week-2

JX-Ma

2024/3/9

1 本周工作

1. 把 im2win 变换的图画出来了。2. 修改了 256 位寄存器使用的数量，确保这些寄存器的数量都小于 16，因为我的 cpu 最大寄存器的数量是 16。3. 尝试更换 openmp 的位置，刚开始测试加上 2 层 Openmp 发现结果对不上，后面发现是指针定义的地方在 openmp 外导致数据错误，修改指针位置后尝试了在 batch+channel,channel+height,channel+width 上面加上 openmp，并把他们内存分析记录下来，还测试了单个 openmp 在 batch,channel,height 上的表现，没有单独测试 width 上增加 openmp 的情况，因为有的 benchmark 输出张量宽度很小，在对输出张量的宽分块后一次卷积就能得到一行的结果。

2 im2win

这里我们假设卷积核宽 x 高为 4x4，步长为 1，输出张量分块大小 4x4。

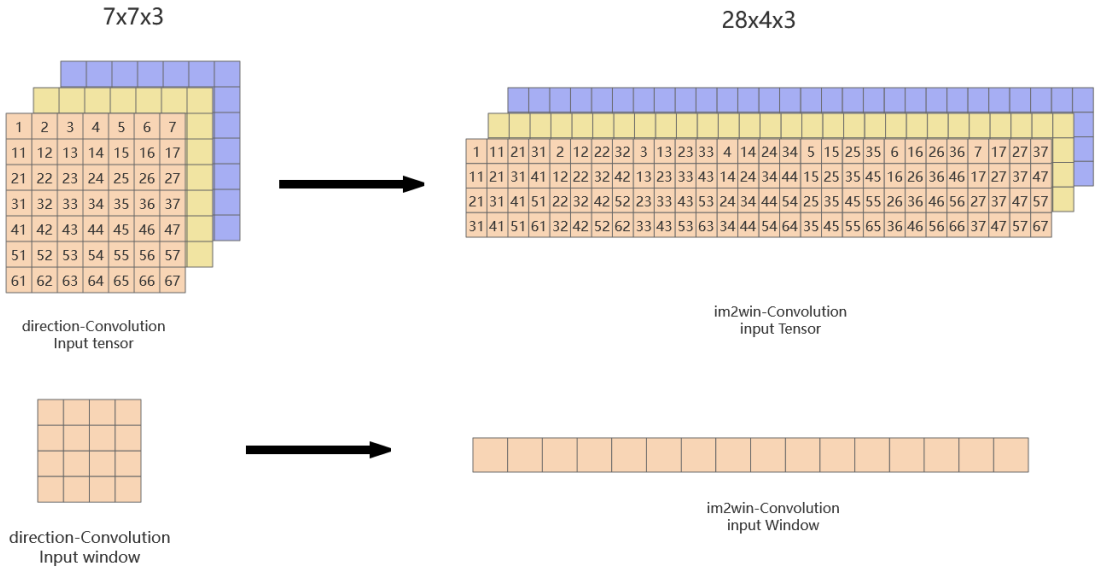

图 1: im2win

3 实验部分

3.1 实验环境

- 系统: Ubantu 22.01
- gcc version : 9.5.0
- 优化选项: -O3 -fopenmp -avx2 -fmadd
- cpu:AMD Ryzen 7 6800H 3.20GHz

3.2 实验结果

实验结果展示了在不同 openmp 组合下的 gflops 结果

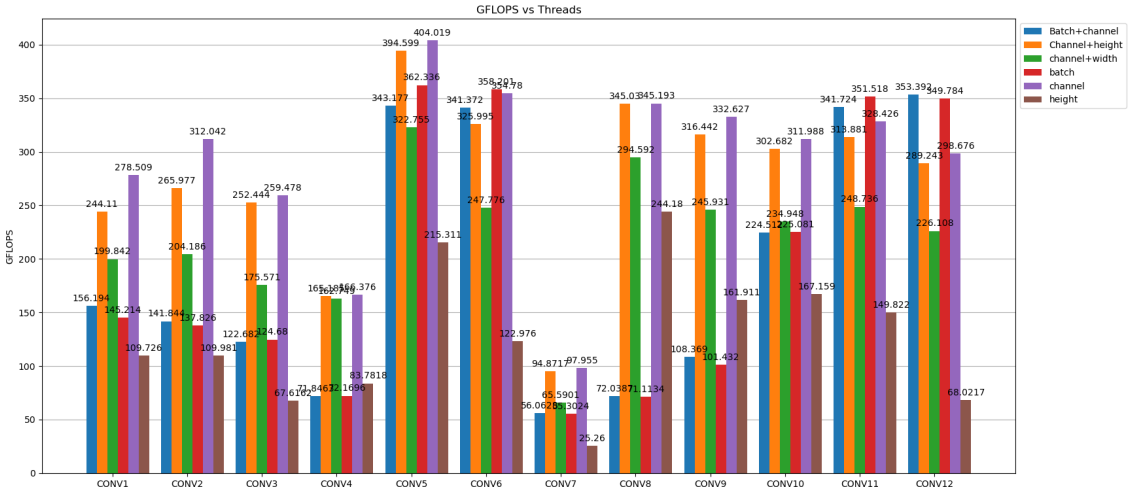

图 2: gflops

4 实验结果和分析

从实验中可以看出，大部分 benchmark 在 channel 维度上加上 openmp 效果最好，但是有的 benchmark 如 6,11,12 在 batch 上面加上 openmp 上比较好，加上两层 openmp 的效果除了在 channel+height，其他表现结果都不行。在看实验结果的时候，对同一个 benchmark 测试多次时发现他们数据波动很大，最小的有 70，最大的时候有 425，所以每个数据取的是中位数。在对内存分析的时候发现对于性能的好坏不能只看 pagefault, 还需要关注 context switches.

- Major (requiring I/O) page faults: 主要页面错误, 主要页面错误通常发生在进程访问虚拟内存中的数据时，但物理内存中不存在该数据的情况下。这可能是因为数据被交换到磁盘上，需要进行

I/O 操作才能将数据加载到内存中。主要页面错误会导致进程暂停执行，直到数据被加载到内存中为止。

- Minor (reclaiming a frame) page faults: 次要页面错误, 次要页面错误通常发生在进程访问虚拟内存中的数据时，数据虽然不在物理内存中，但在其他地方（如磁盘交换区）存在。当数据被重新加载到内存中时，这种错误会发生。次要页面错误不需要 I/O 操作，因为数据在其他地方可用。
- Voluntary Context Switches, 自愿上下文切换, 自愿上下文切换是由进程自身触发的，通常发生在进程主动放弃 CPU 执行权或等待某些资源时。例如，进程主动放弃 CPU 执行权给其他进程，或者进程等待某个事件完成。
- Involuntary Context Switches, 非自愿上下文切换, 非自愿上下文切换是由操作系统强制进行的，通常发生在有更高优先级的进程需要执行、当前进程时间片用尽或发生硬件中断等情况下。操作系统会强制将 CPU 执行权从当前进程转移到其他进程，造成非自愿上下文切换。