AI编译器-后端优化

算子优化

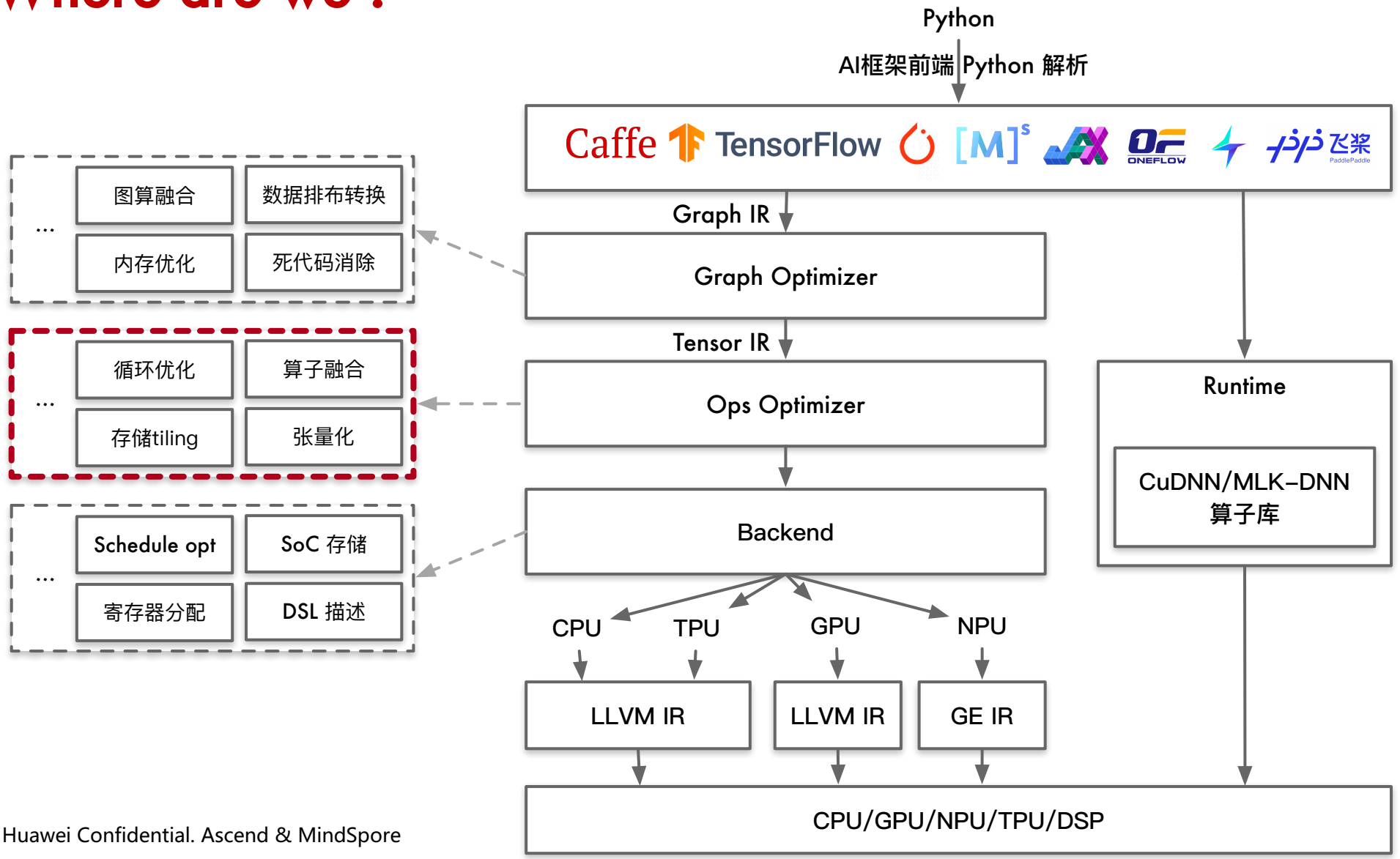
ZOMI

Talk Overview

I. AI 编译器后端优化

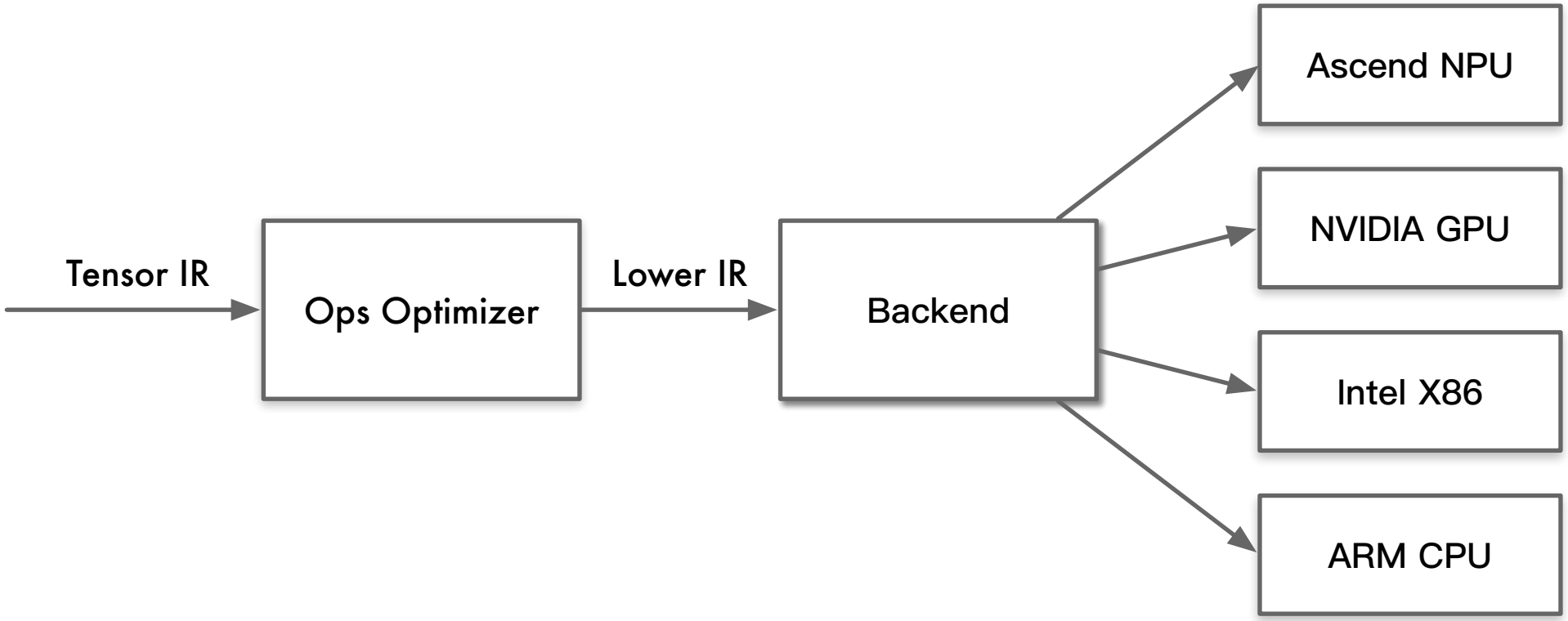
- 后端优化概念
- 算子计算与调度
- 算子调度优化
- Auto-Tuning
- Polyhedral

Where are we ?



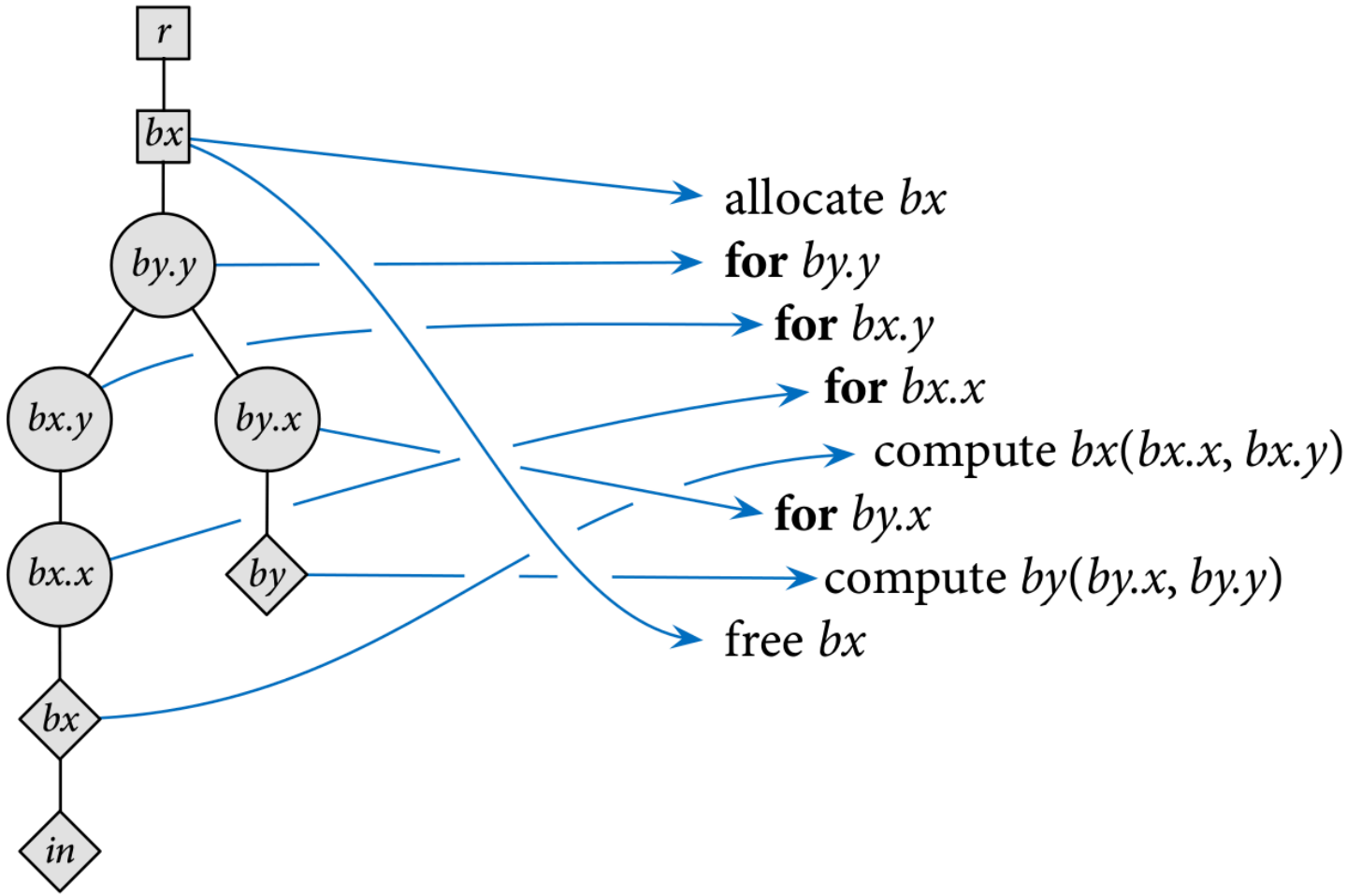
AI 编译器后端部分

1）生成低级IR；2）后端优化；3）代码生成



Schedule trees 调度树

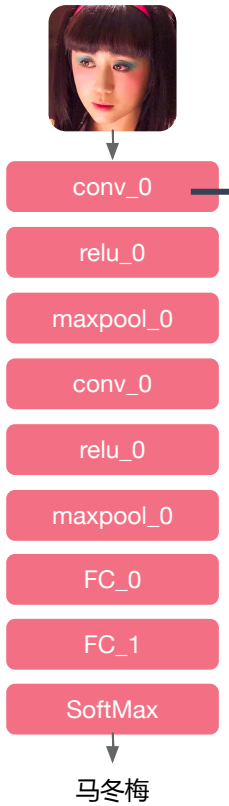
- Root nodes
- Loop nodes
- Storage nodes
- Compute nodes



算子调度优化

计算特征

- 1）多重循环为特点；2）没有复杂控制流；3）以多维张量计算为主要数据；




```
for (int n = 0; n < o_n; ++n) {
        for (int c = 0; c < o_c; ++c) {
            for (int j = 0; j < o_h; ++j) {
                for (int i = 0; i < o_w; ++i) {
                    int d_start = n * i_c * i_h * i_w + j * i_w + i;
                    int temp = 0;
                    for (int kk = 0; kk < k_c; ++kk) {
                        for (int kj = 0; kj < k_h; ++kj) {
                            for (int ki = 0; ki < k_w; ++ki) {
                                int k_idx = kk * k_h * k_w + kj * k_w + ki;
                                int d_idx = d_start + kk * i_h * i_w + kj * i_w + ki;
                                temp += inputs->data[d_idx] * kernel->data[k_idx];
                            }
                        }
                    }
                    res[n * o_c * o_h * o_w + j * o_w + i] = temp;
                }
            }
        }
```

优化方法

Halide

- Reorder(交换)
- Split(拆分)
- Fuse(融合)
- Tile(平铺)
- Vector(向量化)
- 展开(Unrolling)
- 并行(Parallelizing)

TVM

- Reorder(交换)
- Split(拆分)
- Fuse(融合)
- Tile(平铺)
- Vector(向量化)
- Tensor(张量化)
- 展开(Unrolling)
- 并行(Parallelizing)

基于源码的修改

循环交换

- 在多重循环中，采用迭代次数较小的循环驱动内层迭代次数较大的循环能减少内存的消耗，可以加快处理矩阵和图像的代码的速度。

```
1
2   for (int i = 0; i < 10000; i++) {
3       for (int j = 0; j < 200; j++) {
4           C[i] = A[i] + B[j];
5       }
6   }
7
8   # Change To
9
10  for (int i = 0; i <200 ; i++) {
11      for (int j = 0; j < 10000; j++) {
12          C[i] = A[i] + B[j];
13      }
14  }
15
```

循环变量实例化

- 循环体中的实例化变量放在训练外，避免每次都进行对象实例化。

```
for (int i = 0; i < 10000; i++) {
    for (int j = 0; j < 200; j++) {
        C[i] = A[i] + B[j];
    }
}

# Change To

int i = 0;
int j = 0;
for (i = 0; i < 10000; i++) {
    for (j = 0; j < 200; j++) {
        C[i] = A[i] + B[j];
    }
}
```

表达式外放

- 循环体内重复计算的表达式放在训练体外面，避免每次都进行计算，以空间换时间。

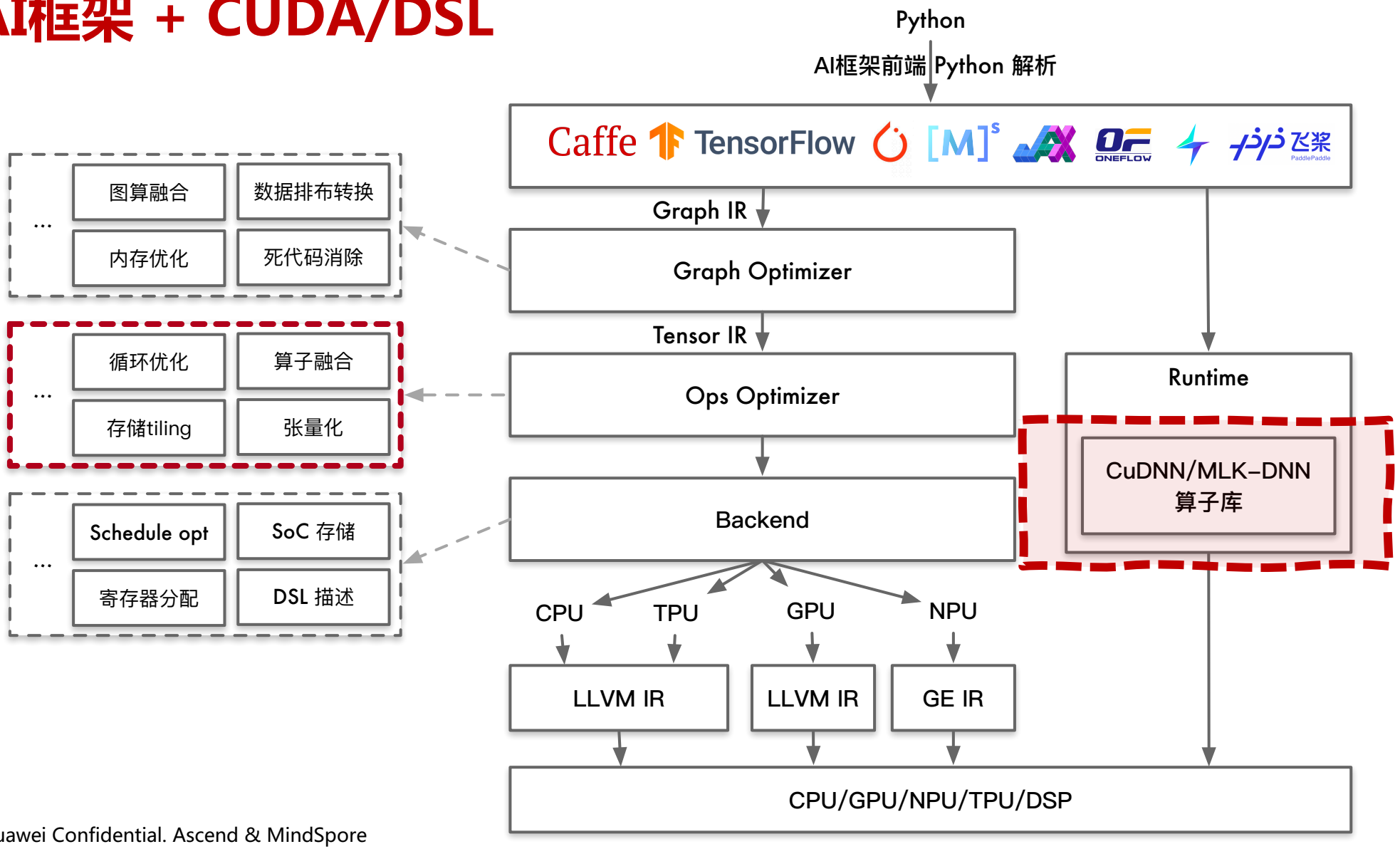
```
    int i,j;
    for (i = 0; i <200 ; i++) {
        for (j = 0; j < 10000; j++) {
            j = i * j * x / (y - 1);
        }
    }

    # Change to

    int i,j,tmp;
    tmp = x / (y - 1);
    for (i = 0; i <200 ; i++) {
        for (j = 0; j < 10000; j++) {
            j = i * j * tmp;
        }
    }
```

循环终止调用

- 消除循环终止时的方法调用

```
1
2    for (int i = 0; i < vec.size(); i++) {
3
4    }
5
6    # Change TO
7
8    int size = vec.size();
9    for (int i = 0; i < size; i++) {
10
11   }
12
```

AI框架 + CUDA/DSL



AI编译器的算子优化

算子调度优化方法

- 循环展开（Loop Unrolling）
- 循环分块（loop tiling）
- 循环重排（loop Reorder）
- 循环融合（loop Fusion）
- 循环拆分（loop Split）
- 向量化（Vector）
- 张量化（Tensor）
- 访存延迟（ Latency Hiding ）
- 存储分配（Memory Allocation）

循环优化（Loop Optimization）

指令优化（Instructions Optimization）

存储优化（Memory Optimization）

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.
- Loop Optimizations: how does the compiler do it? https://johnysswlab.com/loop-optimizations-how-does-the-compiler-do-it/
- Loop Optimizations: taking matters into your hands https://johnysswlab.com/loop-optimizations-taking-matters-into-your-hands/
- 编译优化之 - 通用循环优化 https://blog.csdn.net/qq_36287943/article/details/108542455
- 陈清扬编译优化，并行计算 https://www.zhihu.com/people/chenqingyang
- Loop optimization https://en.wikipedia.org/wiki/Loop_optimization