

AI编译器-系列之前端优化

常量折叠



ZOMI



Talk Overview of Frontend Optimizer

I. AI 编译器前端优化

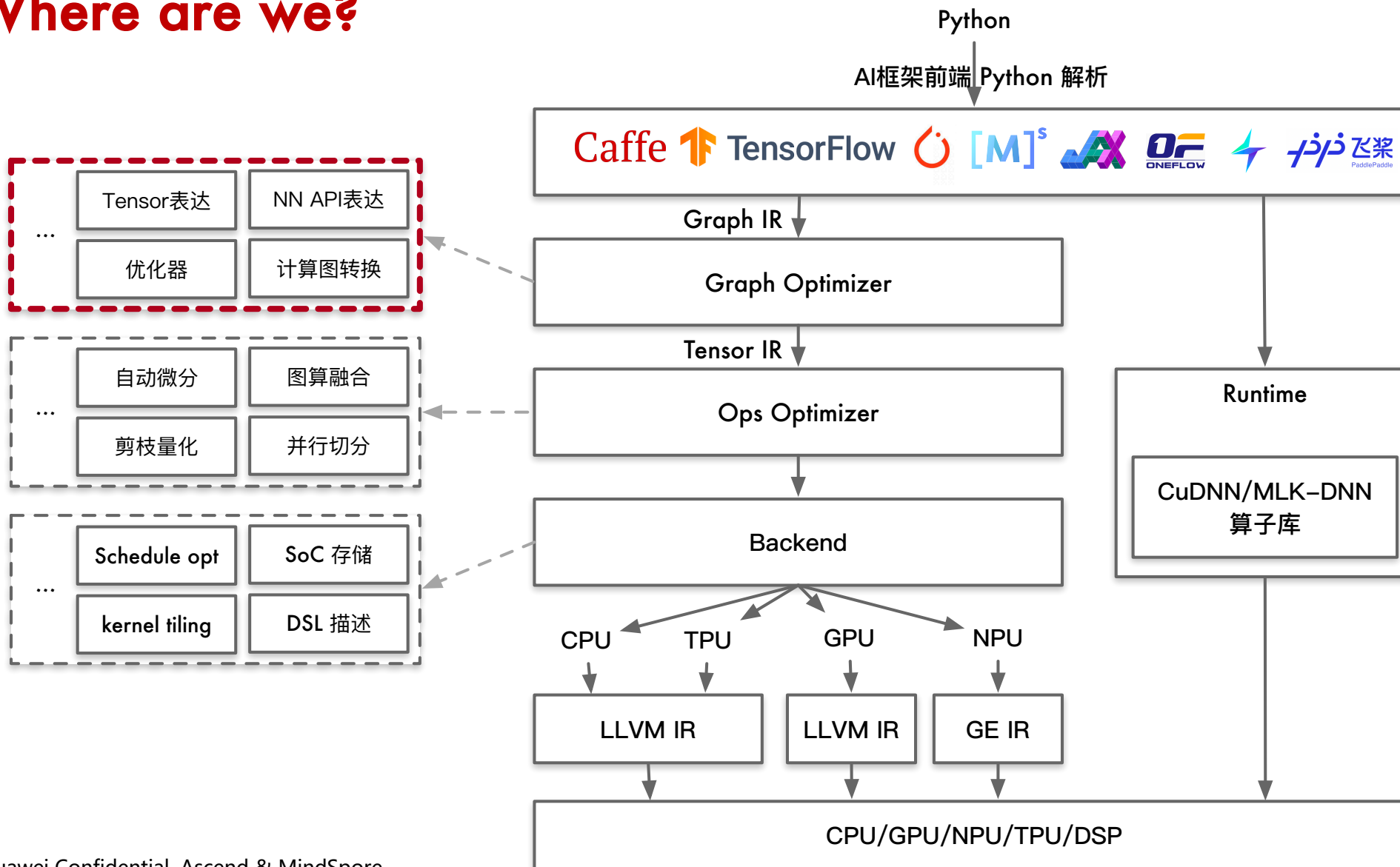
- 图层 - Graph IR
- 算子融合 - OP Fusion
- 布局转换 - Layout Transform
- 内存分配 - Memory Allocation
- 常量折叠 - Constant Fold
- 公共子表达式消除 - CSE
- 死代码消除 - DCE
- 代数简化 - ARM

Talk Overview

Constant Fold – 常量折叠

- 传统编译器中的概念
- AI编译器中的常量折叠

Where are we?



常量折叠与 常量传播

常量折叠

Constant folding , 常量折叠 , 编译器优化技术之一 , 通过对编译时常量或常量表达式进行计算来简化代码 :

```
2
3   i = 320 * 200 * 32
4
```

```
5   %a: load 320
6   %b: load 200
7   %x: mul %a, %b
8   %c: load 32
9   %y: mul %x, %c
```

```
11
12   %a: load 2,048,000
13
```

常量传播

constant propagation，常量传播，编译器优化技术之一，可以在一段代码中，将表达式中的常量替换为相关表达式或字面量，再使用常量折叠技术来简化代码。

```
2   def foo():  
3       x = 14  
4       y = 7 - x / 2  
5       return y * (28 / x + 2)
```

```
10  def foo():  
11      x = 14  
12      y = 7 - 14 / 2  
13      return y * (28 / 14 + 2)
```

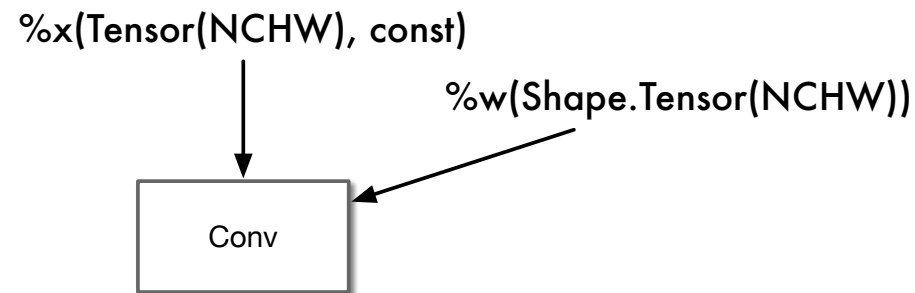
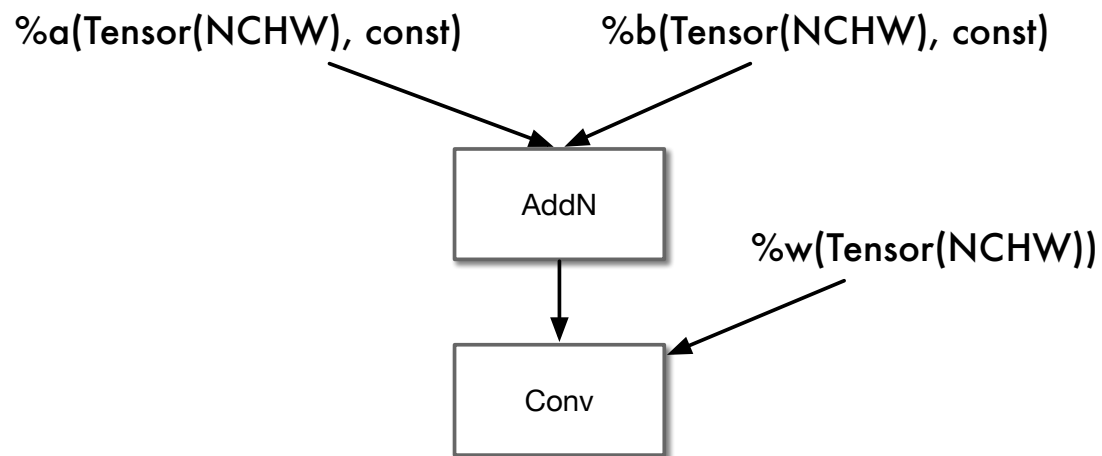
AI编译器 与常量折叠

AI编译器中的常量折叠

- 常量折叠是将计算图中可以预先可以确定输出值的节点替换成常量，并对计算图进行一些结构简化的操作。

AddN

- 对于两个形状大小为 (N,C,H,W) 四维常量Tensor，Add 结果是一定，可以将其合成一个常量放在编译器生成
- 不需要给 Add 节点分配额外的存储资源，在计算图执行的过程中，也不需要反复计算 add 这个操作，可直接进行访问结果



BN 折叠

Batch Normalization 是将各层的输入进行归一化，使训练过程更快、更稳定的一种技术。在实践中，它是一个额外的层，通常添加在Conv、MatMul、Transformer 计算层之后，在非线性之前。它包括两个步骤：

1. 首先减去其平均值，然后除以其标准差
2. 通过 Gama 缩放，通过 Beta 偏移，当网络不需要数据的时候，均值为 0、标准差为 1

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$$

BN 折叠

- 一旦训练结束，每个 BN 层都拥有一组特定的 γ 和 β ，还有 μ 和 σ ，后者在训练过程中使用指数加权平均值进行计算。这意味着在推理过程中，BN 就像是对上一层的结果进行简单的线性转换。
- 由于卷积也是一个线性变换，这也意味着这两个操作可以合并成一个单一的线性变换，这将删除一些不必要的参数，但也会减少推理时要执行的操作数量。

$$z = W * x + b$$

$$out = \gamma \cdot \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

BN 折叠

- 合并 BN 层后的卷积层的权重和偏置可以表示为：

$$w_{\text{fold}} = \gamma \cdot \frac{W}{\sqrt{\sigma^2 + \epsilon}}$$

$$b_{\text{fold}} = \gamma \cdot \frac{b - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

$$z = W_{\text{fold}} * x + b_{\text{fold}}$$

- 一般 Conv 后面接 BN 的时候很多情况下是不带 Bias 的，这个时候上面的公式就会少第二项。

BN 折叠

模型	CPU 前向时间	GPU 前向时间
Resnet50 (合并前)	176.17ms	11.03ms
Resnet50 (合并后)	161.69ms	7.3ms
提升	~8%	~34%

AI编译器

实现常量折叠

分类

1. 传统编译器中的常量折叠，找到输入节点均为常量的节点，提前计算出该节点的值来完整替换该节点。
2. 常量折叠与数据形状 shape 有关系，通过计算图已有信息推断出形状结果后，用来代替原来的节点。
3. 常量折叠与已知常量的代数化简有关。

分类

1. 传统编译器中的常量折叠，找到输入节点均为常量的节点，提前计算出该节点的值来完整替换该节点。
2. 常量折叠与数据形状 shape 有关系，通过计算图已有信息推断出形状结果后，用来代替原来的节点。
3. 常量折叠与已知常量的代数化简有关。

TensorFlow 常量折叠PASS

1. 先处理 “Shape” , “Size” , “Rank” 三类运算节点，其输出都取决与输入Tensor的形状，而与具体的输入取值没关系，所以输出可能是可以提前计算出来的。
2. 把 “Shape” , “Size” , “Rank” 上述三类运算节点，提前计算出输出值替换成 Const 节点，目的是方便后续的常量折叠和替换。
3. 折叠计算图操作：如果一个节点的输入都是常量，那么它的输出也是可以提前计算的，基于这个原理不断地用常量节点替换计算图中的此类节点，直到没有任何可以替换的节点为止。
4. 处理 Sum, Prod, Min, Max, Mean, Any, All 这几类运算节点，这几类节点的共同点是都沿着输入Tensor的一定维度做一定的运算，或是求和或是求平均等等，将符合一定条件的这几类节点替换为 Identity 节点。

Reference

1. 编译器优化 -- 常量折叠 https://blog.caoxudong.info/blog/2013/10/23/compiler_optimizations_constant_folding
2. 神经网络编译器-常量折叠 <https://blog.csdn.net/free1993/article/details/111480268>
3. https://en.wikipedia.org/wiki/Constant_folding





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.