卷积算子优化

1 赛题描述

卷积神经网络(Convolutional Neural Network, CNN)是一类常用的深度学习模型，在计算机视觉、自然语言处理等领域广泛应用。卷积算子是 CNN 的重要组成部分，用于提取图像、文本等数据中的特征。

卷积运算是深度学习中常用的操作之一，但是由于其计算量较大，在大型神经网络中可能成为性能瓶颈，导致训练时间过长。因此，为了提高卷积运算的计算效率，需要通过并行编程等技术手段进行优化。

本比赛，要求参赛者优化卷积算子的并行编程实现。具体而言，给定一个输入张量和一个卷积核张量，要求实现一个高效的卷积运算，并输出卷积结果。卷积（Conv）算子的 DCU 加速优化，希望借助该赛题能够帮助更多选手了解 DCU 加速卡和 DCU 加速编程。

本赛题中，卷积（Conv）算子的形式如下：

$$C = A \ @ \ B$$

@表示卷积核 B 在输入 A 上的卷积运算过程；其中 A 为卷积输入张量、B 为卷积核张量；C 为卷积输出张量；其中卷积输入张量的维度信息为（n，c，h，w）；卷积核张量的维度信息为（k，c，r，s）；卷积输出结果的张量维度信息为：（n，k，oh，ow）；假设卷积核在 h、w 方向上的滑动的步长为 u、v；在 h、w 方向上的补边分别为 p、q；则：

对于卷积运算的三个张量 A、B、C 来说，其各个维度关系有如下关系：

1、A(n)=C(n)，即输入张量的 batchSize 与输出张量的 batchsize 一致；

2、A(c)=B(c)，即输入张量的通道数与卷积核张量的通道数必须一致；

3、B(k)=C(k)，即输出张量的通道数，是由卷积核的个数（k），来决定的；

4、oh=(h-r+2*p)/u + 1; ow=(w-s+2*q)/v + 1;

5、输出张量内的任一点的数值计算方式为卷积核与对应的输入位置数据内积，并与所有卷积核通道 c 上的对应位置累加而得来的；

本次比赛，采用异构编程加速上述卷积算法，要求在异构加速卡上实现相关算法并进行优化。输入张量 A、卷积核张量 B 以及输出张量 C 均采用通道交错（NCHW）存储格式。重点关注单精度浮点数在异构计算平台上的计算性能。所有的张量内的数值统一采用单精度浮点数。

二、评测说明

参赛队需要在比赛的截止日期前一周完成提交。需遵循如下赛题要求：

1）参赛队在给定接口函数的基础上进行卷积性能优化，给定测试代码中，示例代码中给出以伪随机数生成张量作为测试数据，验证算法的性能与正确性，选手可以适当调整 main.cpp 内的相关打印语句，来调试程序，但最终提交的代码中，需要与原始代码一致。Main 函数入参说明，参见附录。

2）参赛队需根据给出的 demo 程序对 conv2d.cpp 中的函数体进行具体实现与优化。选手可以根据各自核函数入参需求，增改 mykernelParamType 结构体内的成员，但需要注意：conv2d.cpp 内各个函数接口请不要改动，以防止 main 函数内无法正常调用选手优化的 kernel。

3）算法实现中如需分配临时存储空间，需要在函数内正确的处理空间分配与释放。参数选手需选取合理的算法，避免占用过多显存。

4）本次比赛输入张量 A、卷积核张量 B 以及输出张量 C 均采用通道交错（NCHW）存储格式；选手实现的相关 kernel，必须是基于该存储格式；否则，验证程序无法验证通过；

5）参赛队不能调用第三方库的实现来加速自己的 kernel 计算。

6）禁止修改 main.cpp 和 verify.h、conv2d.h 等辅助与记时函数，以及相关接口；仅需在 conv2d.cpp 内做优化，保证程序正确编译运行。

7)参数队需提供完整的源码、供赛事组进行评测。尽量保证只修改 demo 中的 conv2d.cpp 文件；源码编译方式采用 demo 中给定的 Makefile 编译参数，不得修改。

8）移植过程中限制使用一块类 GPU 卡+8CPU 核心，测试使用 numactl 限制在一个 NUMA node 上运行。

9）参赛队比赛期间最多可提交 3 次结果。赛事组以最后提交的结果为准。

10）不得抄袭其他参赛队的答案，一经发现成绩作废。

三、评分标准

在计算结果正确无误的情况下，评测主要以算例的运行时间为主要评测依据，总分 100 分，评测数据集与分数占比如下：

（1） 测试数据集

本次比赛采用常用网络中挑选出的几组相关卷积 size，具体参数如下表：

算例	n	c	h	w	k	r	s	u	v	p	q
1	128	3	225	225	32	3	3	2	2	0	0
2	49	128	35	35	384	3	3	2	2	0	0
3	16	128	105	105	256	3	3	2	2	0	0
4	128	3	230	230	64	7	7	2	2	0	0
5	2	3	838	1350	64	7	7	2	2	0	0
6	256	256	28	28	256	2	2	2	2	0	0

(2)　评测算例稀疏度选择及分数占比

评测算例矩阵 A 采用两个数据集，每个数据集对应的矩阵 B 的稀疏度有三种组合，共 6 种组合，算例参数及分数占比见下表。

算例	n	C	h	w	k	r	s	u	v	p	q	评分指标	分数占比
1	128	3	225	225	32	3	3	2	2	0	0	计算耗时	20%
2	49	128	35	35	384	3	3	2	2	0	0	计算耗时	20%
3	16	128	105	105	256	3	3	2	2	0	0	计算耗时	20%
4	128	3	230	230	64	7	7	2	2	0	0	计算耗时	20%
5	2	3	838	1350	64	7	7	2	2	0	0	计算耗时	10%
6	256	256	28	28	256	2	2	2	2	0	0	计算耗时	10%

(3) 正确性：　选手需保证所有算例都计算正确，demo 程序 check 通过。

(4) 计算耗时：　在计算正确基础上，测试每个算例运行时间。测试使用相同的节点，每个算例分别运行 100 次（排除第一轮 WARMUP 时间）并求平均计算时间。

(5) 评分算法：每个算例单一统计耗时，最短耗时除以各选手耗时，100 分满分；所有 6 组分别统计，最后按照单个算例占比，计算总分（100 分）。

附录

参赛队报名后请下载附件中的 demo 测试程序。demo 测试程序中已经为选手规范了主函数入参顺序，为方便各位选手更好的进行优化工作，现将主函数函数参数解释整理如下：

```
int main(int argc, char**argv)
{
    int n = atoi(argv[1]);
    int c = atoi(argv[2]);
    int h = atoi(argv[3]);
```

```
        int w = atoi(argv[4]);
        int k = atoi(argv[5]);
        int r = atoi(argv[6]);
        int s = atoi(argv[7]);
        int u = atoi(argv[8]);
        int v = atoi(argv[9]);
        int p = atoi(argv[10]);
        int q = atoi(argv[11]);

        …
    }
```

1）主函数输入参数，按顺序如下：

argv[1]：输入张量的 batchsize，对应 A(n)维度；

argv[2]：输入张量的通道数，对应 A(c)维度；

argv[3]：输入张量的高度，对应 A(h)维度；

argv[4]：输入张量的宽度，对应 A(w)维度；

argv[5]：卷积核张量的数量，对应 B(k)维度；

argv[6]：卷积核张量的高，对应 B(r)维度；

argv[7]：卷积核张量的宽，对应 B(s)维度；

argv[8]：卷积核在高方向上的滑动步长，对应步长 u；

argv[9]：卷积核在宽方向上的滑动步长，对应步长 v；

argv[10]：卷积核在高方向上的补边，对应补边 p；

argv[11]：卷积核在宽方向上的补边，对应补边 q；