 
         $B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow B_c$  // Pack into  $B_c$ 
Loop 3  for  $i_c = 0 : m - 1$  steps of  $m_c$ 
         $\mathcal{I}_c = i_c : i_c + m_c - 1$ 
         $A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow A_c$  // Pack into  $A_c$ 


---


// Macro-kernel
Loop 2  for  $j_r = 0 : n_c - 1$  steps of  $n_r$ 
         $\mathcal{J}_r = j_r : j_r + n_r - 1$ 
Loop 1  for  $i_r = 0 : m_c - 1$  steps of  $m_r$ 
         $\mathcal{I}_r = i_r : i_r + m_r - 1$ 


---


// Micro-kernel
Loop 0  for  $k_r = 0 : k_c - 1$ 
         $C_c(\mathcal{I}_r, \mathcal{J}_r)$ 
         $+= A_c(\mathcal{I}_r, k_r) B_c(k_r, \mathcal{J}_r)$ 


---


        endfor
    endfor
endfor
endfor
endfor
endfor
    
```

# Auto Tuning

- 优化选择 Optimize Selection
- 优化顺序 Optimize Sequence

# Auto Tuning in AI

- **更低实验开销**：1) 聚焦算子或者 kernel 级别的优化，而非整个程序。2) Cost Model 在CPU上模拟NPU执行，训练和推理推理的模拟速度要求足够快。
- **特定领域结构**：针对神经网络算子或者 kernel 级别的优化，1) 主要是高度循环化、张量化、并行化特点进行优化；2) 大量相类似的算子计算模式。

## Question?

- 如何可以让机器匹配手写优化性能？
  1. 建立一个足够大的搜索空间，保证可能的人工手写优化全部包含在这个搜索空间里面
  2. 快速地搜索这个这个空间，获取优化的实现





# Auto Tuning in AI

1. 参数化 Parameterization
2. 成本模型 Cost Model
3. 搜索算法 Search Algorithm

```
1 for (int n = 0; n < o_n; ++n) {
2     for (int c = 0; c < o_c; ++c) {
3         for (int j = 0; j < o_h; ++j) {
4             for (int i = 0; i < o_w; ++i) {
5                 int d_start = n * i_c * i_h * i_w + j * i_w + i;
6                 int temp = 0;
7                 for (int kk = 0; kk < k_c; ++kk) {
8                     for (int kj = 0; kj < k_h; ++kj) {
9                         for (int ki = 0; ki < k_w; ++ki) {
10                            int k_idx = kk * k_h * k_w + kj * k_w + ki;
11                            int d_idx = d_start + kk * i_h * i_w + kj * i_w + ki;
12                            temp += inputs->data[d_idx] * kernel->data[k_idx];
13                        }
14                    }
15                }
16                res[n * o_c * o_h * o_w + j * o_w + i] = temp;
17            }
18        }
19    }
20 }
```

# Auto Tuning in AI

1. 参数化 Parameterization
2. 成本模型 Cost Model
3. 搜索算法 Search Algorithm

- 参数化 Parameterization：对调度优化问题进行建模，参数化优化空间一般由可参数化变换（Loop）的可能参数取值组合构成，因此需要调度原语进行参数化表示。Halide 将算法和调度解耦，TVM 提供调度模板。

```
1
2  bx, tx = s[C].split(C.op.axis[0], factor=64)
3  s[C].bind(bx, tvm.te.thread_axis("blockIdx.x"))
4  s[C].bind(tx, tvm.te.thread_axis("threadIdx.x"))
5  fadd = tvm.build(s, [A, B, C], target)
6
```

# Auto Tuning in AI

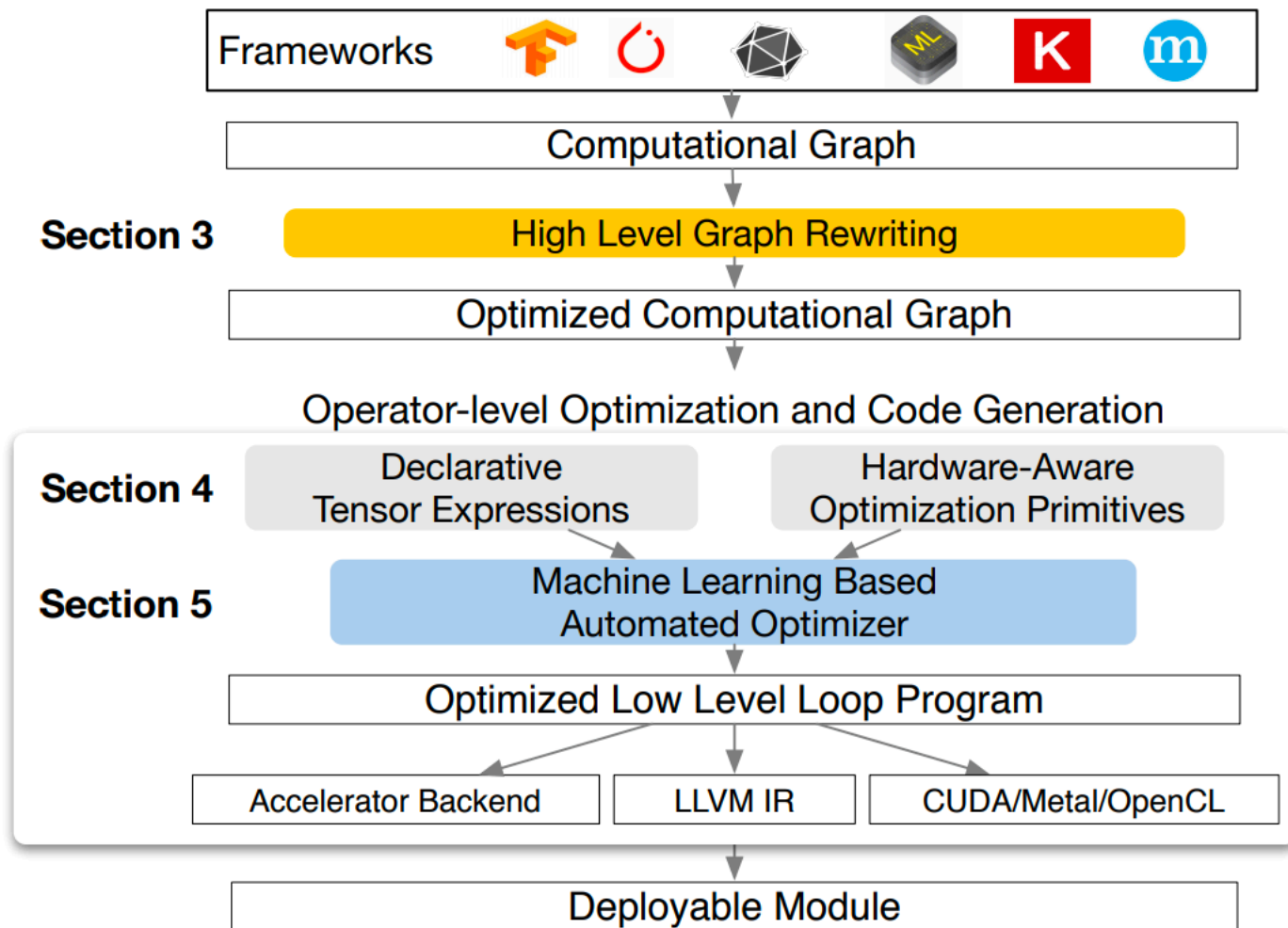
1. 参数化 Parameterization
  2. 成本模型 Cost Model
  3. 搜索算法 Search Algorithm
- 成本模型 Cost Model：用来评价某一参数化下的调度性能，根据对调度额评价来指导最搜索到最优的调度策略。可以从运行时间、内存占用、编译后指令数来评价。实现方式主要有1) 基于NPU硬件的黑盒模型；2) 基于模拟的预定义模型；3) ML-Base 模型，通过机器学习模型来对调度性能进行预测。

# Auto Tuning in AI

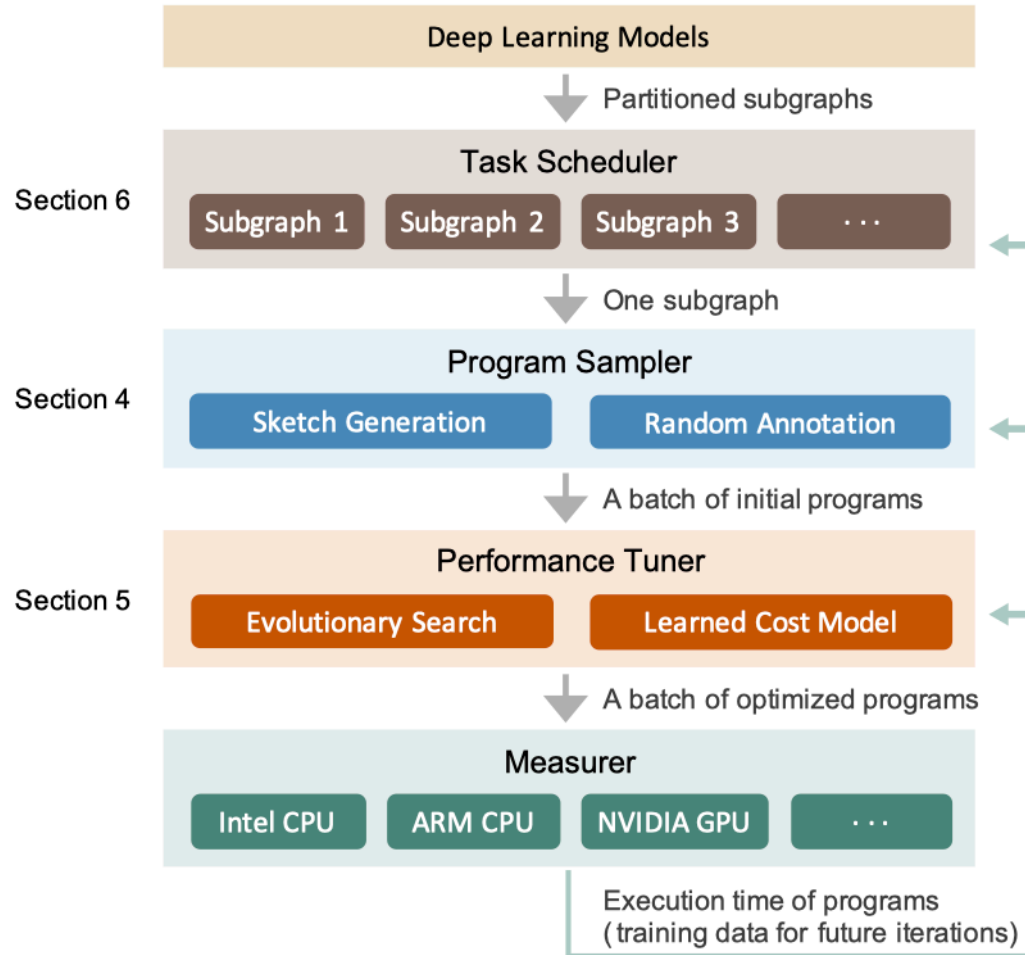
1. 参数化 Parameterization
  2. 成本模型 Cost Model
  3. 搜索算法 Search Algorithm
- 搜索算法 Search Algorithm：确定初始化和搜索空间后，在搜索空间找找到达到性能最优的参数配置。常用的搜索算法有1) 遗传算法、2) 模拟退火算法、3) 强化学习等。

# Auto Tuning in AI

1. 参数化 Parameterization
2. 成本模型 Cost Model
3. 搜索算法 Search Algorithm



# Ansor : Generating High-Performance Tensor Programs for Deep Learning



**Example Input 1:**

**\* The mathematical expression:**  

$$C[i, j] = \sum_k A[i, k] \times B[k, j]$$

$$D[i, j] = \max(C[i, j], 0.0)$$
 where  $0 \leq i, j, k < 512$

**\* The corresponding naive program:**  

```
for i in range(512):
  for j in range(512):
    for k in range(512):
      C[i, j] += A[i, k] * B[k, j]
for i in range(512):
  for j in range(512):
    D[i, j] = max(C[i, j], 0.0)
```

**\* The corresponding DAG:**

```

    graph LR
      A((A)) --> C((C))
      B((B)) --> C
      C --> D((D))
      style A fill:#f96,stroke:#333,stroke-width:1px
      style B fill:#f96,stroke:#333,stroke-width:1px
      style C fill:#f96,stroke:#333,stroke-width:1px
      style D fill:#f96,stroke:#333,stroke-width:1px
        
```

**Generated sketch 1**

```
for i.0 in range(TILE_I0):
  for j.0 in range(TILE_J0):
    for i.1 in range(TILE_I1):
      for j.1 in range(TILE_J1):
        for k.0 in range(TILE_K0):
          for i.2 in range(TILE_I2):
            for j.2 in range(TILE_J2):
              for k.1 in range(TILE_I1):
                for i.3 in range(TILE_I3):
                  for j.3 in range(TILE_J3):
                    C[...] += A[...] * B[...]
          for i.4 in range(TILE_I2 * TILE_I3):
            for j.4 in range(TILE_J2 * TILE_J3):
              D[...] = max(C[...], 0.0)
```

**Sampled program 1**

```
parallel i.0@j.0@i.1@j.1 in range(256):
  for k.0 in range(32):
    for i.2 in range(16):
      unroll k.1 in range(16):
        unroll i.3 in range(4):
          vectorize j.3 in range(16):
            C[...] += A[...] * B[...]
    for i.4 in range(64):
      vectorize j.4 in range(16):
        D[...] = max(C[...], 0.0)
```

**Sampled program 2**

```
parallel i.2 in range(16):
  for j.2 in range(128):
    for k.1 in range(512):
      for i.3 in range(32):
        vectorize j.3 in range(4):
          C[...] += A[...] * B[...]
parallel i.4 in range(512):
  for j.4 in range(512):
    D[...] = max(C[...], 0.0)
```

**Generated sketch 2**

```
for i in range(8):
  for k in range(512):
    C[i, j] = max(A[i, k], 0.0) if k < 400 else 0
for i.0 in range(TILE_I0):
  for j.0 in range(TILE_J0):
    for i.1 in range(TILE_I1):
      for j.1 in range(TILE_J1):
        for k.0 in range(TILE_K0):
          for i.2 in range(TILE_I2):
            for j.2 in range(TILE_J2):
              for k.1 in range(TILE_I1):
                for i.3 in range(TILE_I3):
                  for j.3 in range(TILE_J3):
                    E.cache[...] += C[...] * D[...]
          for i.4 in range(TILE_I2 * TILE_I3):
            for j.4 in range(TILE_J2 * TILE_J3):
              E[...] = E.cache[...]
```

**Sampled program 3**

```
parallel i.0 in range(8):
  for k in range(512):
    C[i, j] = max(A[i, k], 0.0)
      if k < 400 else 0
    for k.0 in range(512):
      vectorize j.3 in range(4):
        E.cache[...] += C[...] * D[...]
    vectorize j.4 in range(4):
      E[...] = E.cache[...]
```

**Example Input 2:**

**\* The mathematical expression:**  

$$B[i, l] = \max(A[i, l], 0.0)$$

$$C[i, k] = \begin{cases} B[i, k], & k < 400 \\ 0, & k \geq 400 \end{cases}$$

$$E[i, j] = \sum_k C[i, k] \times D[k, j]$$
 where  $0 \leq i < 8, 0 \leq j < 4,$   
 $0 \leq k < 512, 0 \leq l < 400$

**\* The corresponding naive program:**  

```
for i in range(8):
  for l in range(400):
    B[i, l] = max(A[i, l], 0.0)
for i in range(8):
  for k in range(512):
    C[i, k] = B[i, k] if k < 400 else 0
for i in range(8):
  for j in range(4):
    for k in range(512):
      E[i, j] += C[i, k] * D[k, j]
```

**\* The corresponding DAG:**

```

    graph LR
      A((A)) --> B((B))
      B --> C((C))
      D((D)) --> E((E))
      C --> E
      style A fill:#f96,stroke:#333,stroke-width:1px
      style B fill:#f96,stroke:#333,stroke-width:1px
      style C fill:#f96,stroke:#333,stroke-width:1px
      style D fill:#f96,stroke:#333,stroke-width:1px
      style E fill:#f96,stroke:#333,stroke-width:1px
        
```

**Generated sketch 3**

```
for i in range(8):
  for k in range(512):
    C[i, k] = max(A[i, k], 0.0) if k < 400 else 0
for i in range(8):
  for j in range(4):
    for k_o in range(TILE_K0):
      for k_i in range(TILE_KI):
        E.rf[...] += C[...] * D[...]
for i in range(8):
  for j in range(4):
    for k_i in range(TILE_KI):
      E[...] += E.rf[...]
```

**Sampled program 4**

```
parallel i in range(8):
  for k in range(512):
    C[i, k] = ...
    for j in range(4):
      unroll k_o in range(32):
        vectorized k_i in range(16):
          E.rf[...] += C[...] * D[...]
parallel i in range(8):
  for j in range(4):
    unroll k_i in range(16):
      E[...] += E.rf[...]
```

# Inference

- Li, Mingzhen, et al. "The deep learning compiler: A comprehensive survey." *IEEE Transactions on Parallel and Distributed Systems* 32.3 (2020): 708-727.
- Ning, Chao, and Fengqi You. "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming." *Computers & Chemical Engineering* 125 (2019): 434-448.
- Xu, Zhiying, et al. "ALT: Breaking the Wall between Graph and Operator Level Optimizations for Deep Learning Compilation." *arXiv preprint arXiv:2210.12415* (2022).
- Baghdadi, Riyadh, et al. "A deep learning based cost model for automatic code optimization." *Proceedings of Machine Learning and Systems* 3 (2021): 181-193.
- Chen, Tianqi, et al. "TVM: end-to-end optimization stack for deep learning." *arXiv preprint arXiv:1802.04799* 11.2018 (2018): 20.
- Purkayastha, Arnab A., et al. "LLVM-based automation of memory decoupling for OpenCL applications on FPGAs." *Microprocessors and Microsystems* 72 (2020): 102909.
- Loop Optimizations: how does the compiler do it? <https://johnysslabs.com/loop-optimizations-how-does-the-compiler-do-it/>
- Loop Optimizations: taking matters into your hands <https://johnysslabs.com/loop-optimizations-taking-matters-into-your-hands/>
- Understanding Latency Hiding on GPUs - UC Berkeley EECS. <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-143.pdf>
- [Reinforcement Learning and Adaptive Sampling for Optimized DNN Compilation, https://arxiv.org/abs/1905.12799](https://arxiv.org/abs/1905.12799)



BUILDING A BETTER CONNECTED WORLD

THANK YOU

Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.