1 REPORT

1. Implementation

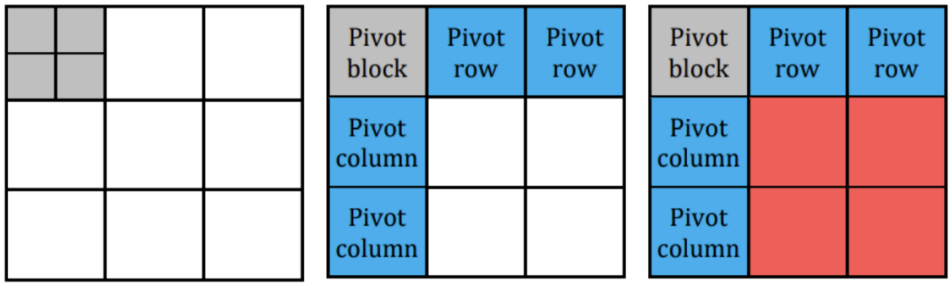
a. Which algorithm do you choose in hw3-1?

利用 OpenMP 平行化 Floyd-Warshall algorithm 的 for 迴圈。

```
for (k = 0; k < n; k++)
#pragma omp parallel for num_threads(cpu_count) schedule(static, 1)
  for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++) {
          if (Dist[i][k] + Dist[k][j] < Dist[i][j] )
              Dist[i][j] = Dist[i][k] + Dist[k][j];
      }
  }
```

b. How do you divide your data in hw3-2, hw3-3?

hw3-2、hw3-3：

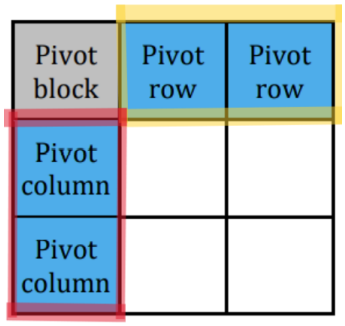


利用範例的方法 divide data。

作法：

除了 phase1 之外，phase2、3 將同一個 phase 拆分為多個部分利用 kernel 分開做，例如下圖為 phase2 的階段，圖中藍色區塊為 phase2 須處理、更新的 data，則須個別處理兩個部分(kernel)，分別為黃色框部分與紅色框部分，因為黃色部分的 size 為 1x2，因此分配此 kernel 的 grid size 為 1 x 2，而紅色部分則會被分配到 2 x 1 的 grid size 。

而在 kernel 的每個 block 中，若 block factor 為 k，則 create 出 k x k 的 threads 個別處理 block 中的 data。



c. configuration in hw3-2, hw3-3? And why?

hw3-2、hw3-3：

- # blocks：詳見 b. 說明
- # threads：(32, 32)

因為同一個 block 中 thread 最多可以開出 1024 條，因此想利用二維 thread 來管理每一個 block 進行計算，因此最多開出的 thread 數量為 32 x 32。

- blocking factor：32

blocking factor 越大，越能夠縮減 shared memory、global memory 之間的搬移次數以提升效能，但因為被 maximum thread 的數量限制住了，所以 blocking factor 最多也只能開到 32 。

d. communication in hw3-3

利用 peer to peer 進行 communication，data 在 GPU memory 之間進行傳輸。

2. Profiling Results (hw3-2)

Data set：p11k1

Occupancy：

Type	Time(%)	Time	Calls	Avg	Min	Max
GPU activities:	97.90%	10.6179s	2746	3.8667ms	2.5920us	15.802ms

sm efficiency：

Metric	Description	Min	Max	Avg
Multiprocessor Activity		2.23%	99.99%	98.93%

shared memory load/store throughput：

```
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: p3_cal(int, int, int, int, int, int*)
       2746                    shared_load_throughput             Shared Memory Load Throughput  0.00000B/s  0.00000B/s  0.00000B/s
       2746                   shared_store_throughput            Shared Memory Store Throughput  685.79MB/s  330.41GB/s  150.18GB/s
    Kernel: p1_cal(int, int, int, int, int, int*)
        688                    shared_load_throughput             Shared Memory Load Throughput  4.2198GB/s  14.163GB/s  13.332GB/s
        688                   shared_store_throughput            Shared Memory Store Throughput  303.83MB/s  1.9549GB/s  993.16MB/s
    Kernel: p2_cal(int, int, int, int, int, int*, int)
       2748                    shared_load_throughput             Shared Memory Load Throughput  3.1472GB/s  652.65GB/s  543.70GB/s
       2748                   shared_store_throughput            Shared Memory Store Throughput  460.39MB/s  292.13GB/s  139.23GB/s
```

global store throughput：

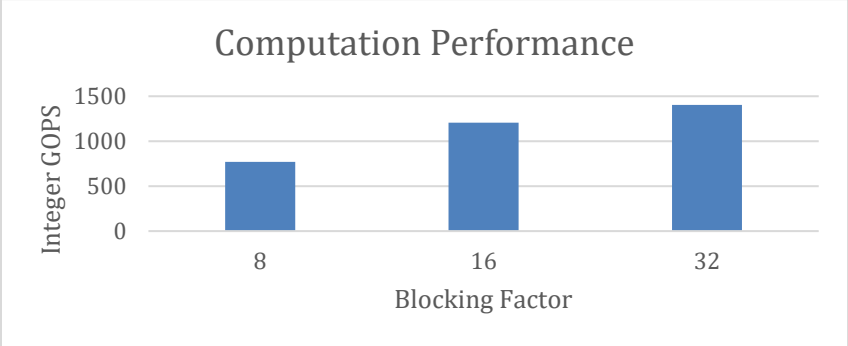
```
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: p3_cal(int, int, int, int, int, int*)
       2746                            gst_throughput                Global Store Throughput  95.367MB/s  77.333GB/s  42.890GB/s
    Kernel: p1_cal(int, int, int, int, int, int*)
        688                            gst_throughput                Global Store Throughput  110.57MB/s  492.22MB/s  450.00MB/s
    Kernel: p2_cal(int, int, int, int, int, int*, int)
       2748                            gst_throughput                Global Store Throughput  72.661MB/s  32.192GB/s  17.613GB/s
```

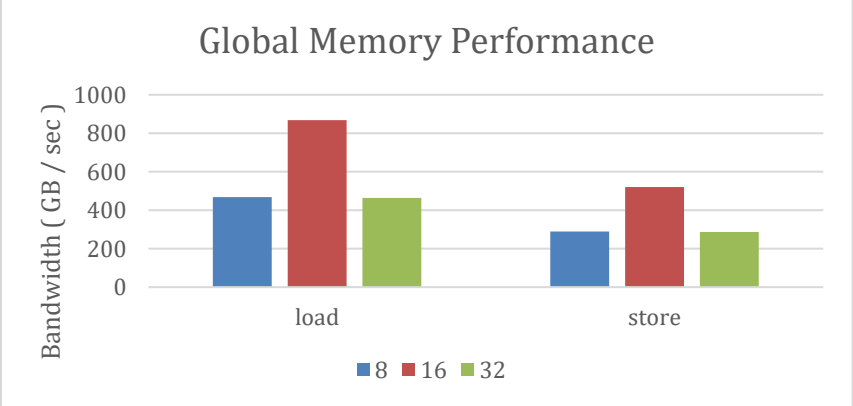
3. Experiment & Analysis

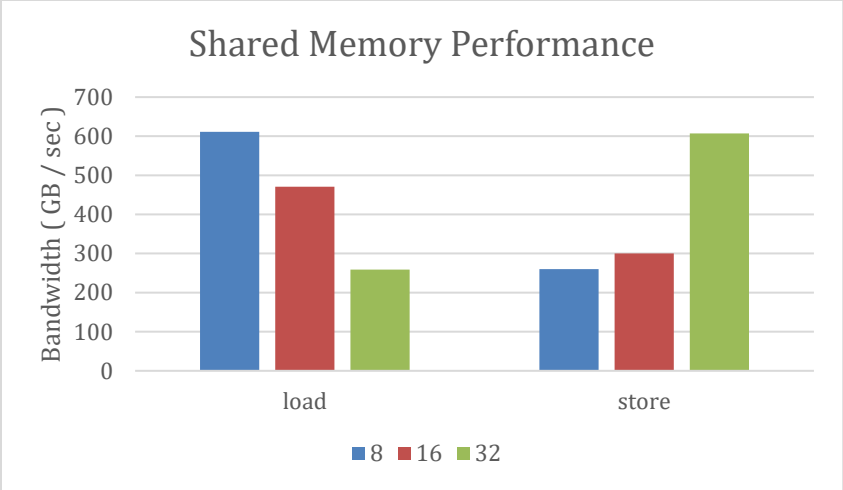
Blocking Factor (hw3-2)

Test case：C21.1

利用 –metrics inst_integer 得出每個 kernel instruction 的數量，再利用 nvprof 得出 kernel 的執行時間以及呼叫次數算出 integer GOPs 值。







由上述結果得知，增加 blocking factor 可以提高 interger GOPs，效能越佳，而我的程式最多只能調到 32，因此 data transfer 成為了 bottleneck。

a. Optimization **(hw3-2)**

- Shared memory

  將 global memory 資料放入 shared memory 再進行計算以減少執行時間。

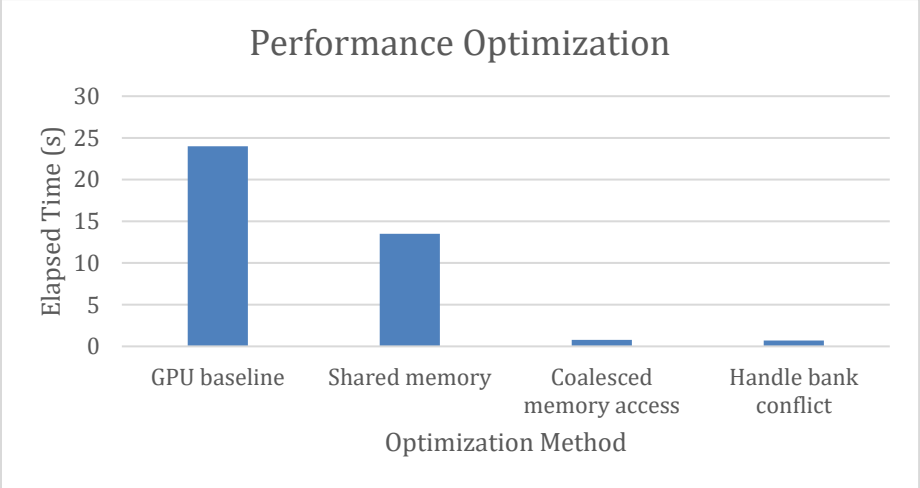
- Coalesced memory access

  依照 row major 從 global memory array 中取出 data，因為運算完後存入 array 時會變成 column major ，因此將預存資料進行轉置後再存入 global memory 中

- Handle bank conflict

  利用 Memory Padding 技術使同個 warp 的 thread 存取不同的 bank

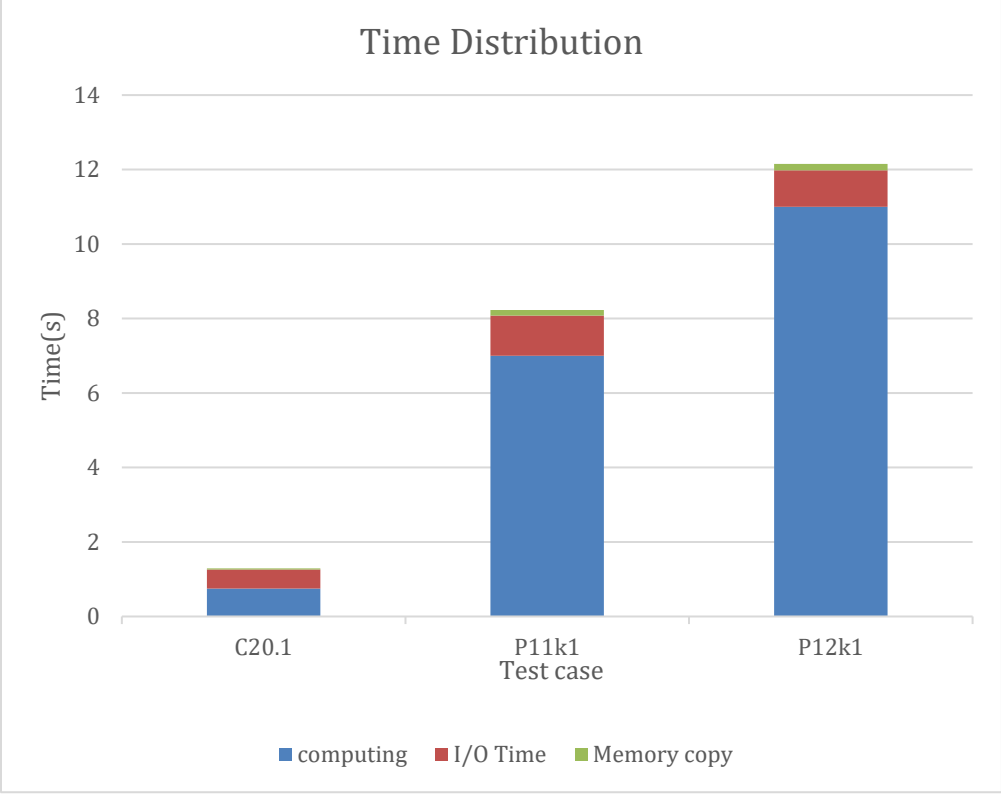
Test case ：c20.1



Time Distribution **(hw3-2)**

b. Analyze the time spent in:

	C20.1	**P11k1**	**P12k1**
Vertex	5000	11000	11817
Edge	7594839	505586	3529066



4. Experience & conclusion

a. learned from this homework

這次對我的程式進行分析過後覺得自己的程式有很多地方需要改進，在我的方法中，block factor 最多只能開到 32，這對效能來說影響非常大，若 block factor 能開的更大，則可以讓更多的資料共享 shared memory，可大幅減少 shared memory 與 global memory 搬移的次數以及資料量，在我的程式裡有些 shared memory 與 global memory 之間的搬移過程也沒有用上全部的 thread，造成這些 thread 沒有功能且占據了 SM 導致效能下降。經過這次的經驗，下次再寫 CUDA code 時會再更徹底的思考如何開出合適的 block、thread 以增加效能。

b. Feedback

在寫作業時助教幫我們寫的 serial code 替我省下不少時間，會有更多時間專注在如何進行平行化。因為自己是第一次碰 CUDA，很多地方還不太熟悉，光是 debug 就花了蠻多時間，前面的 lab 作業還沒寫出來緊接著就需要開始寫 HW3，因此對我來說我覺得這次的 cuda labs 以及作業相較其他主題來說時間上排得稍些緊密，若提供更多的緩衝時間則會更好。