week3实验记录

zxp

March 16, 2024

1 environment

cpu:Inter i5-12400f (2.5 GHz)

System:Ubuntu 22.04.1

Compiler:gcc 12.3

2 code

增加了im2col变换+矩阵乘法+将矩阵乘法的结果转换成output数据布局来实现卷积的代码。

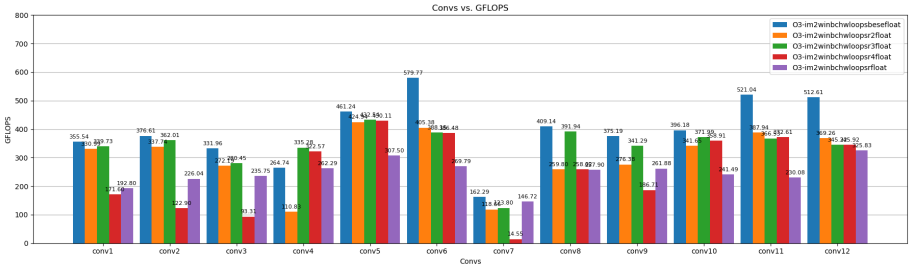

Figure 1: O3

3 Experiment

这个实验是为了测试循环顺序产生的影响。
im2win卷积为了实现前几周使用的优化和出于对读取的用于进行运算的数据连续性的考虑，最里面两层是固定的依次是过滤器的宽和高。然后为了减少读取output张量所花费的时间，倒数第三层固定为过滤器的通道，通过这三层可以计算出一个output的元素而不需要读取output元素。
可改变循环顺序的有output的批次通道（b）和通道（c）和高（h）和宽（w），选择在前几个星期所做了优化的版本（对通道这个纬度进行了并行和对output的宽进行了分块）上面进行循环顺序的改变。
下面是结果，有基准版本循环顺序为bchw（蓝色柱子），bcwh（橙色柱子），bhcw（绿色柱子），bhwc（红色柱子），cbhw（紫色柱子）

3.1 Analysis

除了conv4，都是初始的循环顺序（bchw）更快，因为是对通道批次进行并行化，将通道这个纬度放在更外层（cbhw紫色柱子）历遍批次这个纬度的时候input的数据连续性不好。
将通道这个纬度放在更里面（bhcw绿色柱子，bhwc红色柱子），历遍高和宽的时候为了历遍通道这个纬度input的数据连续性也不好，特别是在conv7，conv7output的宽高非常大，而通道相对宽高较小。

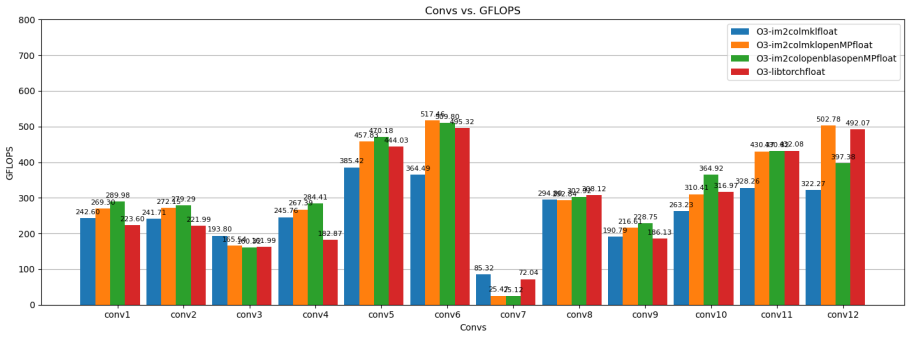

Figure 2: O3

4 Experiment2

将上周写的im2col变换加上矩阵乘法加上将结果转换成output数据布局的函数实现了显示的im2col卷积运算。
矩阵乘法使用了MKL和openblas的level3的矩阵乘矩阵函数。虽然可以将所有批次的张量转换成一个矩阵，但这样存储中间计算结果的数组会很大，也没办法手动加openMP，所以保留了历遍批次的循环，每次历遍使用一次矩阵乘法算出一个批次的output。
测试的时候测试了自己在代码中加并行使用串行版本的线性代数库和不在代码中加并行使用并行版本的线性代数库。下图中，带名字openmp的就是自己在代码中加openmp（橙色和绿色柱子）。
MKL有自己的库来管理线程实现并行，链接MKL库的时候发现怎么都没法链接MKL库的线程库，看说明书官方是推荐使用英特尔自己的编译器，使用其他编译器推荐使用串行的MKL版本，也有解决方法是设置MKL使用gcc的线程库，我这周解决的方式就是设置MKL使用gcc的线程库。
使用openblas的时候发现编译的openblas并行版本有问题，虽然让cpu满载了，但速度好慢，gflops只有5左右，原因不明。

4.1 Analysis

性能表现libtorch（红色）差不多，大部分conv甚至要快一些。im2col需要大量的内存，不仅仅是im2col变换后的矩阵，还有存矩阵运算的结果也要很多空间。隐式的im2col矩阵好像可以节省存矩阵运算结果的空间，还有上周论文指出可以做优化靠复杂的索引映射使得通过卷积运算得到的不是output数据布局的张量，而直接是im2col变换后的矩阵，很复杂不知道具体怎么实现，可能还得改线性代数库中矩阵矩阵乘法。