week-21

JX-Ma

2024/2/3

1 本周工作

继续找优化直接卷积的方法，本周进行了 3 种方法，分别针对 CONV1,CONV5,CONV7, 进行了测试，其中 CONV1 和 CONV7 的优化方法不太行，但是 CONV5 的方法相比之前的有很大的提升，CONV5 在串行的卷积 GFLOPS 可以达到 94。

2 实验部分

2.1 实验环境

- 系统: Ubantu 22.01
- gcc version : 9.5.0
- 优化选项: -O3 -fsse -favx2 -fmadd
- cpu:AMD Ryzen 7 6800H 3.20GHz

2.2 CONV1 的优化

- input: 10 3 227 227
- filter: 96 3 11 11
- output: 10 96 55 55
- stride: 4

针对 CONV1 的优化选择使用了多级指针去存储输入张量不同维度，高度，宽度的地址，具体如下。选择这样做的原因是：输入张量中不同高度的地址不连续，我们采用一个连续的地址去存储这些不连续的地址，但是效果并不太好，gflops 大概到了 10 左右，后面就没有继续做下去。

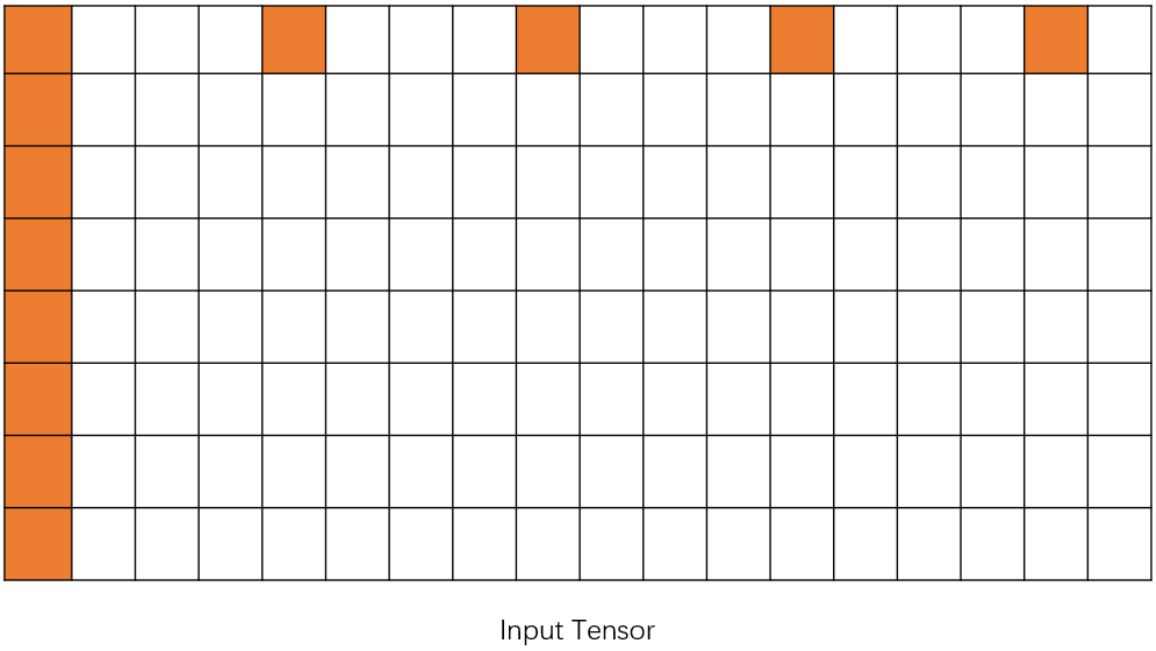

图 1: CONV1

刚开始定义一个地址指针存储输入张量行间隔为 4 元素的地址，后面再存储每一行的地址，最后在存储每个通道的地址。

2.3 CONV7 的优化

- input: 10 3 224 224
- filter: 64 3 3 3
- output: 10 64 222 222
- stride: 1

这里数据布局改变也可以理解只是将张量展开了，本来想着按照通道这一维度展开，但是看了一下 CONV7 的通道为 3，不太好使用 AVX2，所以选择展开了它的 batch，如下图所示，因为也是负优化，所以大概说下数据布局。

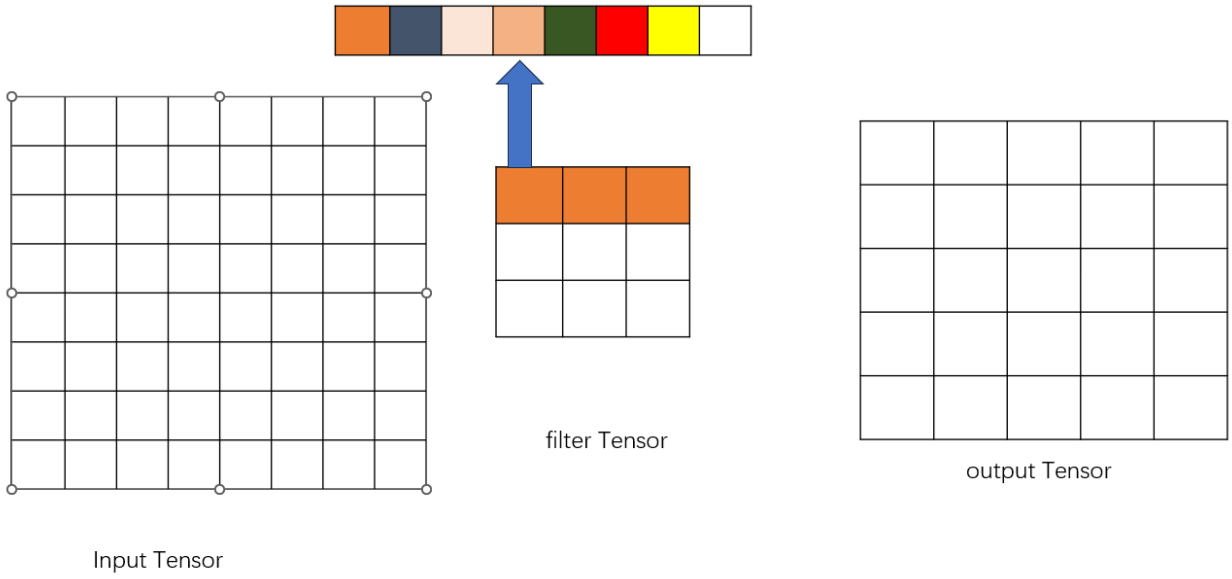

图 2: CONV7

这里改变的仅仅是卷积核的数据布局，其他的不变。

2.4 CONV5 的优化

- input: 10 96 24 24
- filter: 256 96 5 5
- output: 10 256 20 20
- stride: 1

数据布局改变为将 input,filter 按照通道 1x8 平铺，具体如下:

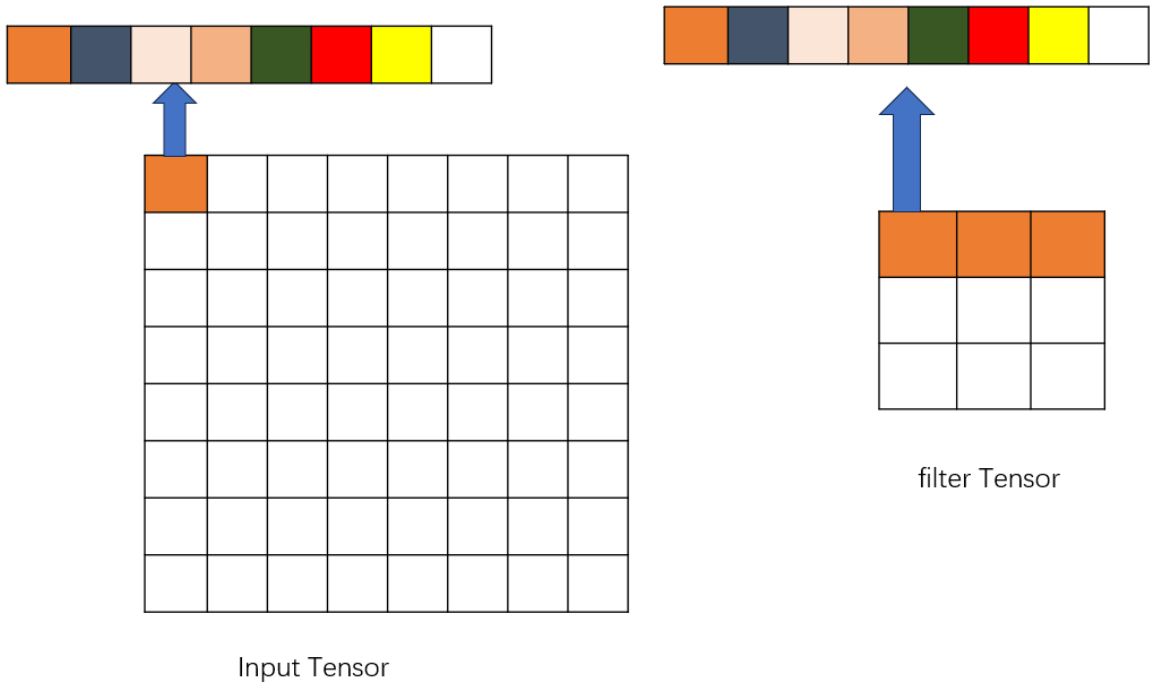

图 3: CONV5

这里改变是卷积核和输入张量的数据布局，其他的不变。

之后分别对这样的布局进行了 unroll 和寄存器分块操作，结果如下图

可以看出在数据布局变化后，优先使用寄存器分块对性能的提升较大。

在做完这个实验后，我尝试对布局进一步展开，按照通道 1x16 展开，但是效果没有按照 1x8 展开好，按照通道 1x16 展开是按照 1x4 分块的 gflops 只有 40 多一点，而 1x8 可以到 50 多，所以后面就没有继续分下去。

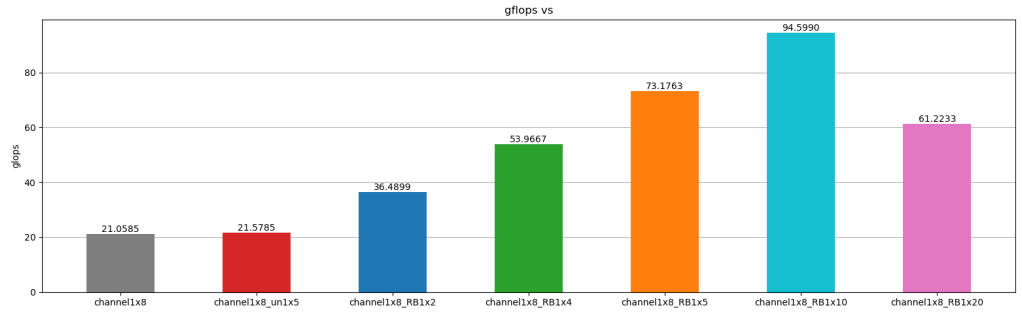

图 4: glops vs

2.5 Openmp 下 CONV5 优化

针对输出张量宽度分块，我测试按照 4,5,10，20 分块下，线程数为 1 2 4 8 16 的 gflops，随着线程的增加，我想应的改变了一下输入张量的 batch，以免卷积速度过快造成的误差。结果如下

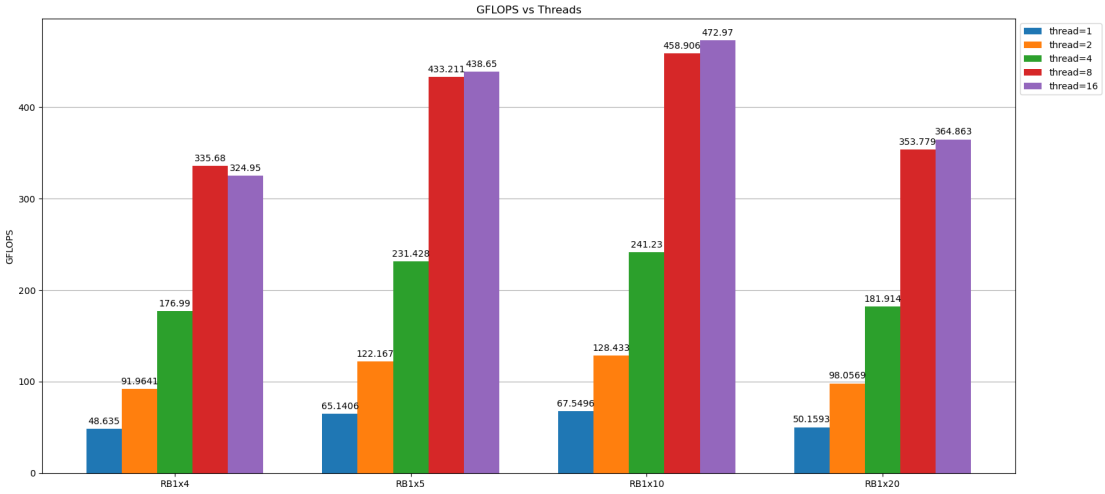

图 5: openmp glops vs

从图中可以看出，首先在设置线程为 1 的时候，卷积的性能都要小于串行的性能，线程数从 1,2，4,8 的 gflops 增长幅度很大，但是从 8-16 这里增长的就不大，甚至有一个还下降了。这里猜测可能是带宽的问题。

3 总结

这周的任务就是继续优化直接卷积，只有 CONV5 的卷积优化做的比较好，这也是理想情况下的卷积速度，因为别的 benchmark 的维度都并不好，这样会造成很多的计算浪费，这里的浪费主要是为了使用 simd 去填充张量，从而造成的计算浪费，而 CONV5 的卷积没有多余的计算浪费。还有一个就是

openmp 线程数从 8-16 这里就提不上去了，这里还需要一点改进。