小作业四：CUDA 并行策略（thread block，shared memory）

计04 何秉翔 2020010944

1. 测量结果

我们首先给出绘图的测量结果，详细测量结果放在报告末尾。

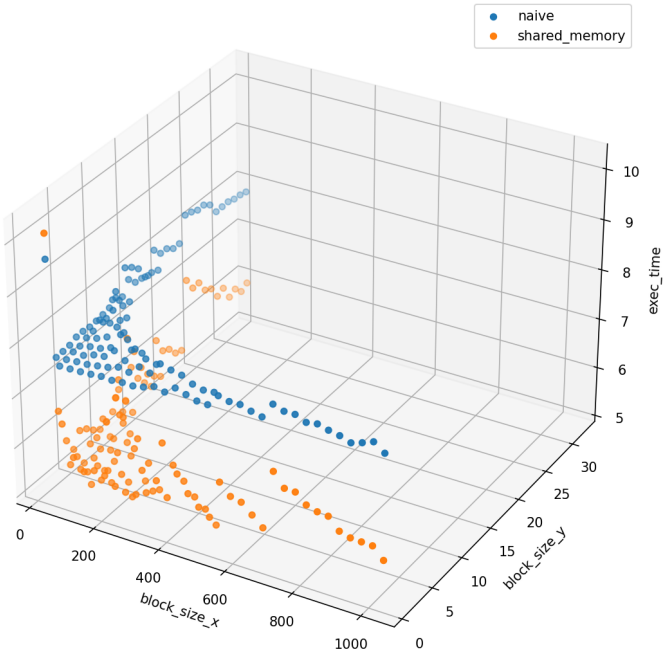

Figure: 三维散点图（block_size_x，block_size_y，exec_time）

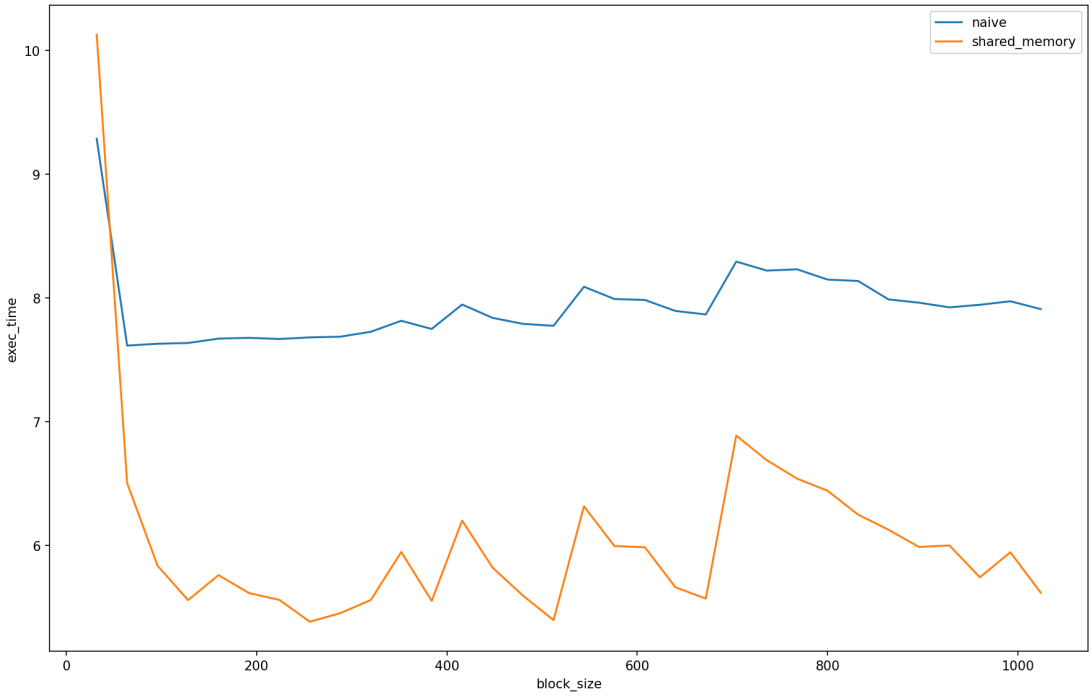

Figure: 二维折线图（block_size，exec_time）

- 不同的 thread block size 的选取，对程序性能的影响；
  - 可以从图中看出，整体上 block_size 越小，程序耗时越少，多数情况下性能越好，而且对于 naive 更明显。

    - 无论是 naive 还是 shared_memory，都出现了明显的锯齿状性能波动，当 block_size 越大时，波动越剧烈。
    - 无论是 naive 还是 shared_memory，都有几处当 block_size 增大时，程序耗时显著增加，但之后几次增大时，整体上耗时又会逐渐减少。
    - 有几处特定的 block_size 程序性能相对都较优，比如 block_size 为 256、512、1024 时，程序性能都比较好。
- Shared memory 是否使用，对程序性能的影响；
    - 可以看到，除去 block_size = 32 这个数据点，其余情况下 shared_memory 的使用都使得程序性能有显著提升。
- 上述两者的相互影响
    - 使用 shared_memory 后，程序性能随着 block_size 的增大波动较大，具体体现在个别 block_size 处发生性能的突变降低，随之而来的是性能逐渐变好。
    - 使用 shared_memory 后，较大的个别特定的 block_size 也能做到和较小的 block_size 接近的性能。

2. 原因分析

- 对于这个程序：
    - 如何设置 thread block size 才可以达到最好的效果？为什么？

      根据性能测试结果，对于 naive 的，设置 block_size 为 64 时效果最好；对于 shared_memory 的，设置为 256 时效果最好。因为线程块大小与许多因素有关，例如算法的特性、输入数据的规模、GPU 的硬件配置和限制等等。

      该程序是一个计算密集型程序，较小的线程块使得更多的线程块可以放到同一个 SM 上执行，当一个线程块的计算操作需要访问内存时，CUDA 会将这个线程块挂起并切换到运行其他线程块。这种方式可以提高 GPU 的利用率，从而提高程序的性能。

      而当使用 shared_memory 时，可以显著减少数据访问延迟。在使用共享内存时，较小的线程块可能无法充分利用SM内的资源，因为 SM 内的硬件资源是有限的，包括寄存器、共享内存等，因此较大的线程块可能更能充分利用SM内的资源，从而提高性能。
    - Shared memory 总是带来优化吗？如果不是，为什么？

      不总是带来优化，比如 block_size = 32 时就反而降低了程序性能。可能有以下几个原因：
        - 如果一个程序是计算密集型的，当共享内存的读写次数相对于计算操作较少时，使用共享内存可能无法带来显著的性能提升。
        - 如果一个程序访问的数据局部性较差，同时线程块较小，提前将数据放到共享内存内，重复利用率低，反而可能增大开销。
        - 另外共享内存的大小是有限的，若在一个线程块内使用较大的共享内存，线程块的数量就会减少，core 的利用率可能下降，也可能带来更多的资源竞争。
    - Shared memory 在什么 thread block size 下有效果，什么时候没有？

      需要选择合适的 block_size，对于较小的线程块，共享内存的优势可能不足以抵消其附加的开销，数据的传输占比较大；而对于较大的线程块，其共享内存的大小可能不足以容纳所有线程所需的数据，为了开辟较大的共享内存，线程块的数目减少，core 的利用率可能下降。
    - 还有哪些可以优化的地方？

      在使用了共享内存的程序里，对于边界条件的处理比较复杂，有较多的分支语句，需要多次判断是否越界，因此可以考虑统一边界和内部的处理代码。
- 对于任意一个给定程序：
    - 应该如何设置 thread block size？
        - 首先考虑所拥有的硬件资源，比如 SM 的数量，线程块中可用的寄存器个数等，一般而言，较大的线程块可以使得 GPU 的资源更充分地利用，但同时也会增大线程块之间的调度开销，需要在性能和资源利用率之间权衡。
        - 其次，需要考虑程序数据的访问是否具有局部性，如果局部连续，那么较大的线程块可以充分利用 shared_memory。
        - 然后再考虑问题的规模，如果问题规模较小，那么应该选择较小的线程块以充分利用资源；问题规模较大时，较大的线程块可以充分利用 GPU 的并行计算能力。
        - 最好的应该是设置不同的线程块大小，选取为 32 的倍数来进行性能测试，根据测试结果选择合适的 block_size。
    - 应该如何决定 shared memory 的使用？

- 看程序中需要访问全局内存的次数。如果需要大量访问全局内存，是内存密集型程序，则可以考虑使用共享内存，减少访问全局内存的次数。
- 看程序的计算复杂度。若计算复杂度较高，需要访问共享内存的线程可能需要等待其他的正在使用共享内存的线程计算完成，造成较大的资源竞争延迟问题。
- 看程序访问的局部性，若不同线程大量访存位于连续的局部的一片地址，则可以考虑使用共享内存。
- 考虑所拥有的硬件 GPU 的特性和限制。
- 考虑共享内存的大小的限制。

3. 详细测量结果

```
naive 32 1 Exec-time: 9.63323 ms
shared_memory 32 1 Exec-time: 10.1296 ms
naive 32 2 Exec-time: 7.61458 ms
shared_memory 32 2 Exec-time: 6.53782 ms
naive 32 3 Exec-time: 7.62615 ms
shared_memory 32 3 Exec-time: 5.82749 ms
naive 32 4 Exec-time: 7.64218 ms
shared_memory 32 4 Exec-time: 5.53656 ms
naive 32 5 Exec-time: 7.70003 ms
shared_memory 32 5 Exec-time: 5.80942 ms
naive 32 6 Exec-time: 7.70741 ms
shared_memory 32 6 Exec-time: 5.73373 ms
naive 32 7 Exec-time: 7.68768 ms
shared_memory 32 7 Exec-time: 5.59649 ms
naive 32 8 Exec-time: 7.70251 ms
shared_memory 32 8 Exec-time: 5.49646 ms
naive 32 9 Exec-time: 7.72659 ms
shared_memory 32 9 Exec-time: 5.54727 ms
naive 32 10 Exec-time: 7.75614 ms
shared_memory 32 10 Exec-time: 5.63854 ms
naive 32 11 Exec-time: 7.8525 ms
shared_memory 32 11 Exec-time: 5.98644 ms
naive 32 12 Exec-time: 7.77099 ms
shared_memory 32 12 Exec-time: 5.64983 ms
naive 32 13 Exec-time: 8.01503 ms
shared_memory 32 13 Exec-time: 6.22615 ms
naive 32 14 Exec-time: 7.85716 ms
shared_memory 32 14 Exec-time: 5.89118 ms
naive 32 15 Exec-time: 7.83439 ms
shared_memory 32 15 Exec-time: 5.69101 ms
naive 32 16 Exec-time: 7.78797 ms
shared_memory 32 16 Exec-time: 5.50335 ms
naive 32 17 Exec-time: 8.19975 ms
shared_memory 32 17 Exec-time: 6.35097 ms
naive 32 18 Exec-time: 8.03587 ms
shared_memory 32 18 Exec-time: 6.03353 ms
naive 32 19 Exec-time: 8.02823 ms
shared_memory 32 19 Exec-time: 5.95703 ms
naive 32 20 Exec-time: 7.93153 ms
shared_memory 32 20 Exec-time: 5.77146 ms
naive 32 21 Exec-time: 7.93109 ms
shared_memory 32 21 Exec-time: 5.7197 ms
naive 32 22 Exec-time: 8.41087 ms
shared_memory 32 22 Exec-time: 7.07482 ms
naive 32 23 Exec-time: 8.37677 ms
shared_memory 32 23 Exec-time: 6.81526 ms
naive 32 24 Exec-time: 8.31703 ms
shared_memory 32 24 Exec-time: 6.81941 ms
naive 32 25 Exec-time: 8.32264 ms
shared_memory 32 25 Exec-time: 6.56587 ms
naive 32 26 Exec-time: 8.22743 ms
```

```
shared_memory 32 26 Exec-time: 6.48944 ms
naive 32 27 Exec-time: 7.99111 ms
shared_memory 32 27 Exec-time: 6.22902 ms
naive 32 28 Exec-time: 7.99554 ms
shared_memory 32 28 Exec-time: 6.29638 ms
naive 32 29 Exec-time: 8.00034 ms
shared_memory 32 29 Exec-time: 6.00517 ms
naive 32 30 Exec-time: 7.95873 ms
shared_memory 32 30 Exec-time: 6.07262 ms
naive 32 31 Exec-time: 7.94808 ms
shared_memory 32 31 Exec-time: 5.9179 ms
naive 32 32 Exec-time: 7.91398 ms
shared_memory 32 32 Exec-time: 5.9975 ms
naive 64 1 Exec-time: 7.61342 ms
shared_memory 64 1 Exec-time: 6.45651 ms
naive 64 2 Exec-time: 7.63403 ms
shared_memory 64 2 Exec-time: 5.51761 ms
naive 64 3 Exec-time: 7.67989 ms
shared_memory 64 3 Exec-time: 5.5682 ms
naive 64 4 Exec-time: 7.68758 ms
shared_memory 64 4 Exec-time: 5.32392 ms
naive 64 5 Exec-time: 7.74461 ms
shared_memory 64 5 Exec-time: 5.60201 ms
naive 64 6 Exec-time: 7.77272 ms
shared_memory 64 6 Exec-time: 5.61578 ms
naive 64 7 Exec-time: 7.8872 ms
shared_memory 64 7 Exec-time: 5.88981 ms
naive 64 8 Exec-time: 7.81567 ms
shared_memory 64 8 Exec-time: 5.41091 ms
naive 64 9 Exec-time: 8.08263 ms
shared_memory 64 9 Exec-time: 6.09707 ms
naive 64 10 Exec-time: 7.96682 ms
shared_memory 64 10 Exec-time: 5.70202 ms
naive 64 11 Exec-time: 8.51258 ms
shared_memory 64 11 Exec-time: 7.09572 ms
naive 64 12 Exec-time: 8.41854 ms
shared_memory 64 12 Exec-time: 6.72169 ms
naive 64 13 Exec-time: 8.28847 ms
shared_memory 64 13 Exec-time: 6.28618 ms
naive 64 14 Exec-time: 8.0246 ms
shared_memory 64 14 Exec-time: 6.04249 ms
naive 64 15 Exec-time: 8.00892 ms
shared_memory 64 15 Exec-time: 5.89124 ms
naive 64 16 Exec-time: 7.95563 ms
shared_memory 64 16 Exec-time: 5.77614 ms
naive 96 1 Exec-time: 7.61974 ms
shared_memory 96 1 Exec-time: 5.84393 ms
naive 96 2 Exec-time: 7.65475 ms
shared_memory 96 2 Exec-time: 5.48714 ms
naive 96 3 Exec-time: 7.67149 ms
shared_memory 96 3 Exec-time: 5.32162 ms
naive 96 4 Exec-time: 7.73907 ms
shared_memory 96 4 Exec-time: 5.49215 ms
naive 96 5 Exec-time: 7.79618 ms
shared_memory 96 5 Exec-time: 5.56314 ms
naive 96 6 Exec-time: 8.01271 ms
shared_memory 96 6 Exec-time: 5.99792 ms
naive 96 7 Exec-time: 7.88723 ms
shared_memory 96 7 Exec-time: 5.54814 ms
naive 96 8 Exec-time: 8.4076 ms
shared_memory 96 8 Exec-time: 6.6415 ms
```

```
naive 96 9 Exec-time: 8.02299 ms
shared_memory 96 9 Exec-time: 6.09269 ms
naive 96 10 Exec-time: 7.99798 ms
shared_memory 96 10 Exec-time: 5.75414 ms
naive 128 1 Exec-time: 7.62816 ms
shared_memory 128 1 Exec-time: 5.62127 ms
naive 128 2 Exec-time: 7.66414 ms
shared_memory 128 2 Exec-time: 5.24619 ms
naive 128 3 Exec-time: 7.74373 ms
shared_memory 128 3 Exec-time: 5.5116 ms
naive 128 4 Exec-time: 7.76551 ms
shared_memory 128 4 Exec-time: 5.29952 ms
naive 128 5 Exec-time: 7.88378 ms
shared_memory 128 5 Exec-time: 5.60515 ms
naive 128 6 Exec-time: 8.30544 ms
shared_memory 128 6 Exec-time: 6.56248 ms
naive 128 7 Exec-time: 8.00188 ms
shared_memory 128 7 Exec-time: 5.93018 ms
naive 128 8 Exec-time: 7.94675 ms
shared_memory 128 8 Exec-time: 5.60015 ms
naive 160 1 Exec-time: 7.64051 ms
shared_memory 160 1 Exec-time: 5.71082 ms
naive 160 2 Exec-time: 7.69651 ms
shared_memory 160 2 Exec-time: 5.40187 ms
naive 160 3 Exec-time: 7.77566 ms
shared_memory 160 3 Exec-time: 5.48023 ms
naive 160 4 Exec-time: 7.84095 ms
shared_memory 160 4 Exec-time: 5.55527 ms
naive 160 5 Exec-time: 8.14017 ms
shared_memory 160 5 Exec-time: 6.38894 ms
naive 160 6 Exec-time: 7.92473 ms
shared_memory 160 6 Exec-time: 5.71252 ms
naive 192 1 Exec-time: 7.66098 ms
shared_memory 192 1 Exec-time: 5.67203 ms
naive 192 2 Exec-time: 7.73181 ms
shared_memory 192 2 Exec-time: 5.41575 ms
naive 192 3 Exec-time: 7.95007 ms
shared_memory 192 3 Exec-time: 5.91158 ms
naive 192 4 Exec-time: 8.15907 ms
shared_memory 192 4 Exec-time: 6.51454 ms
naive 192 5 Exec-time: 7.95446 ms
shared_memory 192 5 Exec-time: 5.61753 ms
naive 224 1 Exec-time: 7.6467 ms
shared_memory 224 1 Exec-time: 5.52495 ms
naive 224 2 Exec-time: 7.79619 ms
shared_memory 224 2 Exec-time: 5.65294 ms
naive 224 3 Exec-time: 7.81719 ms
shared_memory 224 3 Exec-time: 5.37821 ms
naive 224 4 Exec-time: 7.91772 ms
shared_memory 224 4 Exec-time: 5.94568 ms
naive 256 1 Exec-time: 7.66595 ms
shared_memory 256 1 Exec-time: 5.47137 ms
naive 256 2 Exec-time: 7.75812 ms
shared_memory 256 2 Exec-time: 5.26258 ms
naive 256 3 Exec-time: 8.10751 ms
shared_memory 256 3 Exec-time: 6.32511 ms
naive 256 4 Exec-time: 7.91917 ms
shared_memory 256 4 Exec-time: 5.35286 ms
naive 288 1 Exec-time: 7.66235 ms
shared_memory 288 1 Exec-time: 5.49364 ms
naive 288 2 Exec-time: 7.92861 ms
```

```
shared_memory 288 2 Exec-time: 5.86699 ms
naive 288 3 Exec-time: 7.97473 ms
shared_memory 288 3 Exec-time: 5.8972 ms
naive 320 1 Exec-time: 7.70439 ms
shared_memory 320 1 Exec-time: 5.59766 ms
naive 320 2 Exec-time: 7.86805 ms
shared_memory 320 2 Exec-time: 5.51392 ms
naive 320 3 Exec-time: 7.92933 ms
shared_memory 320 3 Exec-time: 5.4243 ms
naive 352 1 Exec-time: 7.77597 ms
shared_memory 352 1 Exec-time: 5.90754 ms
naive 352 2 Exec-time: 8.11079 ms
shared_memory 352 2 Exec-time: 6.55036 ms
naive 384 1 Exec-time: 7.7329 ms
shared_memory 384 1 Exec-time: 5.63358 ms
naive 384 2 Exec-time: 8.05355 ms
shared_memory 384 2 Exec-time: 6.17129 ms
naive 416 1 Exec-time: 7.87561 ms
shared_memory 416 1 Exec-time: 6.17634 ms
naive 416 2 Exec-time: 8.01516 ms
shared_memory 416 2 Exec-time: 5.9161 ms
naive 448 1 Exec-time: 7.81074 ms
shared_memory 448 1 Exec-time: 5.84846 ms
naive 448 2 Exec-time: 7.90478 ms
shared_memory 448 2 Exec-time: 5.66559 ms
naive 480 1 Exec-time: 7.74927 ms
shared_memory 480 1 Exec-time: 5.64366 ms
naive 480 2 Exec-time: 7.85144 ms
shared_memory 480 2 Exec-time: 5.48623 ms
naive 512 1 Exec-time: 7.74098 ms
shared_memory 512 1 Exec-time: 5.51038 ms
naive 512 2 Exec-time: 7.87163 ms
shared_memory 512 2 Exec-time: 5.23173 ms
naive 544 1 Exec-time: 7.97919 ms
shared_memory 544 1 Exec-time: 6.28137 ms
naive 576 1 Exec-time: 7.93031 ms
shared_memory 576 1 Exec-time: 6.06306 ms
naive 608 1 Exec-time: 7.93607 ms
shared_memory 608 1 Exec-time: 6.01569 ms
naive 640 1 Exec-time: 7.871 ms
shared_memory 640 1 Exec-time: 5.82856 ms
naive 672 1 Exec-time: 7.82466 ms
shared_memory 672 1 Exec-time: 5.63984 ms
naive 704 1 Exec-time: 8.13525 ms
shared_memory 704 1 Exec-time: 6.82859 ms
naive 736 1 Exec-time: 8.06259 ms
shared_memory 736 1 Exec-time: 6.56219 ms
naive 768 1 Exec-time: 8.06646 ms
shared_memory 768 1 Exec-time: 6.55601 ms
naive 800 1 Exec-time: 7.97699 ms
shared_memory 800 1 Exec-time: 6.36893 ms
naive 832 1 Exec-time: 8.01318 ms
shared_memory 832 1 Exec-time: 6.3025 ms
naive 864 1 Exec-time: 7.95683 ms
shared_memory 864 1 Exec-time: 6.2829 ms
naive 896 1 Exec-time: 7.9171 ms
shared_memory 896 1 Exec-time: 6.05546 ms
naive 928 1 Exec-time: 7.84523 ms
shared_memory 928 1 Exec-time: 5.99299 ms
naive 960 1 Exec-time: 7.91283 ms
shared_memory 960 1 Exec-time: 5.98292 ms
```

```
naive 992 1 Exec-time: 7.99628 ms
shared_memory 992 1 Exec-time: 5.97444 ms
naive 1024 1 Exec-time: 7.84246 ms
shared_memory 1024 1 Exec-time: 5.75574 ms
```