HUAWEI

华为云学院

学以致用　云世界大有可为

基于昇腾AI处理器的算子开发

TBE算子开发实战

华为云 +智能，见未来

www.huaweicloud.com

目标

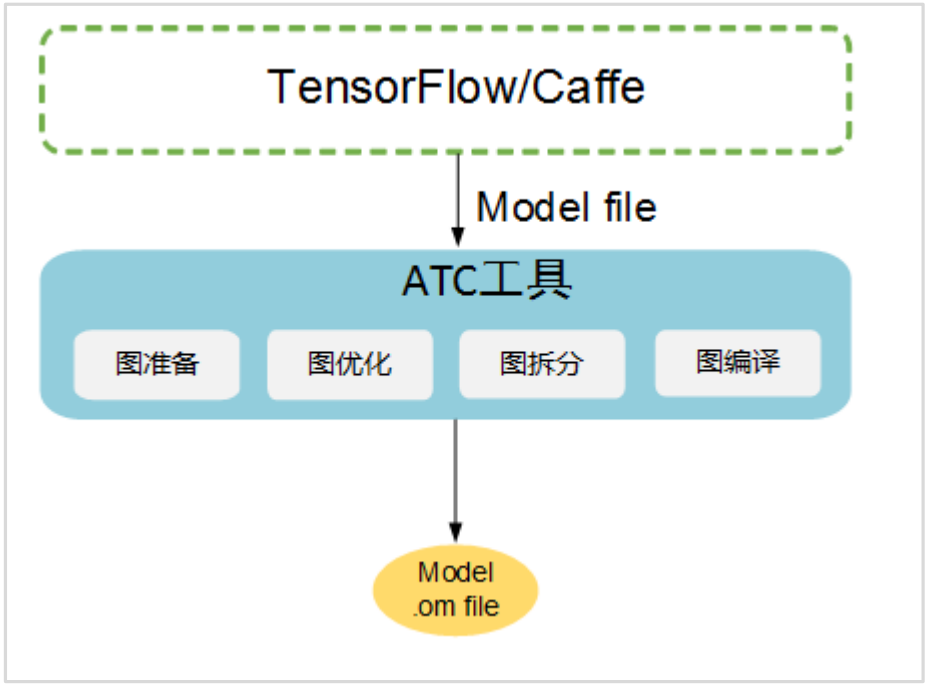
- 学完本课程后，您将能够：
  - 了解TBE和算子的基本知识和基本概念
  - 基于TBE工具开发自定义算子

目录

Ascend 模型转换导航

模型转换概述

模型转换，即将开源框架的网络模型（如Caffe、TensorFlow等），通过ATC（Ascend Tensor Compiler）模型转换工具，将其转换成昇腾AI处理器支持的离线模型，模型转换过程中可以实现算子调度的优化、权值数据重排、内存使用优化等，可以脱离设备完成模型的预处理。



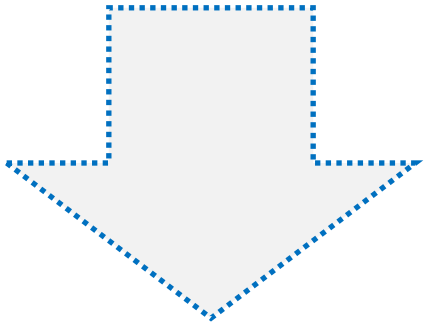
模型转换常见问题

离线模型转换过程中，80%左右的问题，集中在算子不支持。

1、新网络，其中算子未开发或发布；

2、原框架自定义算子，需要在新框架重新适配开发；

3、算子泛化不够，某些数据大小不支持。



TBE自定义算子

目录

什么是TBE

- TBE（Tensor Boost Engine）自定义算子开发工具：
  - 一款华为自研的算子开发工具，用于开发运行在NPU上的TBE算子
  - 在TVM（ Tensor Virtual Machine ）框架基础上扩展
  - 提供了一套Python API来实施开发活动

为什么要自定义TBE算子

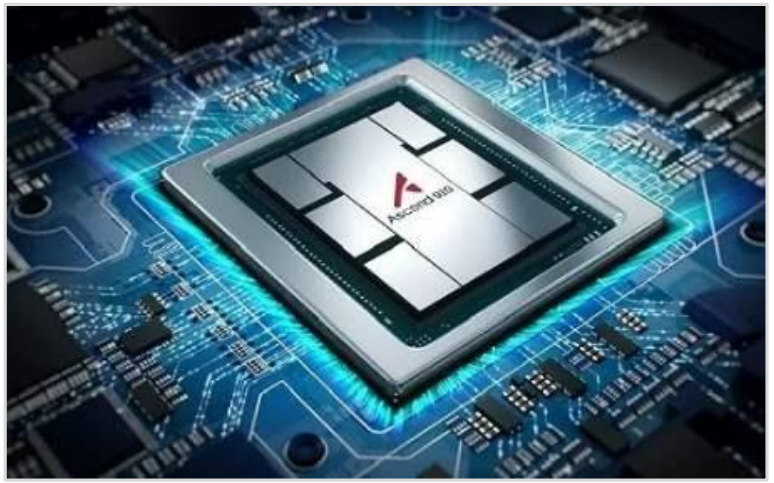
- 昇腾AI软件栈不支持开发者网络中的算子。
- 开发者想要自己开发算子来提高计算性能。
- 开发者想要修改现有算子中的计算逻辑

TBE基本概念 - NPU

- NPU（Neural-network Processing Unit），神经网络处理器。
- 在维基百科中，NPU这个词条被直接指向了“人工智能加速器”，释义是这样的：

“**人工智能加速器**（英语：**AI accelerator**）是一类专用于人工智能（特别是人工神经网络、机器视觉、机器学习等）硬件加速的微处理器或计算系统。典型的应用包括机器人学、物联网等数据密集型应用或传感器驱动的任务。”

- 本系列课程中，NPU可以特指为昇腾AI处理器。



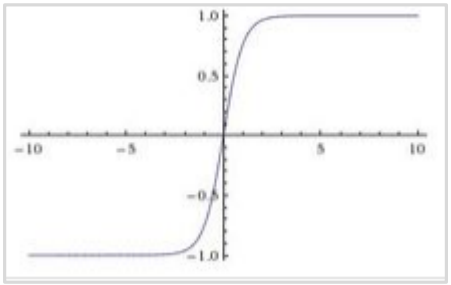
TBE基本概念 - 算子

■ 算子基本概念

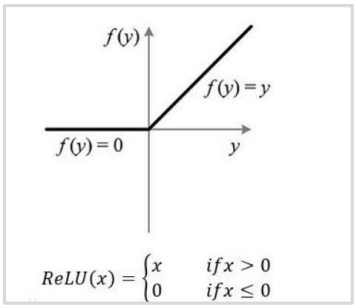
算子是一个函数空间到函数空间上的映射O：X→X；广义的讲，对任何函数进行某一项操作都可以认为是一个算子。于我们而言，我们所开发的算子是网络模型中涉及到的计算函数。在Caffe中，算子对应层中的计算逻辑，例如：卷积层（Convolution Layer）中的卷积算法，是一个算子；全连接层（Fully-connected Layer， FC layer）中的权值求和过程，是一个算子。

■ 算子举例：

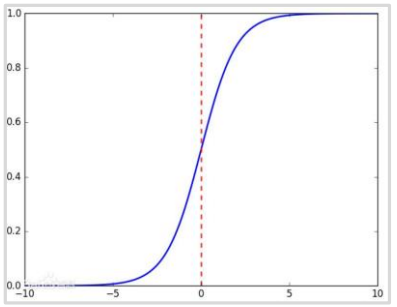
在网络模型中被用作激活函数的算子：tanh ReLU Sigmoid等。



Tanh函数图像



ReLU函数图像

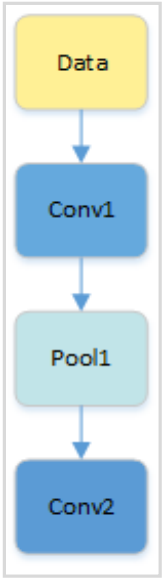


Sigmoid函数图像

图片出处：https://baike.baidu.com/item/%E6%BF%80%E6%B4%BB%E5%87%BD%E6%95%B0/2520792?fr=aladdin

TBE基本概念 - 算子类型/名称

- 算子类型（Type）：
  - 算子的type，代表算子的类型，例如卷积算子的类型为Convolution，在一个网络中同一类型的算子可能存在多个。
- 算子名称（Name）：
  - 算子的名称，用于标识网络中的某个算子，同一网络中每一个算子的名称需要保持唯一。如下图所示conv1，pool1，conv2都是此网络中的算子名称，其中conv1与conv2算子的类型为Convolution，表示分别做一次卷积运算。



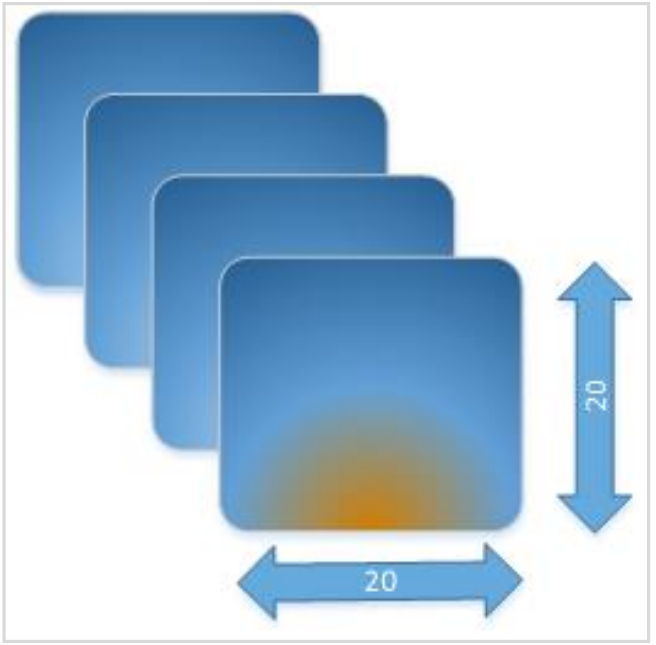
TBE基本概念 - 张量

- 张量（Tensor）：
  - Tensor是TBE算子中的数据，包括输入数据与输出数据，TensorDesc（Tensor描述符）是对输入数据与输出数据的描述，TensorDesc数据结构包含如下属性：

属性	定义
名称（name）	用于对Tensor进行索引，不同Tensor的name需要保持唯一。
形状（shape）	Tensor的形状，比如（10,）或者（1024，1024）或者（2，3，4）等。 默认值：无 形式：(i1, i2,...in)，其中i1，i2，in为正整数
数据类型（dtype）	功能描述：指定Tensor对象的数据类型。 默认值：无 取值范围：float16, float32, int8, int16, int32, uint8, uint16, bool。 说明 不同计算操作支持的数据类型不同，详细请参见9 接口参考。
数据排布格式（format）	多个维度的排布顺序。

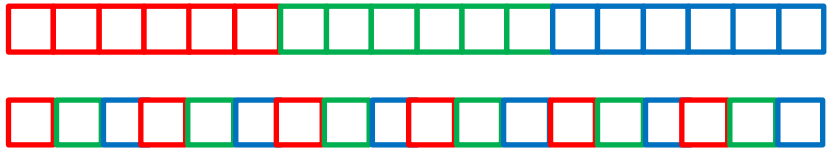
TBE基本概念 - 张量形状（Shape）

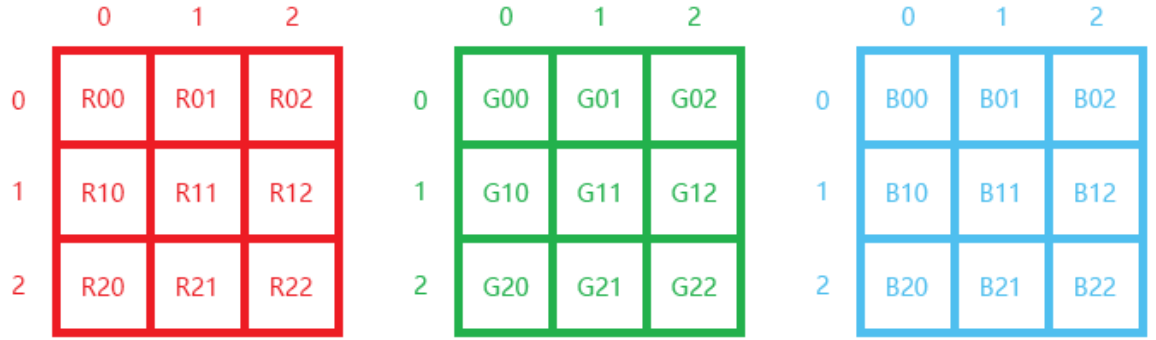
- 张量形状的物理含义：
  - 假设我们有这样一个shape=(4, 20, 20, 3)。
  - 假设有一些照片，每个像素点都由红/绿/蓝3色组成，即shape里面3的含义，照片的宽和高都是20，也就是20*20=400个像素，总共有4张的照片，这就是shape=(4, 20, 20, 3)的物理含义。



TBE基本概念 - 数据排布格式

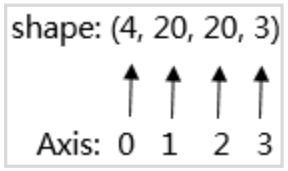
- 张量（Tensor）：
  - 数据排布格式（Format）：
    - 在深度学习框架中，多维数据通过多维数组存储，比如卷积神经网络的特征图用四维数组保存，四个维度分别为批量大小（Batch，N）、特征图高度（Height，H）、特征图宽度（Width，W）以及特征图通道（Channels，C）。
    - 由于数据只能线性存储，因为这四个维度有对应的顺序。不同深度学习框架会按照不同的顺序存储特征图数据，比如Caffe，排列顺序为[Batch, Channels, Height, Width]，即NCHW。Tensorflow中，排列顺序为[Batch, Height, Width, Channels]，即NHWC。
    - 如下图所示，以一张格式为RGB的图片为例，NCHW实际存储的是“RRRGGGBBB”，同一通道的所有像素值顺序存储在一起，而NHWC实际存储的则是“RGBRGBRGB”，多个通道的同一位置的像素值顺序存储在一起。





TBE基本概念 - 算子属性

- 算子属性
  - 不同的算子包含的属性值不同，下面介绍几个常见的算子属性：
  - 轴（Axis）：
    - axis代表张量中维度的下标，比如张量a是一个5行6列的二维数组，即shape是(5,6)，则axis=0表示是张量中的第一维，即行。axis=1表示是张量中的第二维，即列。
    - 例如张量数据[[[1,2],[3,4]], [[5,6],[7,8]]]，Shape为(2,2,2)，则轴0代表第一个维度的数据即[[1,2],[3,4]]与[[5,6],[7,8]]这两个矩阵，轴1代表第二个维度的数据即[1,2]、[3,4]、[5,6]、[7,8]这四个数组，轴2代表第三个维度的数据即1，2，3，4，5，6，7，8这八个数。
    - 轴axis可以为负数，此时表示是倒数第axis个维度。
    - N维Tensor的轴有：0，1，2,……，N-1。



TBE基本概念 - 升维 (1)

- 升维—广播机制（Broadcast）：
  - TBE支持的广播规则：可以将一个数组的每一个维度扩展为一个固定的shape，需要被扩展的数组的每个维度的大小或者与目标shape相等，或者为1，广播会在元素个数为1的维度上进行。
  - 例如：原数组a的维度为（2, 1, 64），目标shape为（2, 128, 64），则通过广播可以将a的维度扩展为目标shape（2, 128, 64）。
  - TBE的计算接口加、减、乘、除等不支持自动广播，要求输入的两个Tensor的shape相同，所以操作前，我们需要先计算出目标shape，然后将每个输入Tensor广播到目标shape再进行计算。

TBE基本概念 - 升维 (2)

例如，Tensor A的shape为（4, 3, 1, 5），Tensor B的shape为（1, 1, 2, 1），执行Tensor A + Tensor B，具体计算过程如下：

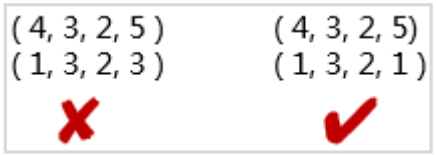
1. 计算出目标shape。

   – 取Tensor A与Tensor B中每个维度的大值，作为目标Shape，C（4, 3, 2, 5）。

2. 调用广播接口分别将Tensor A与Tensor B扩展到目标Shape C。

3. 调用计算接口，进行Tensor A + Tensor B。

   – 如果需要广播的tensor的shape不满足每一个维度的大小与目标shape相等或者为1的要求，则无法进行广播，如下示例所示：



?

(8, 3, 1)

(8, 1, 1, 5)

TBE基本概念 - 降维 (1)

- 降维（Reduction）：
  - 前面我们介绍升维-广播机制，有升就有降，Reduction是将多维数组的指定轴做降维操作。
  - Reduction算子的关键属性：
    - **ReductionOp**：常见支持的操作类型，包含四种类型：

算子类型	说明
SUM	对被reduce的所有轴求和。
MIN	对被reduce的所有轴求最小值。
MAX	对被reduce的所有轴求最大值。
PROD	对被reduce的所有轴求乘积。

    - **Axis**：Reduction需要指定若干个轴，会对这些轴进行reduce操作，取值范围为：[-N，N-1]。
    - **keepdim**: 对指定轴降维后，该轴是被消除还是保留为1，即此维度是否保留。取值为0表示消除，1表示保留。
      - 比如，输入的张量的形状为(5,6,7,8)。
        - » 如果指定的轴是3，keepdim是0，则输出tensor的形状为(5,6,7)。
        - » 如果指定的轴是2，keepdim是1，则输出tensor的形状为(5,6,1,8)。
        - » 如果指定的轴是{1,2}，keepdim是0，则输出tensor的形状为(5,8)。
        - » 如果指定的轴是{0,1,2}，keepdim是1，则输出tensor的形状为(1,1,1,8)。

TBE基本概念 - 降维 (2)

- 下面通过示例了解降维操作。
  - 如果对轴Axis=0 进行降维, 对2维矩阵来说就是行，也就是对这个2维矩阵每行对应的数据进行相加，得到[2,2,2]，降为1维，如图所示：

```
[[1 , 1 , 1]
  +   +   +
 [1 , 1 , 1]]
=[ 2 , 2 , 2]
```

  - 如果Axis=1，就是每列对应的数据进行相加。

```
[[1 , +1 , +1]      [3,
 [1 , +1 , +1]] =    3]
```

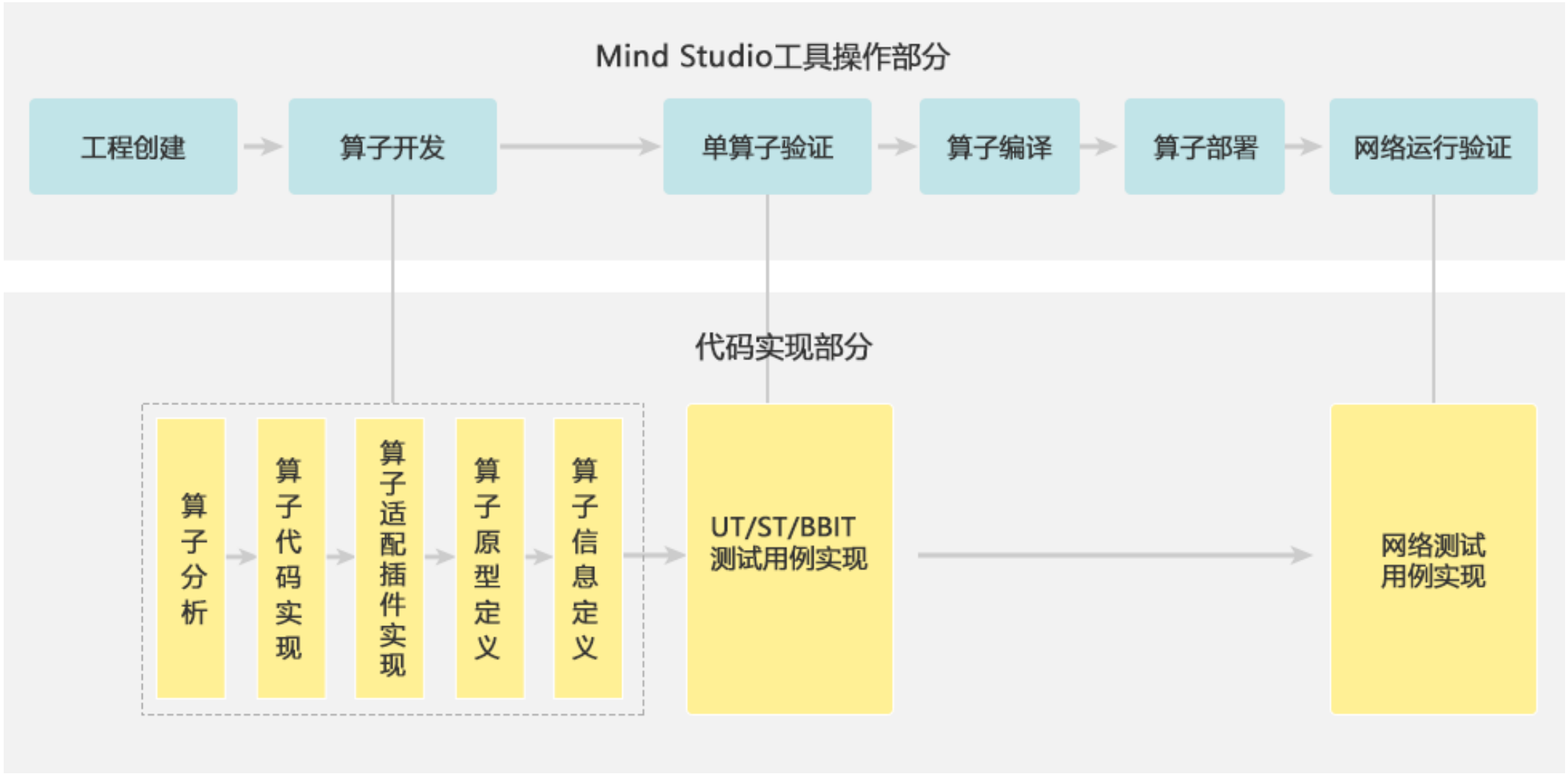
  - 如果Axis=[0,1]，可以理解为先对轴0进行降维求和得到[2,2,2], 再对[2,2,2]继续降维求和得到6，最后得到是0维；
  - 如果Axis=[]，就是不降维，原样输出。
  - 如果Axis为空，就是对所有维度进行降维，最后得到0维的标量。

TBE算子开发方式

开发方式	特点
DSL	开发难度： 易 适用人群：入门开发者，仅需要了解神经网络和TBE DSL相关知识 特点：TBE工具提供自动优化机制，给出较优的调度流程
TIK	开发难度： 难 适用人群：高级开发者，需要掌握Davinci硬件缓冲区架构，对性能有较高要求 特点：接口偏底层，用户需要自己控制数据流以及算子的硬件调度（schedule）

目录

TBE算子开发流程



举个小栗子

- 目标：

  用TBE-DSL方式开发一个Sqrt算子

- 确定算子功能：

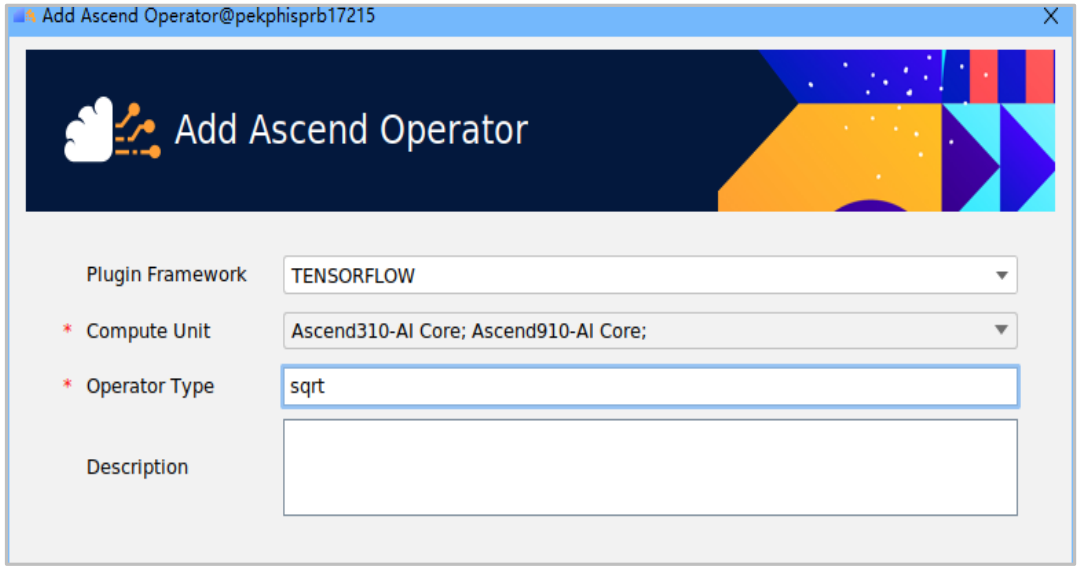
  Sqrt算子功能是对Tensor中每个原子值求开方，数学表达式子为**y=**$\sqrt{x}$。

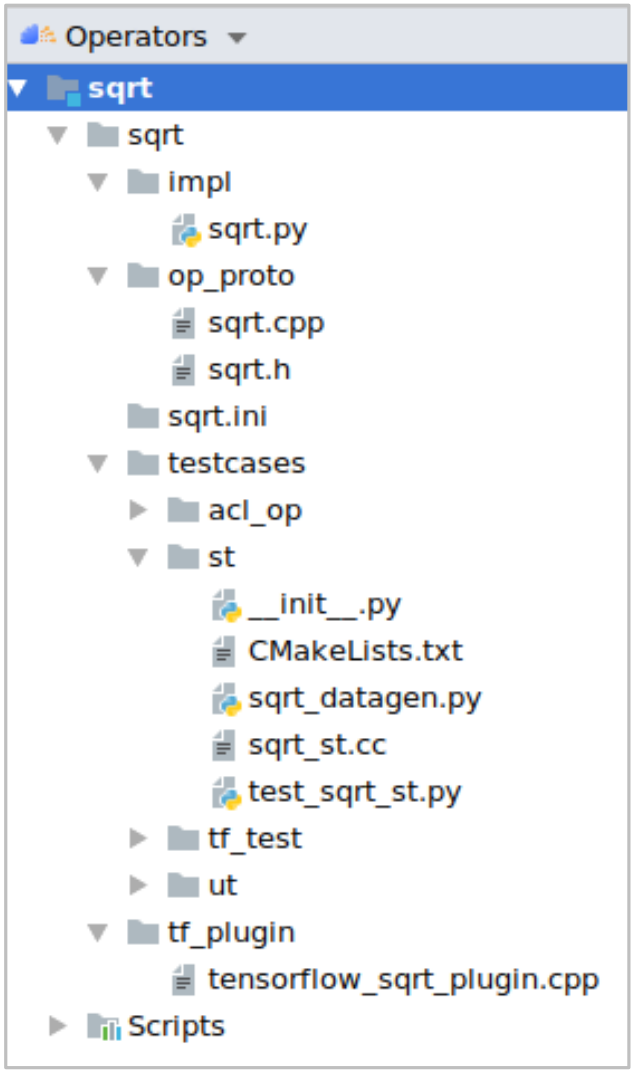
- 确定使用的计算接口：

  根据当前TBE框架可支持的计算描述API，可采用如下公式来表达Sqrt算子的计算过程：

$$y = exp(0.5*log(x))$$

工程创建





算子实现

```
def sqrt(input_x, output_y, kernel_name="sqrt"):
    shape = input_x.get("shape")
    dtype = input_x.get("dtype")
    input_dtype = dtype.lower()
    data = tvm.placeholder(shape, name="data", dtype=input_dtype)

    log_val = te.lang.cce.vlog(data)
    const_val = tvm.const(0.5, "float32")
    mul_val = te.lang.cce.vmuls(log_val, const_val)
    res = te.lang.cce.vexp(mul_val)

    with tvm.target.cce():
        schedule = generic.auto_schedule(res)

    config = {"name": kernel_name,
              "tensor_list": [data, res]}
    te.lang.cce.cce_build_code(schedule, config)
```

算子方法签名&入参

输入Tensor占位符

定义计算过程

定义调度过程

算子构建

算子实现 - 算子入参

- TBE算子基本入参

```
def sqrt(input_x, output_y, kernel_name="sqrt"):
    """
    Parameters
    ----------
    input_x : dict
              shape and dtype of input
    output_y : dict
               shape and dtype of output
    kernel_name : str
                  kernel name, default value is "sqrt"
    """
```

>shape : Tensor的属性，表示Tensor的形状，用list或tuple类型表示。
例如（3, 2, 3）、（4, 10）。
>dtype ： Tensor的数据类型，用字符串类型表示。
例如“float32”、“float16”、“int8”等。

算子实现 - 输入占位符

- TBE算子输入占位符

```
data = tvm.placeholder(shape, name="data", dtype=input_dtype)
```

tvm.placeholder()是tvm框架的API，用来为算子执行时接收的数据占位，通俗理解与C语言中%d、%s一样，返回的是一个Tensor对象，上例中使用data表示；入参为shape，name，dtype,是为Tensor对象的属性。

这里的输入是指算子执行时的输入数据，与编译时期入参不同，编译时期入参input_x(shape,type)是为了得到算子执行文件的入参。

算子实现 - 定义计算过程

- TBE算子中定义计算过程

  定义计算过程是指使用DSL语言，根据数学算式，描述出实现算子功能的计算步骤，以算子sqrt为例：

$$y = \exp(0.5*\log(x))$$

```
log_val = te.lang.cce.vlog(data)
const_val = tvm.const(0.5, "float32")
mul_val=te.lang.cce.vmuls(log_val, const_val)
res = te.lang.cce.vexp(mul_val)
```

根据sqrt算子的函数表达式，描述sqrt的计算过程如上所示，其中te.lang.cce.*name*()皆为TBE框架的API.

算子实现 - 调度

- TBE算子中调度（schedule）操作

样例：

```
with tvm.target.cce():
    schedule = generic.auto_schedule(res)
```

计算过程描述完之后，就会做调度；调度是与硬件相关的，功能主要是调整计算过程的逻辑，意图优化计算过程，使计算过程更高效，以及保证计算过程中占用硬件存储空间不会超过上限。

算子实现 - 构建

- TBE算子中的构建操作

样例：

```
config = {"name": kernel_name,
                  "tensor_list": [data, res]}
te.lang.cce.cce_build_code(sch, config)
```

TBE框架提供了cce_build_code()API，传入schedule以及相关的配置项，即可完成编译，生成最终硬件可执行文件。

算子 Compute API

- TBE框架提供描述计算过程的API：

TBE算子都是调用框架提供的computeAPI来描述计算过程，接口皆为te.lang.cce.name的形式；compute API 现根据功能类型可分为以下几类：

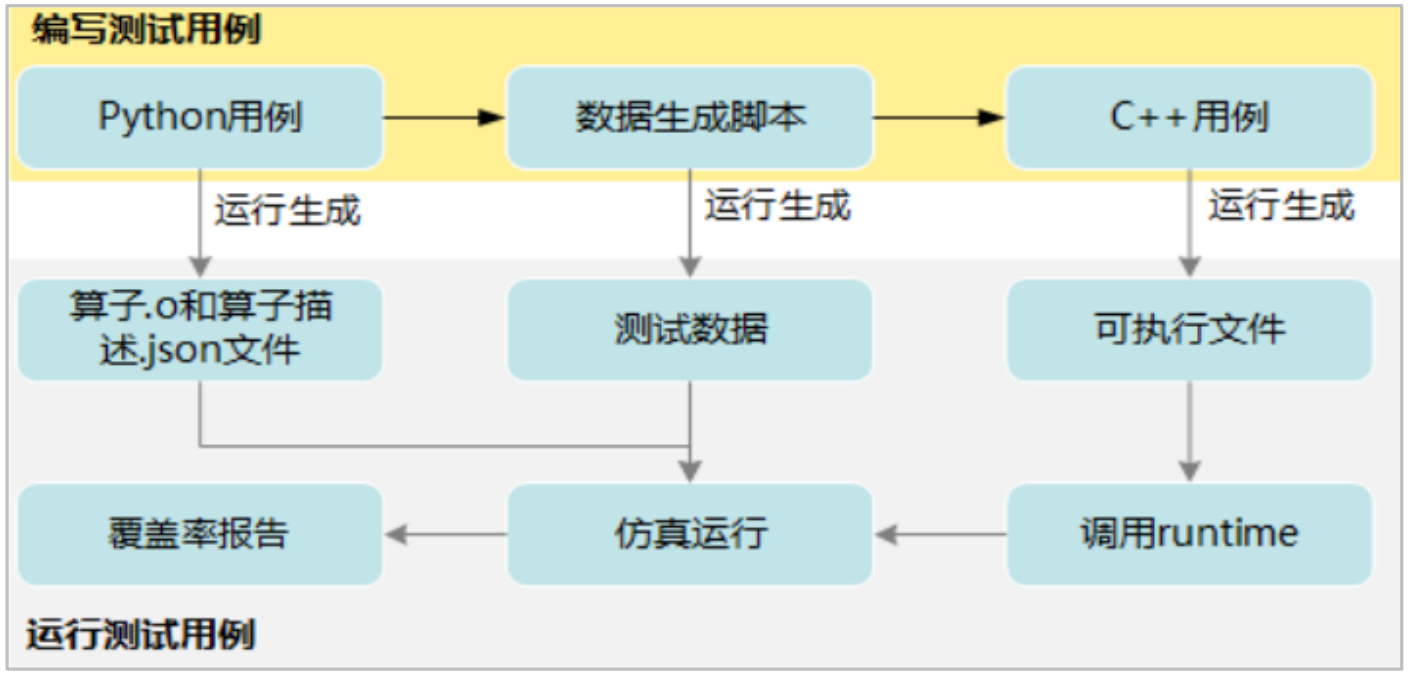
接口分类	作用
elewise_compute	对Tensor中每个原子值分别做相同操作，样例中te.lang.cce.vabs 即对每个数值x求绝对值
reduction_compute	对Tensor按轴进行操作，例如te.lang.cce.sum(data,axis)表示对data按axis进行累加
segment_compute	对Tensor进行分段操作
cast_compute	对Tensor中数据的数据类型进行转换，例如float16转换为float32
broadcast_compute	对Tensor按照目标shape进行广播，如shape为（3,1,2）的Tensor广播成(3,3,2)的Tensor
mmad_compute	支持矩阵乘法
卷积相关compute	除以上几种compute，还有一些专门针对卷积的compute

目录

算子测试 - ST

ST测试即系统测试（System Test），在仿真环境下，验证算子功能的正确性，包括：

- 正确的生成.o和.json文件。ST侧重测试各种数据类型、shape等场景下生成的算子的正确性。
- 测试算子逻辑的正确性。ST通过C++用例调用runtime的接口，然后端到端计算出执行结果，并取回结果和预期结果（Tensorflow或者numpy生成）进行比较，来测试算子逻辑的正确性。也可进行性能分析。
- ST测试用例在仿真环境下运行。



本章总结

- 本课程介绍了TBE算子的基本知识、概念以及两种开发方式（DSL、TIK），并对一个简单DSL算子进行拆解并介绍了如何进行功能测试。通过一个单算子开发的完整流程讲解，希望对于后续的算子开发有一定的帮助。

学习推荐

- TBE认证培训课程（初级）
- TBE认证培训课程（高级）

TBE：高效算子开发工具启动公测

Scan for the tool

扫码获取工具

HUAWEI

华为云学院

学以致用　云世界大有可为

Thank You.