Lab3 CUDA

实验环境

lab4提供的slurm平台

优化思路

Shared memory

将数组a中的数据存入共享内存，我选择开了一个shareA[in_channel][block_size+kernel-1][block_size+kernel-1]大小的三维数组，将一个block中需要和卷积核卷积计算的数值全部存入shareA，减少访问数组的开销；当时也尝试过将数组w的值存入共享内存，但出现了时间变慢的情况，所以最后选择了指将shareA放入共享内存

另外为了避免重复赋值，我采用了多重的判断条件，避免数组a中的数值赋给shareA时出现重复赋值的情况；当不需要考虑padding时，直接赋值即可，当需要考虑padding时再通过if判断是否越界进行赋值；

划分block

为了减去判断线程是否大于size的判断，我将block_size设成了10；之前尝试过4，8，16等大小的block_size，但由于会出现未知的bug，所以最后选择了速度较优且无bug的10

Virtual Thread Split

当shareA开成shareA[block_size+kernel-1][block_size+kernel-1][in_channel]时，会出现bank conflict的情况，若干个线程同时访问同一个bank，并行性会严重下降，时间也会变成原本的4倍，所以我采用了shareA[in_channel][block_size+kernel-1][block_size+kernel-1]这种形式，避免了bank confict

Cache miss

baseline的代码中出现cache miss的情况很常见，我改变了循环的顺序，将in_channel的循环和out_channel的循环放在了合适的位置，尽可能地增加cache的命中率

实验结果

baseline

```
wh-summer@GPU06:~/CUDA$ nvcc conv_baseline.cu -o conv_baseline -O3 -cudart=share
d -Xcompiler -fopenmp
wh-summer@GPU06:~/CUDA$ ./conv_baseline
Generating input and kernel tensor...
Conv2d Cuda Kernel Start...
Verifying...
Conv2d CPU Kernel Start...
Checking Results...
Correct
5145.32 milliseconds
```

基准时间为5.145s

优化结果

```
wh-summer@GPU06:~/CUDA$ make run
nvcc conv.cu -o conv -O3 -cudart=shared -Xcompiler -fopenmp
./conv
Generating input and kernel tensor...
Conv2d Cuda Kernel Start...
Verifying...
Conv2d CPU Kernel Start...
Checking Results...
Correct
228.617 milliseconds
```

时长228.617ms，加速比22.51

```
wh-summer@GPU06:~/CUDA$ make run
nvcc conv.cu -o conv -O3 -cudart=shared -Xcompiler -fopenmp
./conv
Generating input and kernel tensor...
Conv2d Cuda Kernel Start...
Verifying...
Conv2d CPU Kernel Start...
Checking Results...
Correct
243.327 milliseconds
```

时长243.327ms，加速比21.15