week8实验记录

zxp

April 20, 2024

1 environment

cpu:Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz (使用时申请了56个核心并且使用独占指令)

gpu:rtx3090(使用时申请了一块)

System:CentOS7

Compiler:9.5

2 code

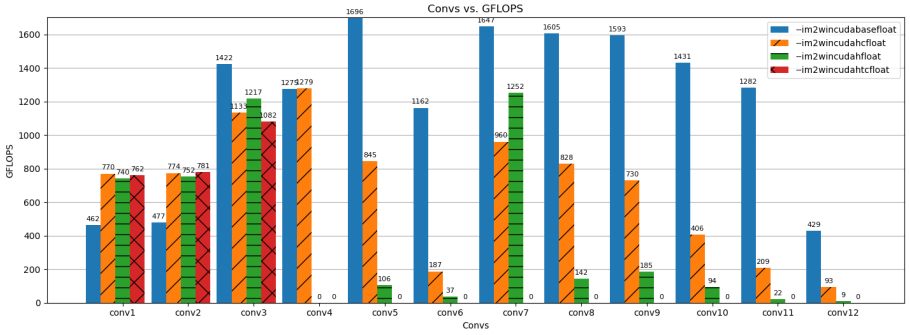

Figure 1: GPU

3 Experiment

之前在GPU上实现了im2win（蓝色柱子），这周尝试对im2win进行优化。因为GPU的特性，数据从全局内存会花费大量的时间，GPU中有可控的高速内存共享内存，共享内存在线程块中是共用可以被线程块中的每个线程访问，卷积运算中有大量数据是需要重复使用的，先将数据从全局内存搬运到共享内存就可以节省从全局内存读取重复数据的时间。
但GPU中的共享内存十分珍贵，并且不同硬件不同，英伟达是按计算能力给的（近几年的GPU的共享内存至少大于48kb），在3090中共享内存只有100kb。如果在没优化的卷积运算中直接用共享内存，每个线程块算1024个ouput的元素所需要的数据在共享内存中是肯定放不下，于是我将每个线程块改成只算一行的output元素（绿色柱子），因为每行中复用的元素是最多的，这样做共享内存能放下需要计算的数据（除了conv4），但是这样每个线程块中在工作的线程就小了很多闲置的线程导致性能下降了很多（特别是conv5这样每个线程块只有5个线程在运行）。因为这样分conv4也还是放不下，我按input的通道分的更细，线程块改成只算一行的output元素中的一部分，因为是并行最后还需要原子加将数据加到output中，这样要用到的数据是能塞进共享内存中，但绝大部conv性能下降的特别严重（橙色柱子）。为了补救，让更多线程用起来，在线程块中将计算按input的通道大小分开（红色柱子），又出现新问题，分出的线程大于1024，只有conv1-3能运行，其他的conv都不行，性能也很差。性能下降太多，这周的写法肯定是不对的。（图中的GFLOPS未计算数据从内存搬运到显存的时间，这段时间开销挺大的，和没优化的im2win卷积运算花的时间55开）

3.1 Analysis

数据搬运需要花费大量时间，做的优化重点都是让数据搬运花的时间少点，不必要的计算少点。共享内存能缩短总的时间搬运时间，共享内存肯定是重要的。但因为共享内存大小有限，怎么用好共享内存怎么给数据分块真的好难。即使这周共享内存用的很差劲，conv1和conv2性能还是翻倍了，说明共享内存

有用，接下来是更合理的利用让其他conv效果也好起来。

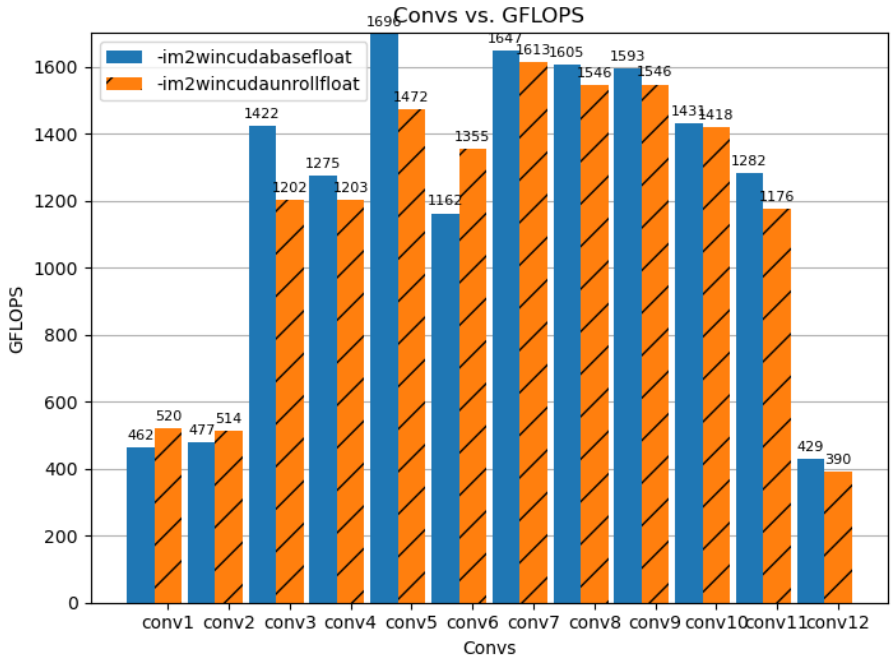

Figure 2: GPU

4 Experiment2

还尝试了unroll，unroll可以让核心少做判断不断地执行计算，特别是GPU上的核心比CPU上处理能力弱的情况下更加重要，GPU上使用unroll在循环外面加上#pragma unroll编译器就会自动将循环全部展开，这周尝试了在最朴素的im2win卷积上每个循环加了#pragma unroll。

4.1 Analysis

性能反而下降了，不过是在最朴素版本上加的，后面在优化后的版本上加试试