深度学习网络性能调优实践

wangxiaolei14@Huawei.com

Security Level:

HUAWEI

QCon+ 案例研习社

扫码了解 QCon+

学习前沿案例，向行业领先迈进

40 个
热门专题
—
行业专家把关内容筹备，
助你快速掌握最新技术发展趋势

200 个
实战案例
—
了解大厂前沿实战案例，
为 200 个真问题找到最优解

40 场
直播答疑
—
40 位技术大咖，每周分享最新
技术认知，互动答疑

365 天
持续学习
—
结合配套 PPT，学习社群引导，
畅学 365 天

InfoQ PRO

关注 InfoQ Pro 服务号

你将获得：

- InfoQ 技术大会讲师 PPT 分享
- 最新全球 IT 要闻
- 一线专家实操技术案例
- InfoQ 精选课程及活动
- 定期粉丝专属福利活动

目录

给定任务KPI和预算，应该怎么样压缩成本？

- 我们常问的面试题……
- 给定精度指标，每秒完成3000张1080p图片推理，你需要多少钱？

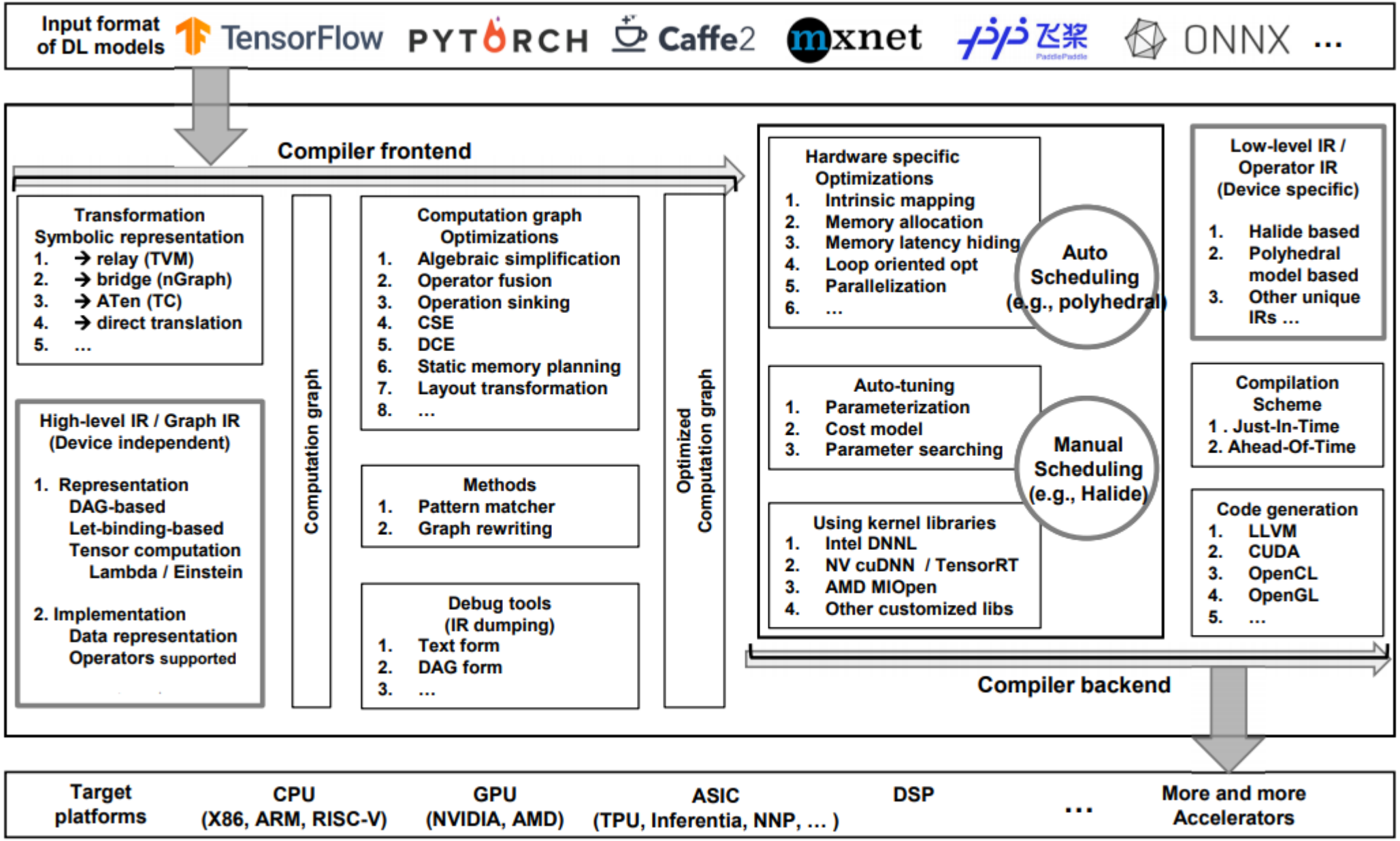
如何加速模型的执行效率？

- 结构优化
- 剪枝
- 量化
- 蒸馏
- ……

- 思考：
- 以上种种都需要投巨大人力……有没有什么省力的办法？

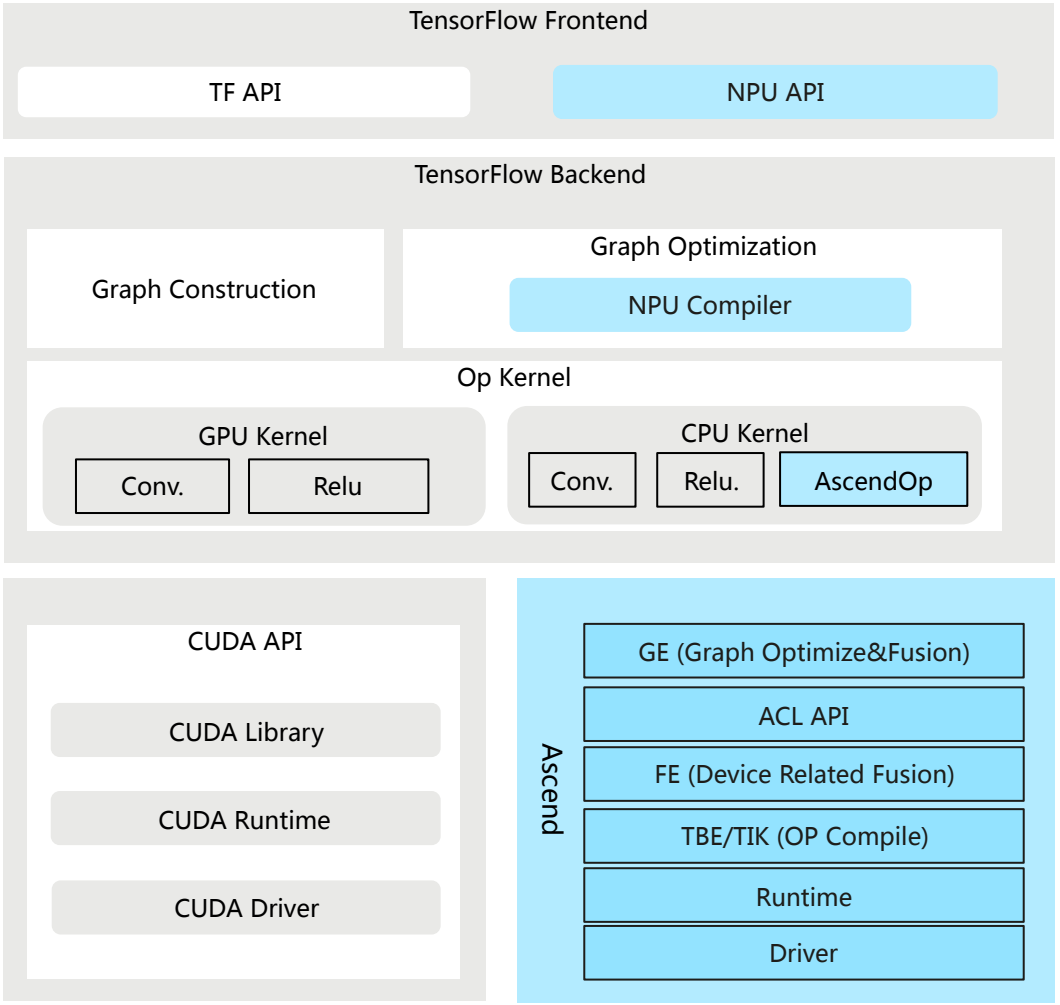
目录

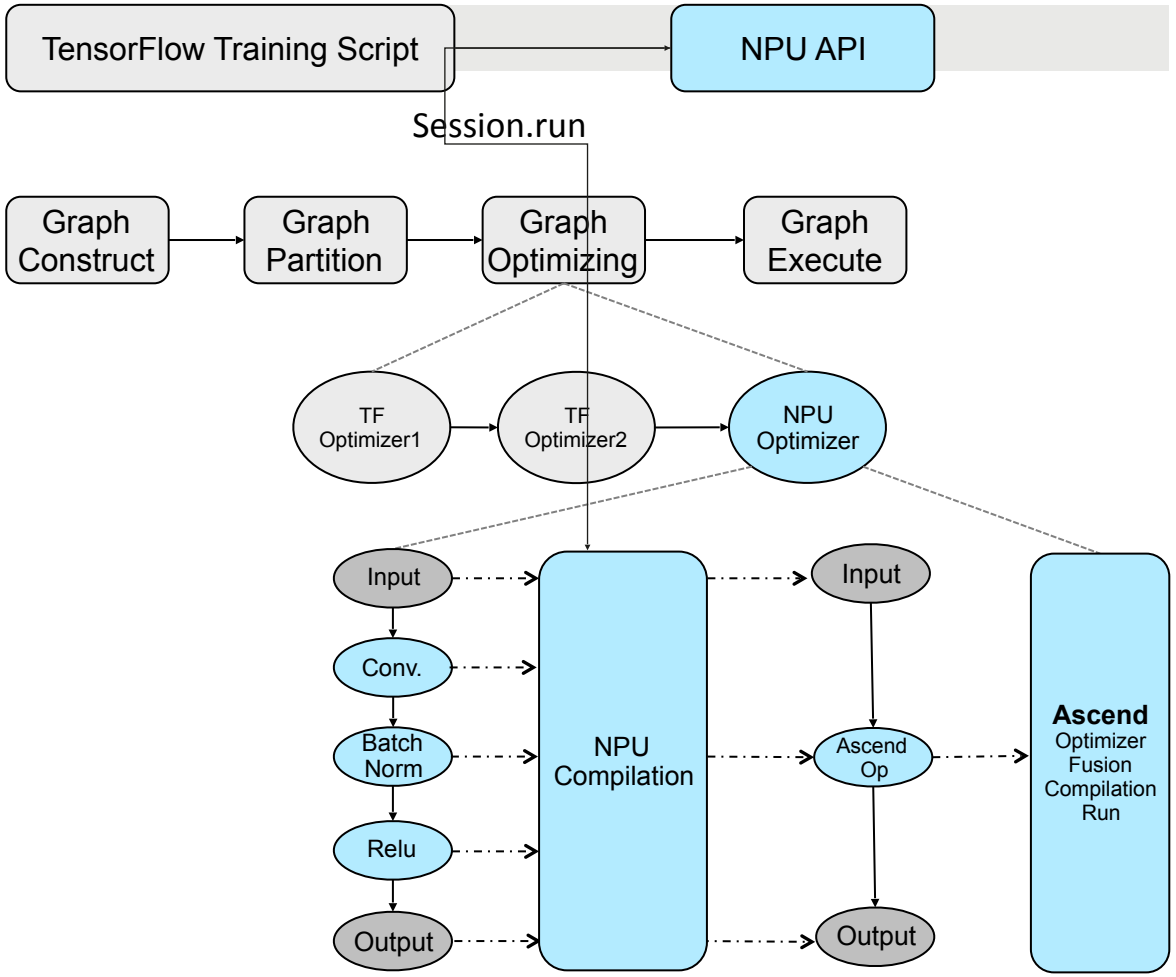
- 深度学习网络性能调优实践
- 计算框架
  - 计算框架设计
  - 图优化编码实战
- 实战技巧
  - “硬”
  - “凑”
  - “紧”
  - “裁”



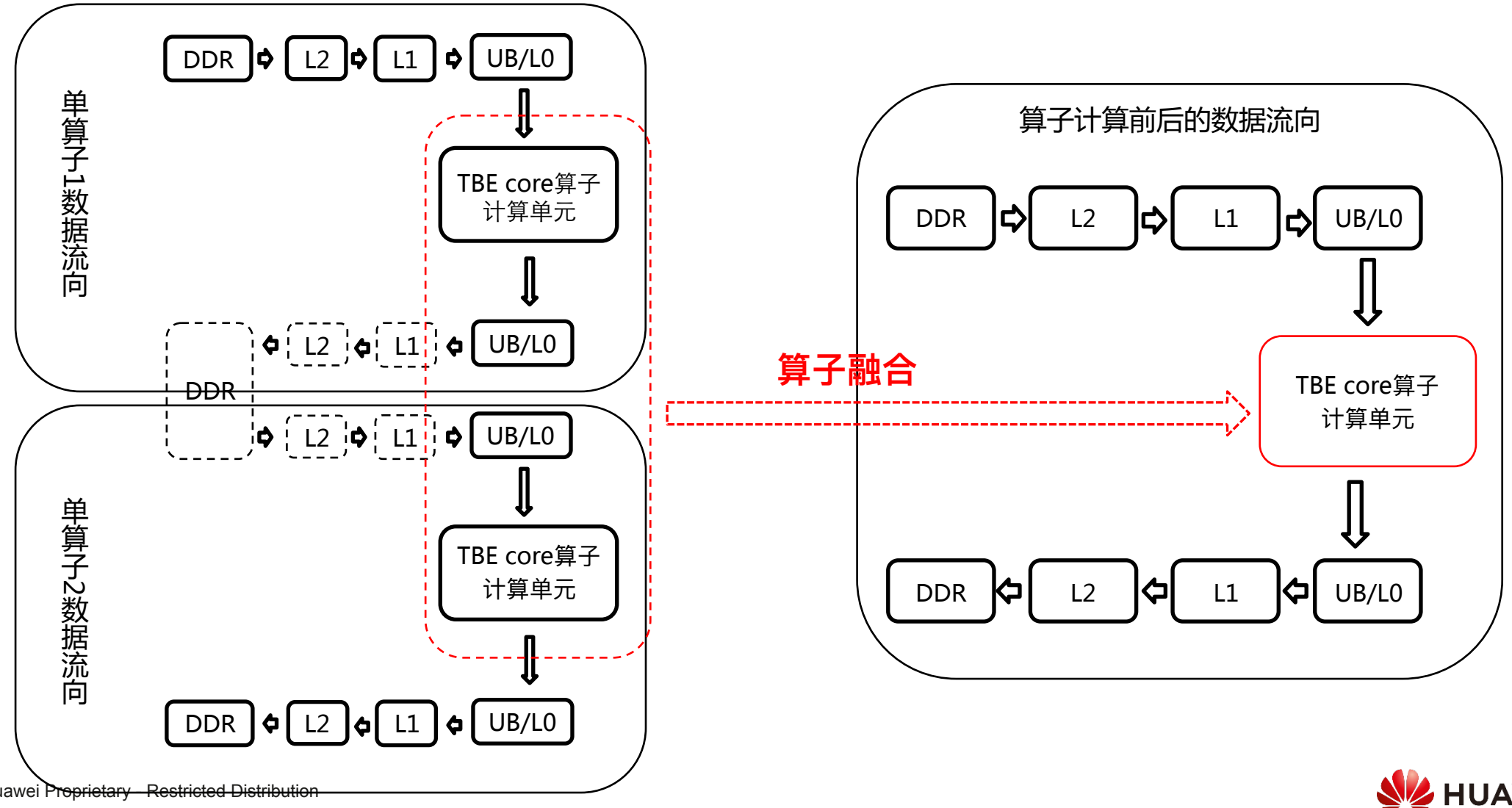
https://arxiv.org/pdf/2002.03794v4.pdf

比如说……





一个典型的提速方法：算子融合



Add（加法）算子实现

```
def add_compute(input_x, input_y, output_z, kernel_name="add"):

#形状推导
shape_x = te.lang.cce.util.shape_to_list(input_x.shape)
shape_y = te.lang.cce.util.shape_to_list(input_y.shape)

shape_x, shape_y, shape_max = util.produce_shapes(shape_x, shape_y)
util.check_shape_size(shape_max, SHAPE_SIZE_LIMIT)
input_x = te.lang.cce.broadcast(input_x, shape_max)
input_y = te.lang.cce.broadcast(input_y, shape_max)

#计算逻辑
res = te.lang.cce.vadd(input_x, input_y)

return res
```

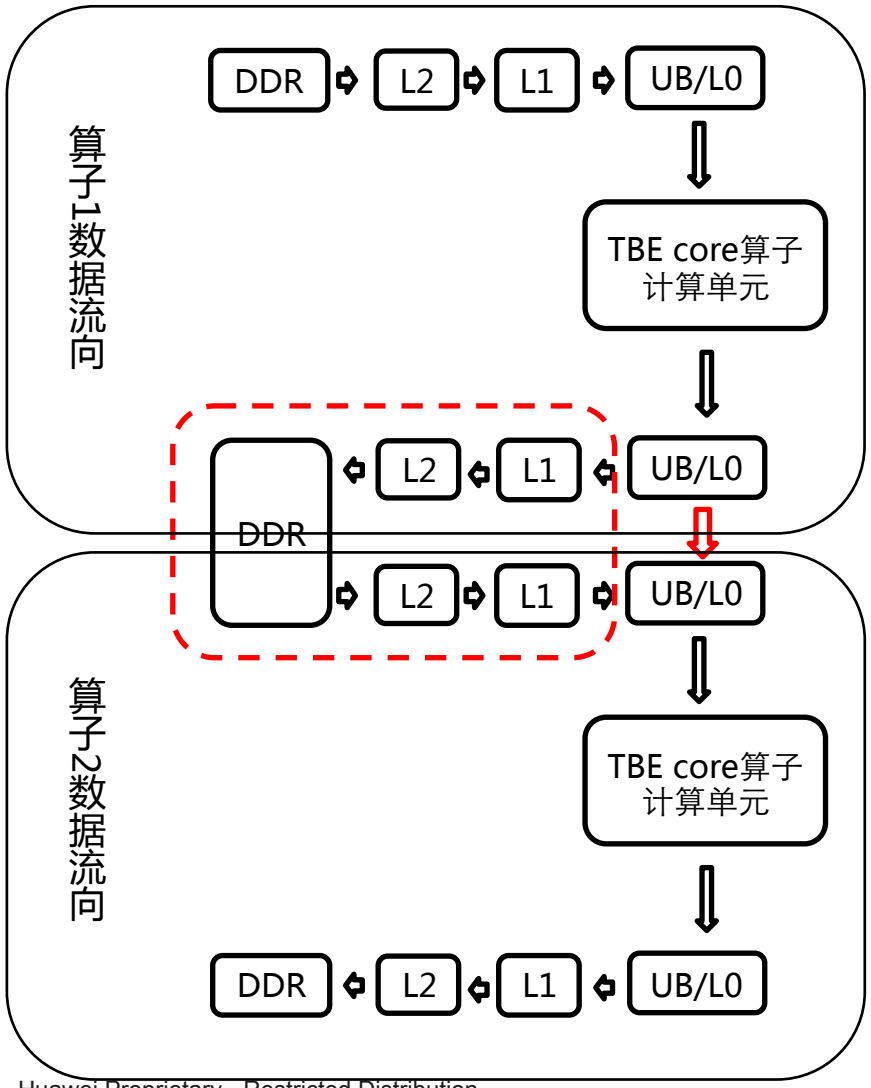
ReLu算子实现

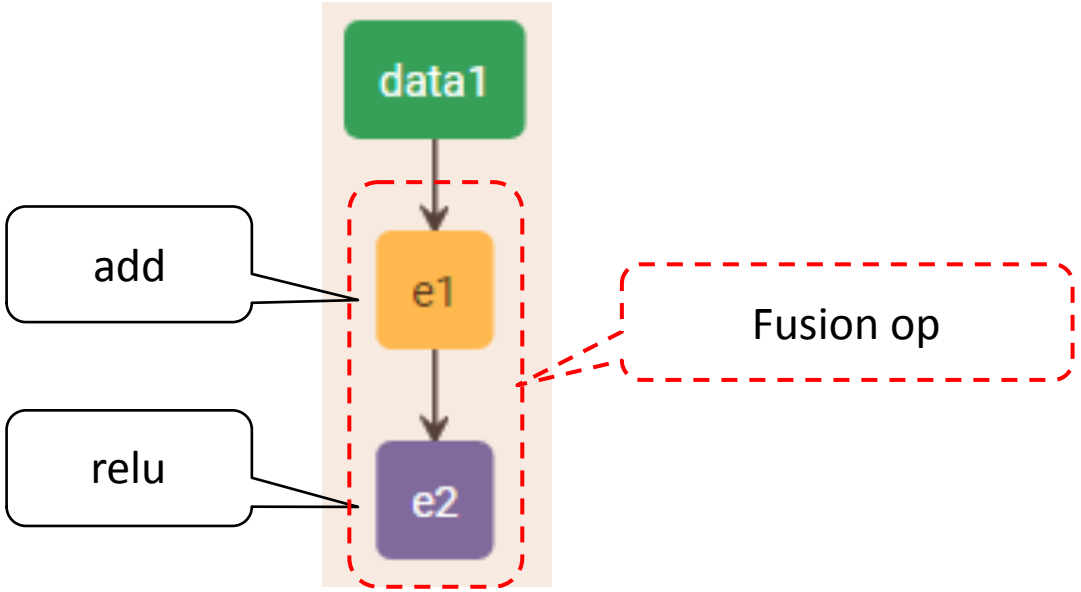
```
def relu_compute(input_x, output_y, kernel_name="relu"):

# 不涉及形状推导
data_res = te.lang.cce.vrelu(input_x)

return data_res
```

add+relu 的算子融合实现

```
@fusion_manager.register(“fuse_add_relu”) #算子注册
def fuse_add_relu_compute(input_x, input_y, output_z,
kernel_name="fuse_add_relu"):
#形状推导
shape_x = te.lang.cce.util.shape_to_list(input_x.shape)
shape_y = te.lang.cce.util.shape_to_list(input_y.shape)

shape_x, shape_y, shape_max = util.produce_shapes(shape_x, shape_y)
util.check_shape_size(shape_max, SHAPE_SIZE_LIMIT)
input_x = te.lang.cce.broadcast(input_x, shape_max)
input_y = te.lang.cce.broadcast(input_y, shape_max)
#计算逻辑：Add
addres = te.lang.cce.vadd(input_x, input_y)
#计算逻辑：ReLu
res = te.lang.cce.vrelu(addres)
return res
```

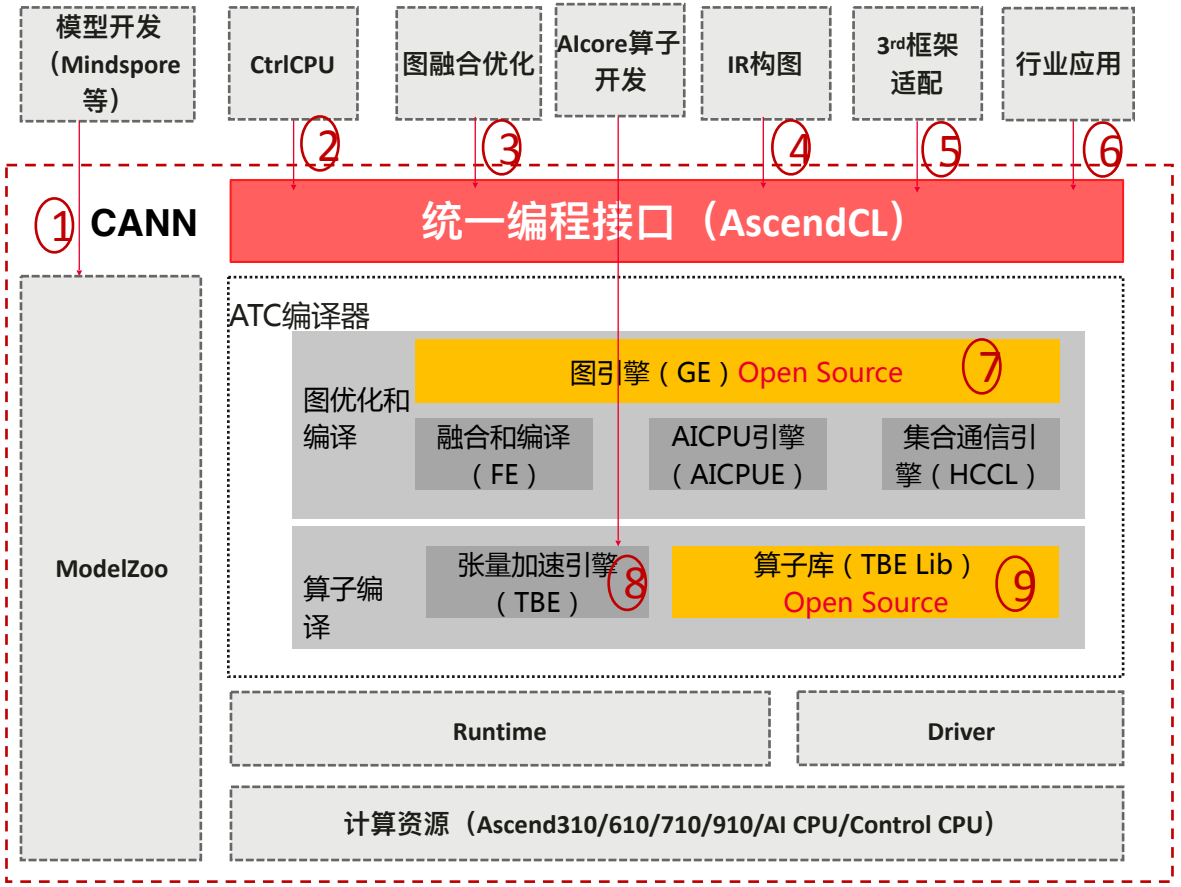
回顾：add和relu的算子融合





本节小结：CANN九大开放性设计，E2E覆盖全场景应用

- 全方位的开发入口：九大开发设计，覆盖模型的开发，应用的开发，算子开发
- 完善的工具，提升开发效率：包括模型分析，代码开发，调试，调优，诊断
- 全面开源：所有的模型脚本，算子代码，应用Demo代码



序号	描述	说明
1	Modelzoo：预置大量预训练模型	用户可以直接使用，也可以重训或Fine-tune。
2	CtrlCPU开放	用户可以将应用软件运行在昇腾芯片的ARMCPU上。
3	图融合优化接口开放	用户高效自定义算子融合规则
4	IR构图接口开放	用户高效自定义模型
5	3rd主流框架适配	TF/PYTORCH/CAFFE等第三方框架适配
6	行业应用资源调用接口开放	易于调用AI算力
7	提供图引擎的开源版本	用户可自主修改和定制图引擎代码，满足个性需求
8	TBE，算子开发体系	用户高效开发自定义算子
9	预置算子库以源码发布	用户进行小量修改即可自定义算子

目录

如何加速模型的执行效率？

- 结构优化
- 剪枝
- 量化
- 蒸馏
- ……

- 思考：
- 和硬件相关的加速方法有哪些？

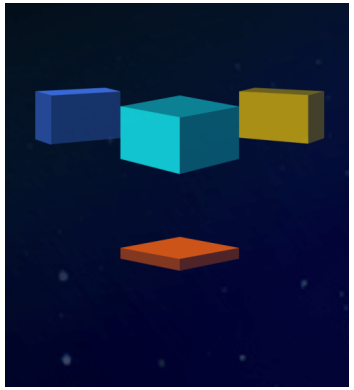
方法1： “硬设计”

Fast

```
int A[M][K], B[K][N], C[M][N];
//Scalar
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < K; k++) {
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}
//Vector
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        C[i][j] = mac(A[i][*], B[*][j]);
    }
}
//Cube
C = gemm(A, B);
```

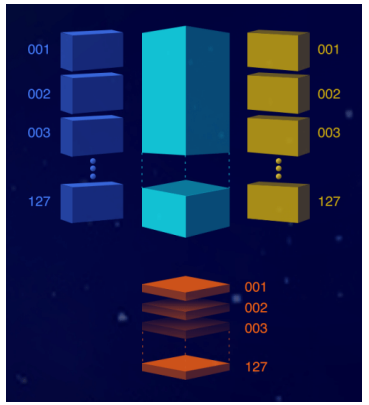
Flexible

CPU
Scalar Compute



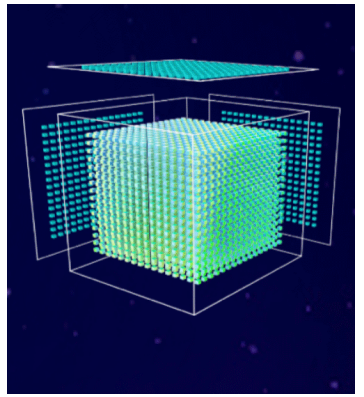
0.00X TOPS / W

GPU
Vector Compute



0.X TOPS / W

DaVinci
Tensor Compute

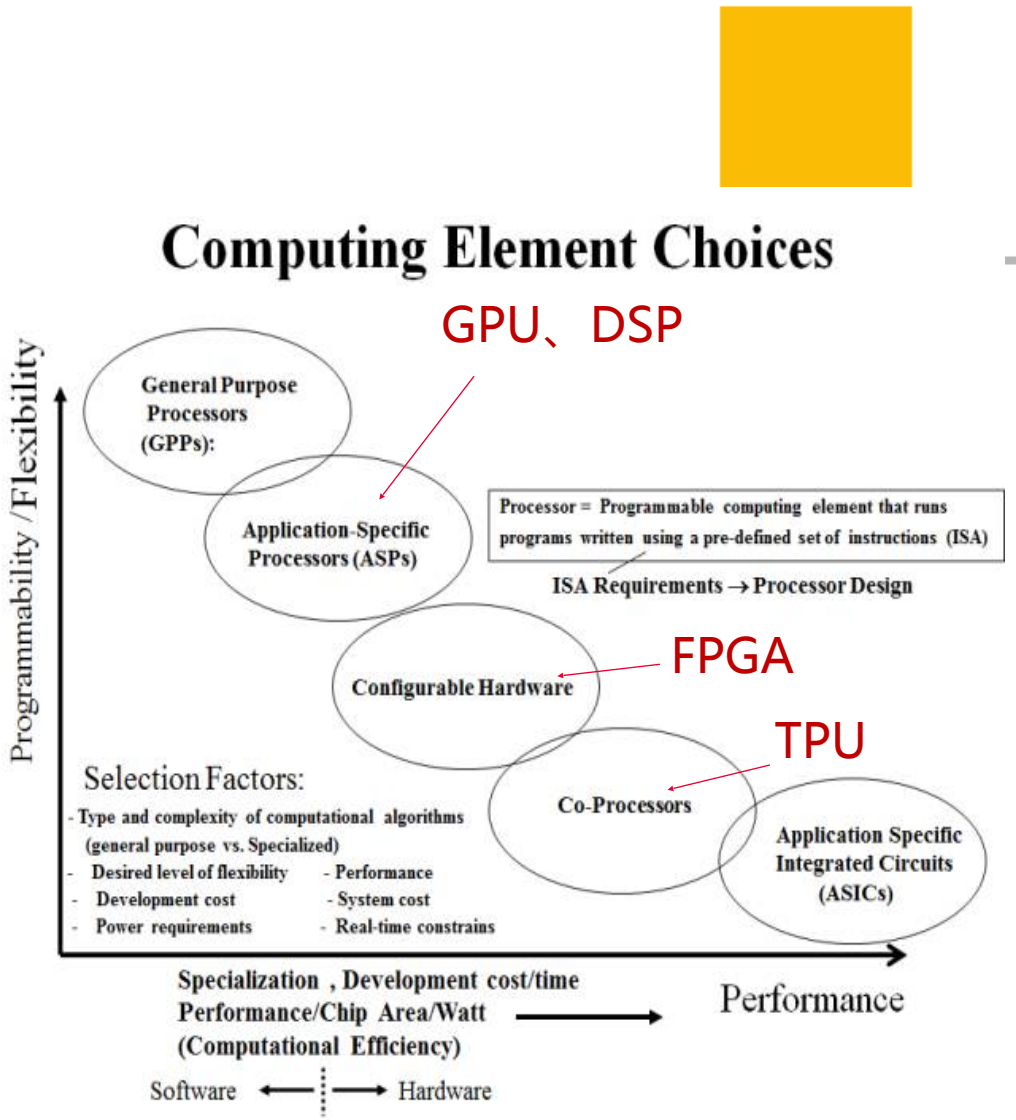


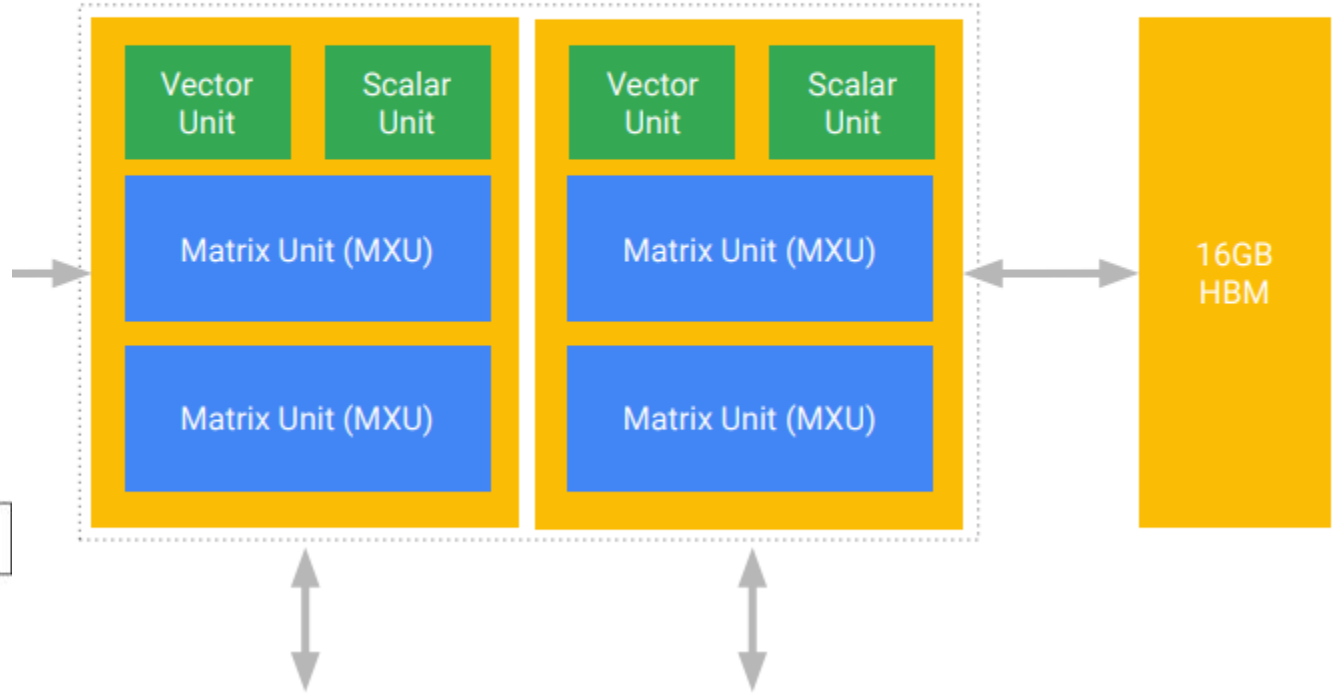
X TOPS / W

- 16

HUAWEI

Cloud TPU v3 Chip Architecture

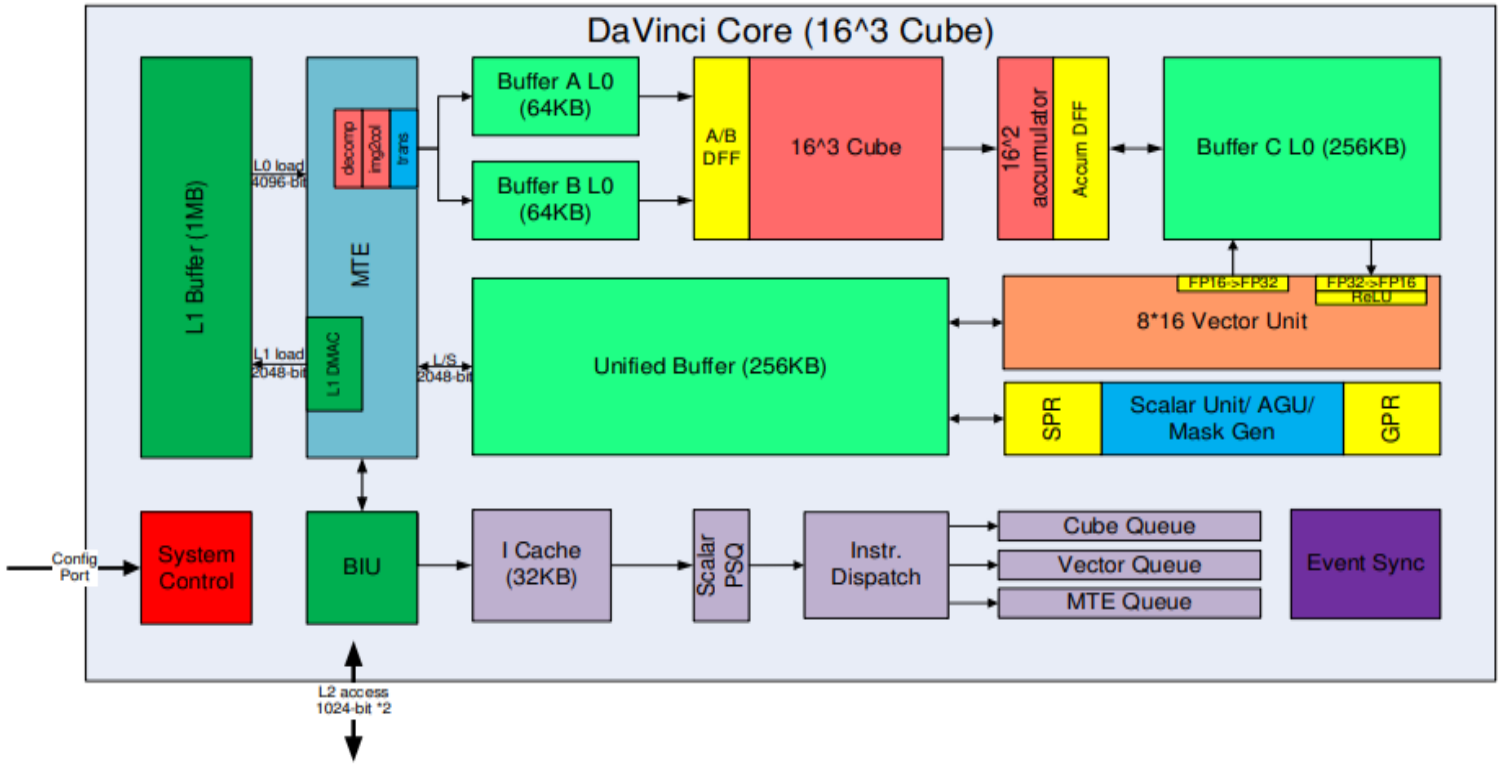




HUAWEI

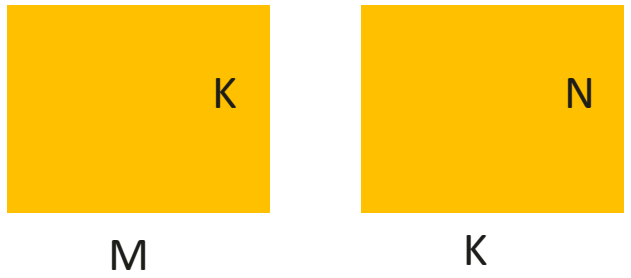
方法2：提高Cube的使用效率

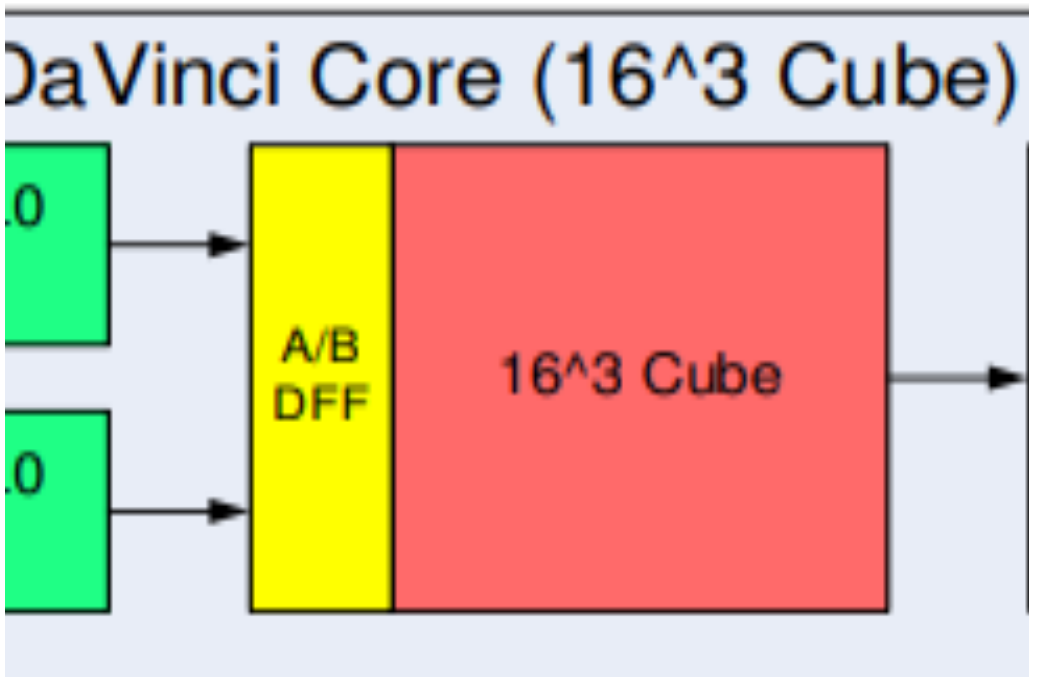
- 要想提升在昇腾310处理器上的性能，就要尽量提升cube的使用效率。



路径A：提升cube的数据利用效率

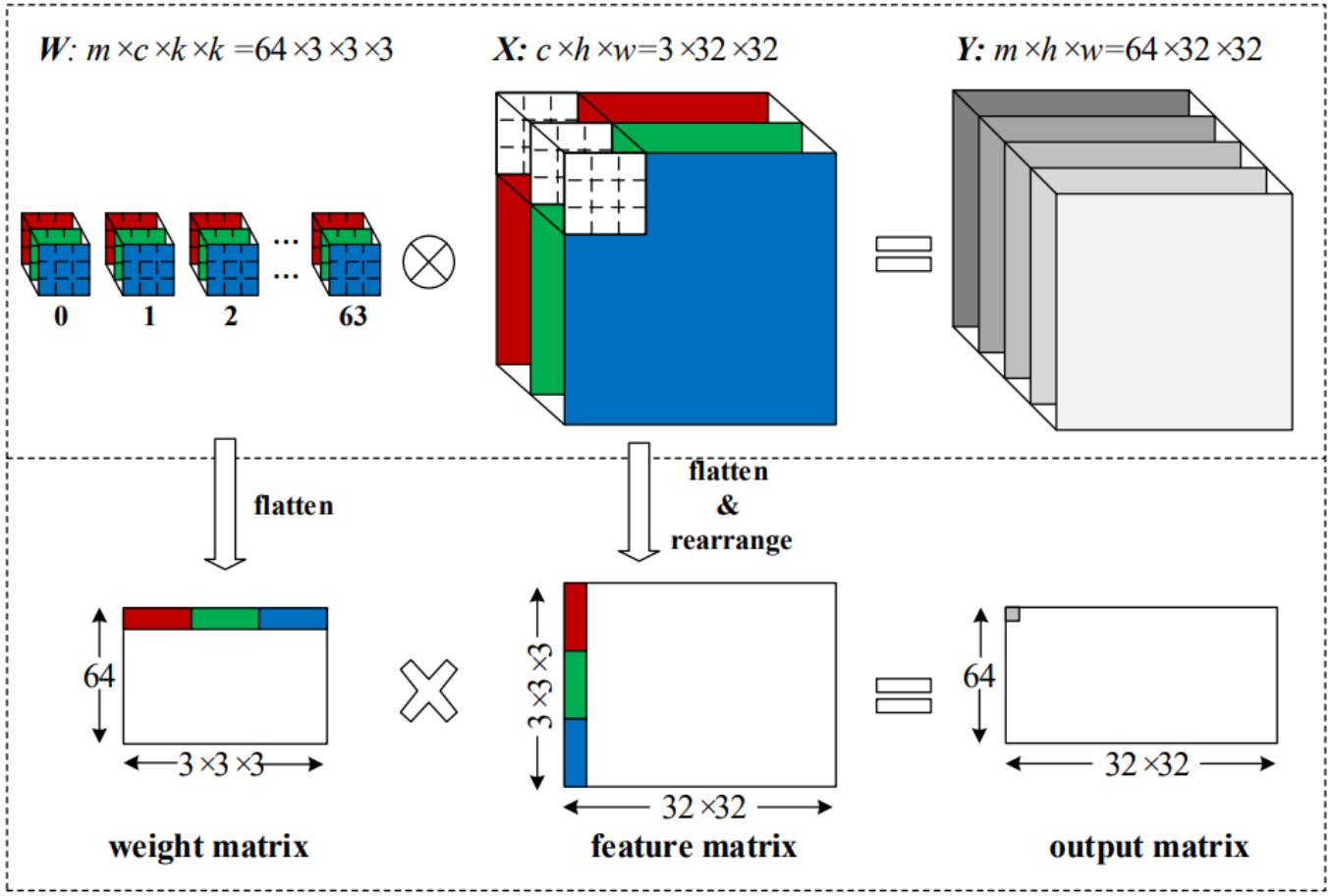
- 矩阵乘法的MKN，尽量取16的倍数。





- 思考：
- 如果想要卷积展开之后的矩阵乘法MKN为16的倍数，这个卷积应该是什么样子的？

卷积如何展开成矩阵乘法……



https://www.researchgate.net/publication/330315719_A_Uniform_Architecture_Design_for_Accelerating_2D_and_3D_CNNs_on_FPGAs/link/5c38962892851c22a36cd6db/download

案例：卷积在16*16*16的cube上执行

- fp16计算时，输入channel最好是16的倍数；
- Int8计算时，输入channel最好是32的倍数；

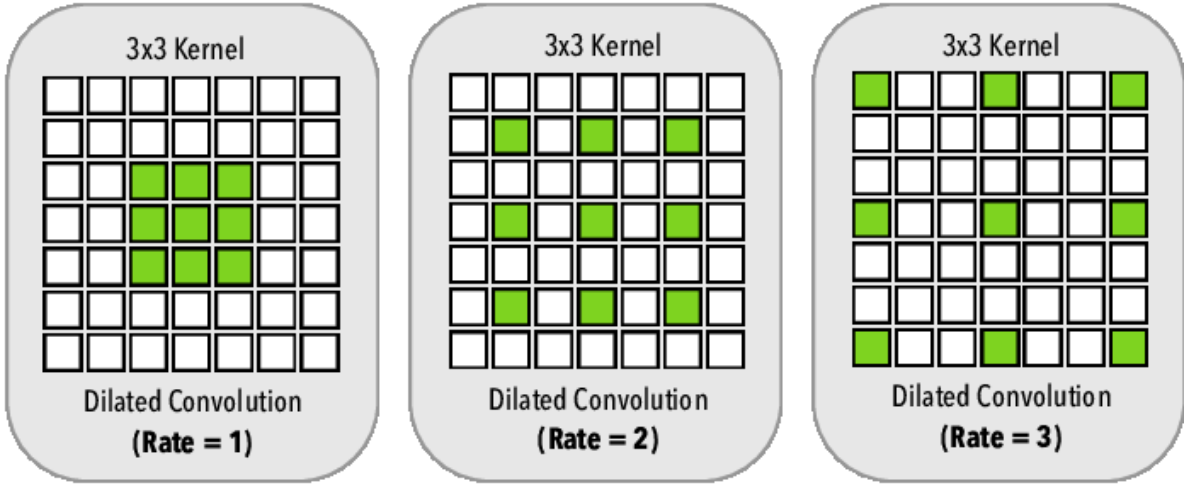
- 思考：

- DepthwiseConvolution效率高吗？为什么？

- V100的TensorCore是4*4*4，有什么好处？有什么坏处？

路径B： 增加数据复用率

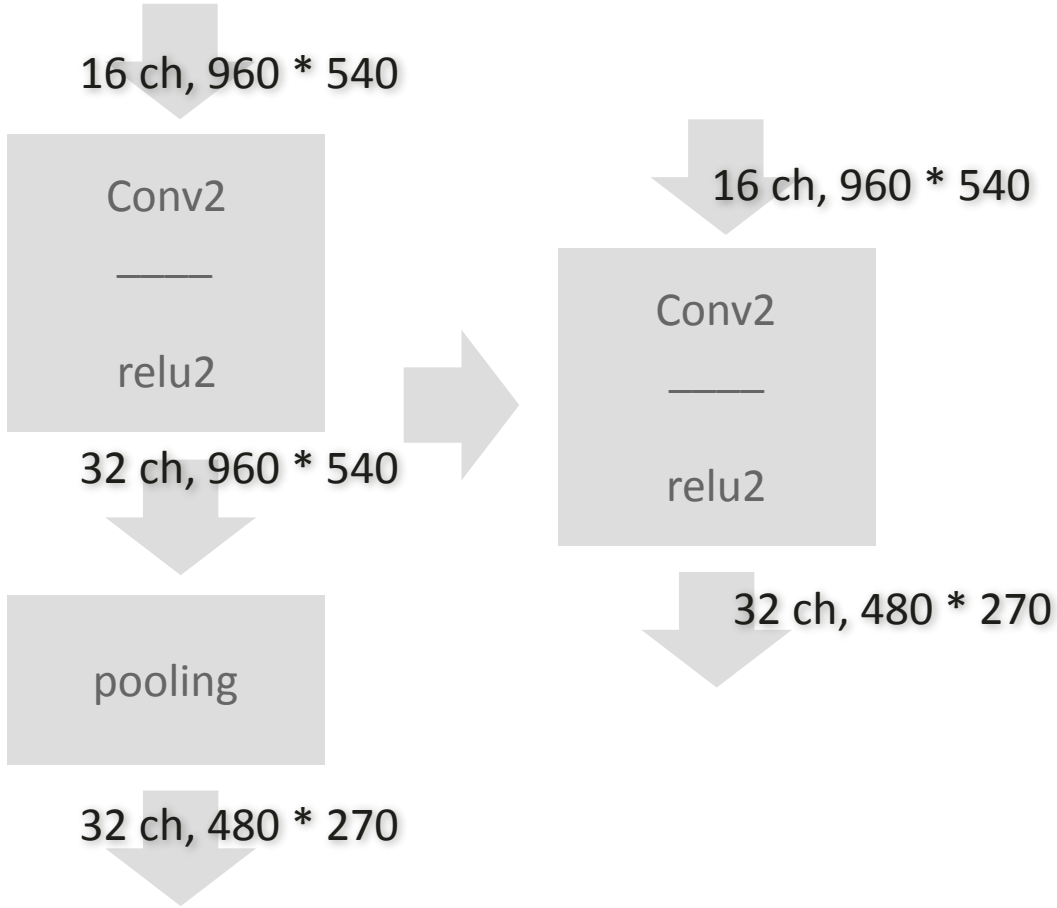
- 一个参数的利用次数越多（比如feature map很大的卷积）带宽的瓶颈越小，反之（比如不带batch的全连接，一个参数只用一次）则带宽瓶颈越大。
- 所以算法上可以考虑增加filter的复用次数，例如
  - > 增加feature map大小
  - > 避免过大的stride
  - > 避免过大的dilation
  - > 增加batchsize



路径C：算法模型优化

- 比如：
- Pooling层用stride=2的convolution层替代
- 1. 减少vector bound；
- 2. 减少convolution层开销；

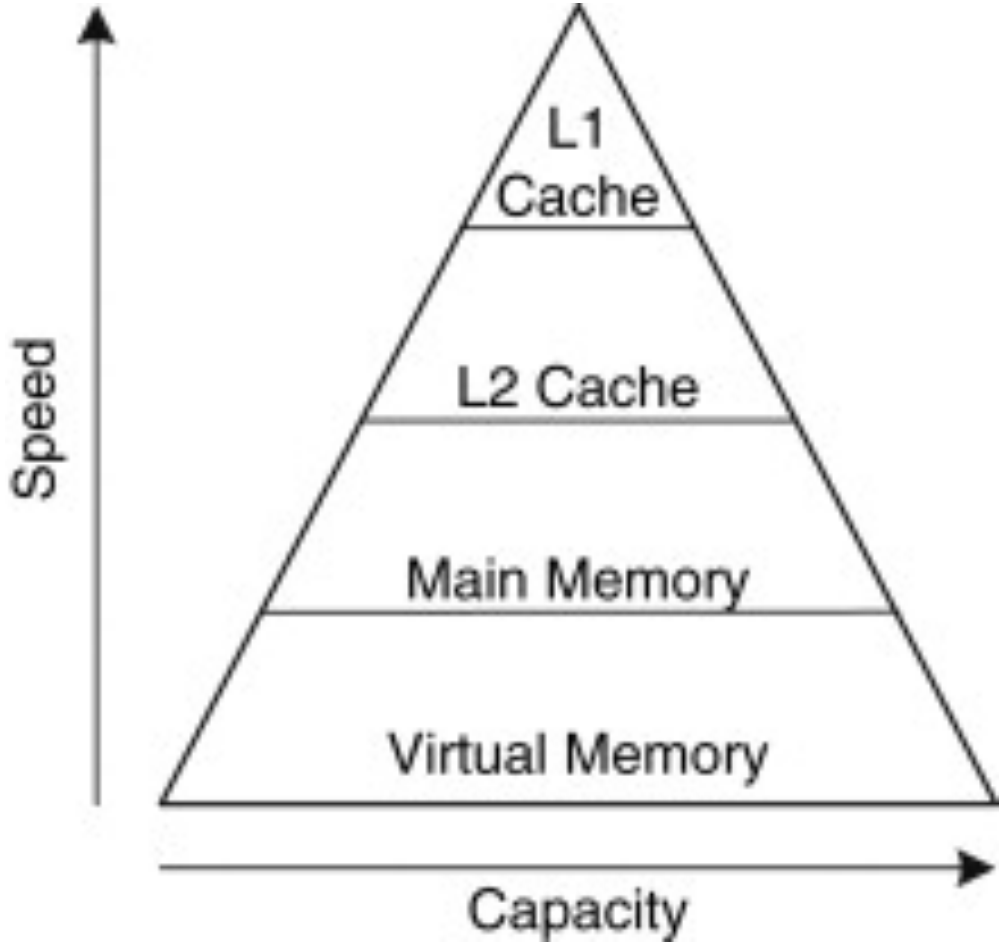
- Pooling层的作用：
- 1. 减少计算量；
- 2. 强化网络的不变性；



路径D：L2 Buffer = 8M

- 这意味着，如果feature map加上weights的大小超过8M，那么数据会被存放在DDR上。

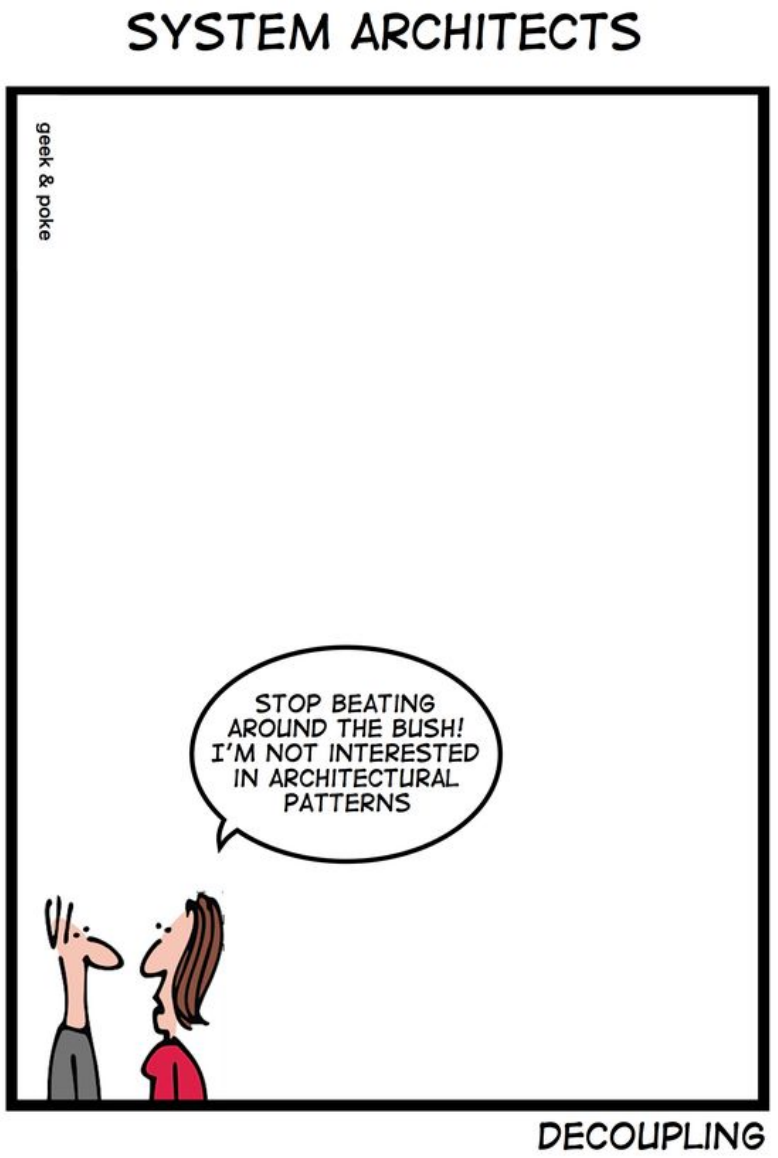
- 例如，假设FC层输入维度是1024，输出维度也是1024，则权值矩阵的大小是2048*2048*2Btyp=8M，加上feature map，无法在L2放下。数据就会被放在外部。



本节小结：“硬” “凑” “紧” “裁”

- 一要算得快
- 二要供得上

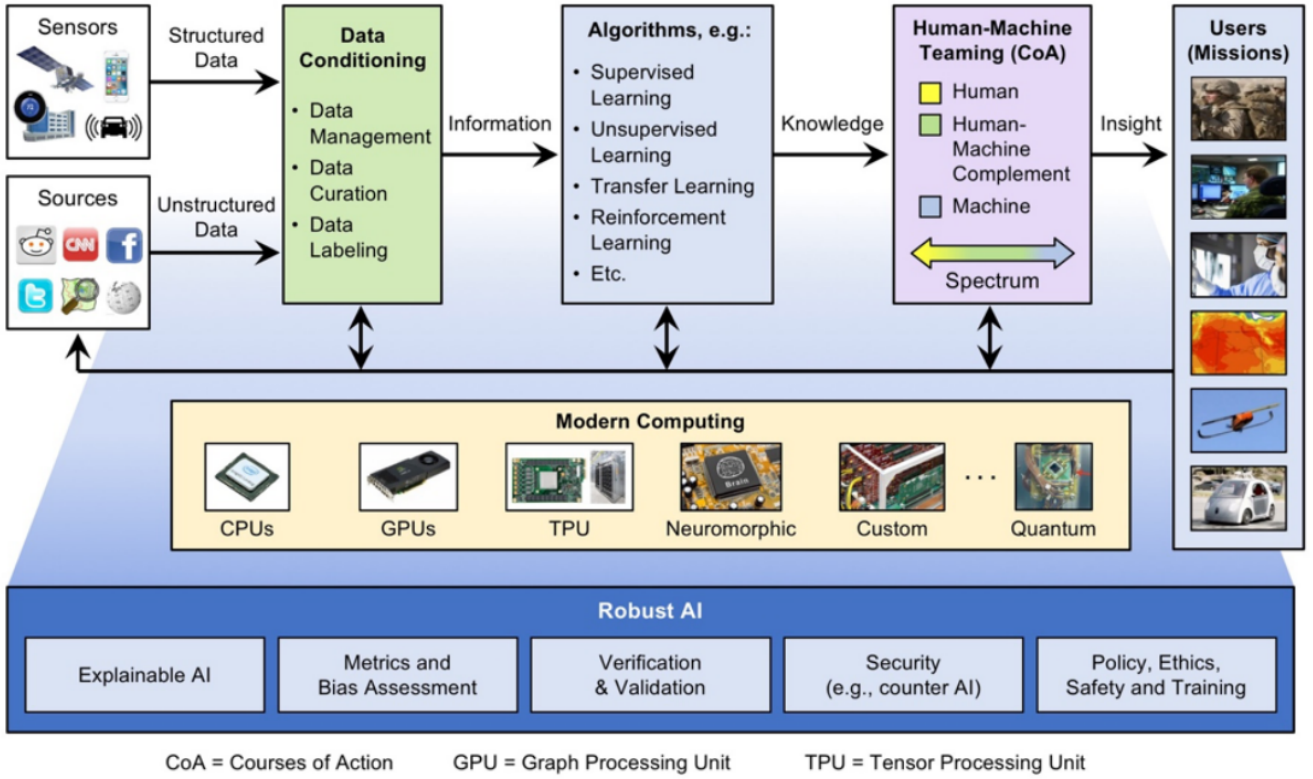
- 思考：
- 组网策略与模型并行/数据并行的关系

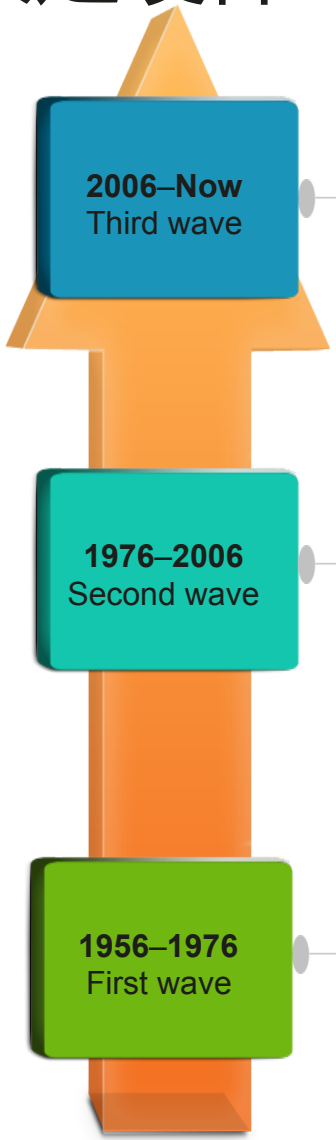


目录

- 深度学习网络性能调优实践
- 计算框架
  - 计算框架设计
  - 图优化编码实战
- 实战技巧
  - “硬”
  - “凑”
  - “紧”
  - “裁”

思考：是硬件决定算法？还是算法决定硬件？





Deep Learning + Cloud Computing & Big Data

- Progress of convolutional neural network model and parameter training skills
- Hardware progress: Moore's law, considerable computing power, cloud computing
- With the Internet + massive data sets, the accuracy of deep learning increases with the data volume.

Knowledge Engineering + Machine Learning

- Knowledge engineering is embodied in various expert systems, such as medicine and engineering. It is difficult to summarize and obtain knowledge. Some experts are reluctant to share experience.
- AI is introduced into various machine learning algorithms, so that machines can automatically learn from data and obtain knowledge.

Symbolism + Logic Methods

- Logic inference method based on decision theory.
- knowledge inference by means of machine proof, such as proving mathematical theorem through logic programming language.
- Symbolism is introduced in statistical methods for semantic processing to realize human-machine interaction.

https://arxiv.org/pdf/1908.11348.pdf

HUAWEI

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI