高性能计算导论 PA3 实验报告

Bernard Tan

June 2, 2022

1 实现方法

总共有三个 GPU kernel，分别对应分块 Floyd-Warshall 算法的三个阶段。每个阶段中的每个线程块都由 $32 \times 32 = 1024$ 个线程组成。每个线程块处理矩阵中划分好的一个块。

设每块大小为 $b \times b$，总共 $m \times m = \lceil \frac{n}{b} \rceil \times \lceil \frac{n}{b} \rceil$ 块。

1.1 阶段一

只调用中心块对应的线程块。

1.2 阶段二

调用十字块对应的 $2(m-1)$ 个线程块，利用已经计算好的中心块的结果进行更新。

1.3 阶段三

调用所有 $m \times m$ 个块，在阶段三的函数里面判断，如果为中心块或者十字块，就不进行之后的计算过程。

1.4 利用各级存储

拷贝到 shared_memory 上再进行计算。 graph 是在设备内存上的，读写速度相对于共享内存慢了很多倍。因此在每个 kernel 函数里面，都先把 graph 拷贝到线程块的共享内存上，再进行计算。计算结束后赋值回 graph。

kernel 函数内增加循环。 原先 Floyd-Warshall 算法中，中间节点 k 的枚举循环是在最外层的。可以将这层循环挪到 kernel 函数里面，在循环进行的过程中（k 每增加 1）就进行 ___syncthreads 线程同步，以免错误发生。这样做等效于把 k 的循环放在最外层。结果是大大提高了计算访存比。

线程块内，每个线程负责计算四个位置。 程序中取 $b = 64$，也就是说 32×32d 线程块负责计算 64×64 的区域。每个线程，假设编号 (i,j)，负责计算 (i,j)，(i+32,j)，(i,j+32)，(i+32,j+32) 四个位置。这样可以减少 kernel 函数的调用次数；在一个 kernel 函数中，同时拿到的共享内存量多了四倍，访存、计算的语句也变为了原来的四倍，可能有利于流水线和超标量加速。

循环内重复使用的共享内存用寄存器代替。 由于上述优化，循环中计算的语句为原来的四倍，重复使用的共享内存变量可以用寄存器变量代替计算，最后统一返还给共享内存。

2 实验结果

2.1 实验数据

n	耗时 (ms)	相对朴素实现的加速比
1000	2.207898	6.6842
2500	17.640705	21.3675
5000	107.578822	27.6178
7500	342.341705	29.2483
10000	796.963642	28.3438