CUDA 优化实验报告

容逸朗　2020010869

测量结果

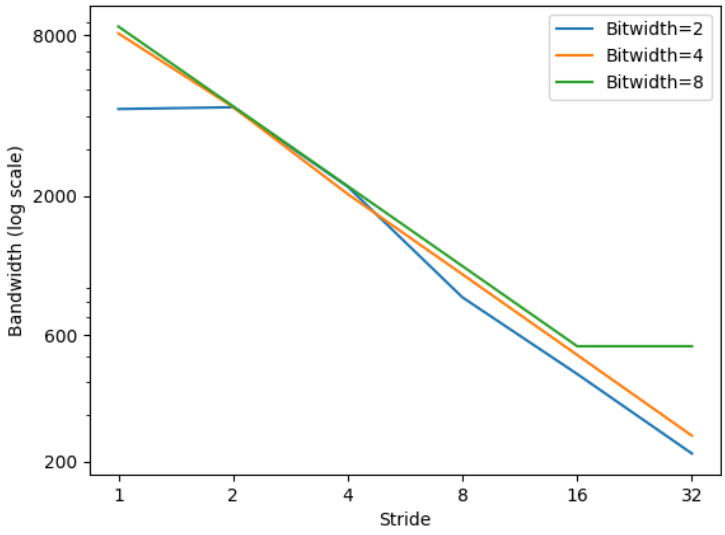
- 使用 global memory 的测量结果如下：

stride	bandwidth
1	530.02
2	182.489
4	91.9961
8	46.2886

- 使用 shared memory 的测量结果如下：

stride	bitwidth: 2	bitwidth: 4	bitwidth: 8
1	4236.73	8149.93	8648.63
2	4303	4309.13	4339.57
4	2161.88	2028.73	2173.54
8	829.905	1012.96	1087.65
16	427.725	504.478	544.071
32	214.939	250.814	544.07

- 对比不同 stride 下的性能：



性能分析

`test_gmem.cu` 的性能变化来源

- 性能变化的主要来源是 GPU 的哪种机制引起的?

  主要是由于合并访存 (Coalesced Memory Access) 引起的。

- 这种机制如何影响该程序的性能?

  在有合并访存机制的情况下，每次访存应该集中在特定几个的 sector (32bytes) 中。然而，当 stride 增大时，一个 sector 内读到的有效数据变少了，因此需要使用更多的 sector 来完成访存，导致程序变慢。

- 是否有其他的硬件功能参与了该程序的执行过程，它们会如何影响该程序的执行效率?

  在全局内存之间还有一层 L2 Cache，可以减少程序访问全局内存的时间。

`test_smem.cu` 的性能变化来源

- 固定 BITWIDTH 时，程序的性能变化来源于哪种硬件机制?

  主要来自 Bank Conflict 机制。

- BITWIDTH 设置为 2, 8 时性能变化的趋势相比于 BITWIDTH 设置为 4 时有什么不同之处，请解释。

  当 Bitwidth = 4 时，带宽随 stride 减少指数增长，但对于 Bitwidth = 2 和 8 时，出现了例外:

  - 当 Bitwidth = 2 时，设置 Stride = 1, 2 时的带宽是相同的:

    首先，我们知道一个 Bank 的大小是 4bytes 的，当 stride = 1 时，不同 thread 访存地址为 0, 2, 4, 6, ..., 62，尽管他们访问了相同的 Bank，但是访问的 byte 是不同的，因此不会出现冲突，此时每个 Wrap 中仅有前 16 个 Bank 被使用。

    当 stride = 2 时，一个 swap 刚好使用了所有的 Bank，但是每个 Bank 内只使用了一半的数据 (2B)，因此访问的数据量和 stride = 1 时是相同的，因此两者的带宽也相同。

  - 当 Bitwidth = 8 时，设置 Stride = 16, 32 时的带宽是相同的:

    这是因为，无论访存地址是 $16 \times 8 \times tid$ 或是 $32 \times 8 \times tid$ ，他们模 32×4 的余数都是零，即每次访存都对应 Bank-0，故这两种访存方式是 32 way-conflict 的，所以两者带宽相同。