week1实验记录

zxp

March 2, 2024

1 environment

cpu:Inter i5-12400f (2.5 GHz)

System:Ubuntu 22.04.1

Compiler:gcc 12.3

2 code

把代码整理了，把主要的代码分类后放在了同文件里面，详细见github。
这周是测试了unroll和分块，我优化的是将通道展开的那个数据结构，模仿马吉祥对直接卷积做的优化写的，对每个conv的unroll和分块不一样。

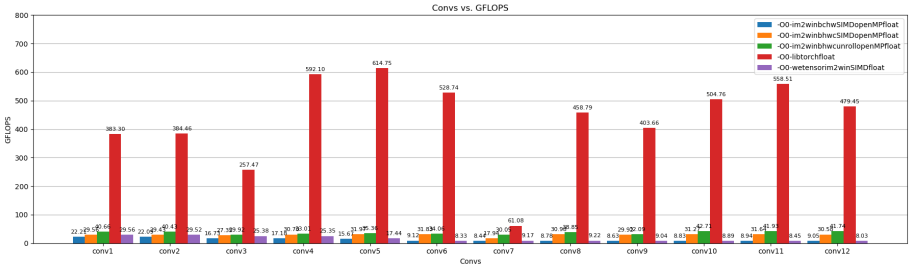

Figure 1: O0

3 Experiment

为了有个参考，将libtorch里面的conv2D卷积方法和github上面wetensor的优化后的im2win卷积测试了一下，开启了openMP，batch设置为48（（因为机器是6核12线程的）。
然后测试了unroll和分块的效果。

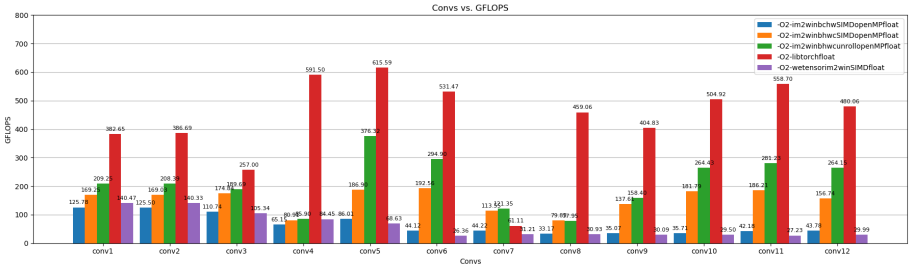

Figure 2: O2

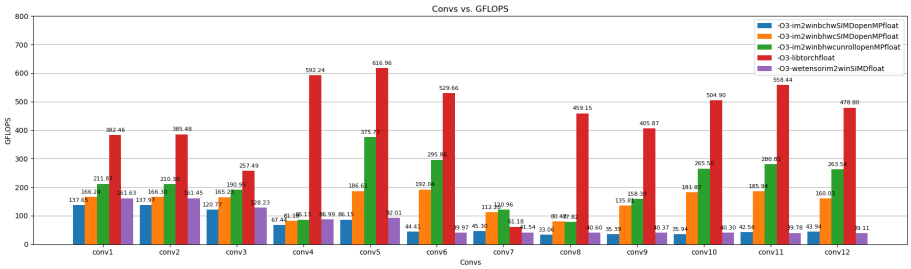

Figure 3: O3

3.1 Analysis

比起只用SIMD，使用了unroll和分块后是要更快的，但libtorch太快了，除了conv7，在其他的conv比libtorch慢很多，libtorch普遍能到400或者500gflops但tensor1D普遍只有200，libtorch最快的conv5甚至能到600，达到roofline（800）的75%。wetenor的unroll逻辑是能先按64unroll剩下的按32unroll然后是16、8直到分完，只做这样的优化wetensor最慢。
也粗略的测试了不将通道全部展开的那个数据bchw（（虽然马吉祥的优化中在部分conv上也对通道展开，但没全部，只放了8个进去），马吉祥的那个要快点，但大部分的gflops也才300，还是比libtorch慢，bhwc这个数据结构unroll更慢可能是把全部通道展开让计算一个输出元素用到数据太多缓存塞不下。