# 中期作业报告

李孟霖 2311067

## 1 规约求和实现概述

在这个规约求和的实现中，首先将输入数组按 block 求和，返回每个 block 的规约求和结果构成的数组，然后在 CPU 端使用 C++ 的 `numeric` 库进行最终的求和。整个过程将使用 CUDA 事件来测量时间。

相关的核函数及其声明写在 `kernels.cu` 和 `kernels.cuh` 中。

### 1.1 第一版规约求和

这个版本中，设备端求和完全使用全局内存。下面是运行截图和 ncu 分析结果。

```
(tmux) flyingbucket@H3C-R5300-G5-A30:~/cuda/midterm$ ./global.out
--- global memory version ---
block size: 256
elapsed time : 4.177312
final sum on global memory : 100000000
```

图 1: 第一版规约求和运行截图

考虑使用共享内存来进行初步优化，使用 shuffle 指令来进一步加速 block 内的规约求和。

### 1.2 第二版规约求和

在第二版中，使用共享内存来进行 block 内的规约求和。每个 block 的线程将数据加载到共享内存中，但暂不使用 shuffle 指令。下面是运行截图和 ncu 分析结果。

```
(tmux) flyingbucket@H3C-R5300-G5-A30:~/cuda/midterm$ ./shared.out
--- shared memory version ---
block size: 256
elapsed time : 4.044064
final sum on global memory : 100000000
```

图 2: 第二版规约求和运行截图

在第三版中，使用了 shuffle 指令来进一步加速 block 内的规约求和。每个 block 的线程将数据加载到共享内存中，并使用 shuffle 指令进行规约求和。下面是运行截图和 ncu 分析结果。

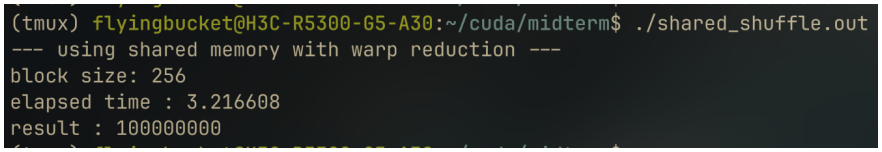


图 3: 第三版规约求和运行截图

## 2 总结

通过这三个版本的实现，我们可以看到使用共享内存和 shuffle 指令对规约求和的性能有显著提升。每一版相较上一版在运行时间上都有大约 10% 到 25% 的提升。