Parallel Programming hw4-1

tags: PP20

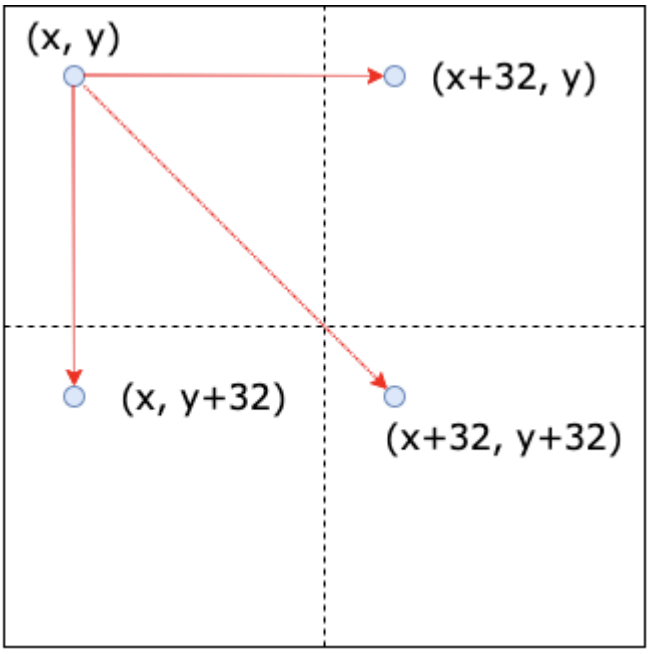
106062230 徐嘉欣

Implementation

首先，我在做input時，有做padding的部分，讓整個2D array的長寬都是64的倍數(因為**Blocking factor取64**)，這樣在device端就不用怕存取到超過memory範圍的部分。

```
fread(&v, sizeof(int), 1, file);
fread(&m, sizeof(int), 1, file);
if(v%64) n = v + (64 - v%64);
else n = v;
cudaMallocHost( &Dist, sizeof(int)*(n*n));
```

而在做blocked floyd warshall時，因為device的每個block最多只能包含1024個threads，因此這邊取的**thread是dim3(32, 32)**，比起每次都只有做一遍的computing，一次做4遍的computing(有點像在模擬(64,64)的thread)、盡可能最大地利用shared memory的大小會使效能變得好些。



而每個2D array切割成的各個block剛好會分給device中的不同block，因此device的block數取得則是各個phase中，長與寬分別有多少block(i.e., **dim3(block_width, block_height)**)。

```
dim3 thread(32, 32);

/* Phase 1*/
phase1<<< 1, thread, B*B*sizeof(int) >>>(B, r,    r,  r, n, d_dist,
pitch_int);
```

```
/* Phase 2*/
phase2_2<<<dim3(r, 1),             thread, 8192*sizeof(int)>>>(B, r, n,
d_dist, pitch_int, r, 0);  // up
phase2_2<<<dim3(round - r - 1, 1), thread, 8192*sizeof(int)>>>(B, r, n,
d_dist, pitch_int, r, r+1);  // down
phase2_1<<<dim3(1, r),             thread, 8192*sizeof(int)>>>(B, r, n,
d_dist, pitch_int, 0, r); // left
phase2_1<<<dim3(1, round - r - 1), thread, 8192*sizeof(int)>>>(B, r, n,
d_dist, pitch_int, r+1, r); // right

/* Phase 3*/
phase3<<<dim3(r, r), thread, 8192*sizeof(int)>>>(B, k_min, n, d_dist,
pitch_int, 0, 0);
phase3<<<dim3(round - r - 1, r), thread, 8192*sizeof(int)>>>(B, k_min, n,
d_dist, pitch_int, 0, r+1);
phase3<<<dim3(r, round - r - 1), thread, 8192*sizeof(int)>>>(B, k_min, n,
d_dist, pitch_int, r+1, 0);
phase3<<<dim3(round - r - 1, round - r - 1), thread, 8192*sizeof(int)>>>
(B, k_min, n, d_dist, pitch_int, r+1, r+1);
```

phase1與phase2, phase3大部分蠻相近的，只有一小部分的念頭不同，因此先解釋phase1，phase2與3只挑選與phase1不同的部分做解釋。

會將phase1~phase3改寫成各個不同的function，是因為不同的phase中，`Dist[i][j]`、`Dist[i][k]`、`Dist[k][j]`有可能會有重疊在同一塊block的情況，如phase1中，`Dist[i][j]`、`Dist[i][k]`、`Dist[k][j]`都會落在同一個block中，因此只需要load global memory至一個shared memory中就好了。

首先先計算出`Dist[i][j]`、`Dist[i][k]`、`Dist[k][j]`所在的block的左上方的頂點，再來對於`dist[i][j]`load 4遍的global memory，因為thread的大小最大是1024，因此不能開(64, 64)的thread，因此用(32, 32)的話，要再多讀3個global memory。讀完之後就可以進行floyd warshall了，在判斷是否有更小的路徑時，原先使用if去做判斷，但**這樣可能會有diversity**的狀況，致使效能降低，因此改用`min`去取較小的值，讓warp中的所有人都做一樣的事，降低diversity。最後再存回global memory就完成phase1了。

```
__global__ void phase1(int B, int Round, int block_start_x, int
block_start_y, int n, int* d_dist, int p) {

    int b_i = (block_start_x << 6) + threadIdx.y;
    int b_j = (block_start_y << 6) + threadIdx.x;

    extern __shared__ int shared_mem[];

    #pragma unroll
    for(int r=0; r<2; ++r){
        int idx = threadIdx.y + (r << 5);
        shared_mem[idx*B + threadIdx.x] = d_dist[(b_i + (r << 5))*p +
b_j];
        shared_mem[idx*B + threadIdx.x + 32] = d_dist[(b_i + (r << 5))*p +
b_j + 32];
    }
```

```
    #pragma unroll
    for (int k = 0; k < 64; ++k) {
        __syncthreads();
        for(int r=0; r<2; ++r){
            int idx = threadIdx.y + (r << 5);
            shared_mem[idx*B+threadIdx.x] =
min(shared_mem[idx*B+threadIdx.x], shared_mem[idx*B+k] +
shared_mem[k*B+threadIdx.x]);
            shared_mem[idx*B+threadIdx.x + 32] =
min(shared_mem[idx*B+threadIdx.x + 32], shared_mem[idx*B+k] +
shared_mem[k*B+threadIdx.x + 32]);
        }
    }

    #pragma unroll
    for(int r=0; r<2; ++r){
        d_dist[(b_i + (r << 5))*p + b_j] = shared_mem[(threadIdx.y + (r <<
5))*B + threadIdx.x];
        d_dist[(b_i + (r << 5))*p + b_j + 32] = shared_mem[(threadIdx.y +
(r << 5))*B + threadIdx.x + 32];
    }
}
```

phase2則分成`phase2_1`與`phase2_2`兩個:

- `phase2_1`是負責pivot左右的長條狀block們，因為負責的是pivot左右兩塊，因此`Dist[i][j]`與`Dist[i][k]`其實是落在同一個block中，因此這兩個可以共同儲存在同一個shared memory中。
- `phase2_2`是負責pivot上下的長條狀block們，因為負責的是pivot上下兩塊，因此`Dist[i][j]`與`Dist[k][j]`其實是落在同一個block中，因此這兩個可以共同儲存在同一個shared memory中。

phase3的話，因為`Dist[i][j]`、`Dist[i][k]`、`Dist[k][j]`都沒有重疊到，再加上其實只有自己會用到`Dist[i][j]`(前面的phase因為`Dist[i][j]`皆有與其他Dist重複到，因此要load進shared memory做共用)，所以不需要再開shared memory給`Dist[i][j]`儲存，只需儲存`Dist[i][k]`、`Dist[k][j]`就好，`Dist[i][j]`可以使用register去記錄。

Profiling Results

The results below are based on running testcase c21.1.

```
==729731== NVPROF is profiling process 729731, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==729731== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
==729731== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==729731== Profiling result:
==729731== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
        310                             sm_efficiency                   Multiprocessor Activity      3.58%      99.72%      94.66%
        310                         achieved_occupancy                        Achieved Occupancy    0.489591    0.958671    0.928407
        310                             gld_throughput                  Global Load Throughput  7.0360GB/s  219.26GB/s  201.95GB/s
        310                             gst_throughput                  Global Store Throughput  2.3453GB/s  73.087GB/s  67.316GB/s
        310                      shared_load_throughput             Shared Memory Load Throughput  112.58GB/s  3508.2GB/s  3231.1GB/s
        310                     shared_store_throughput            Shared Memory Store Throughput  4.6907GB/s  146.17GB/s  134.63GB/s
    Kernel: phase1(int, int, int, int, int, int*)
         79                             sm_efficiency                   Multiprocessor Activity      4.57%      4.64%       4.63%
         79                         achieved_occupancy                        Achieved Occupancy    0.497469    0.497553    0.497509
         79                             gld_throughput                  Global Load Throughput  583.15MB/s  663.14MB/s  591.04MB/s
         79                             gst_throughput                  Global Store Throughput  583.15MB/s  663.14MB/s  591.04MB/s
         79                      shared_load_throughput             Shared Memory Load Throughput  109.63GB/s  124.66GB/s  111.11GB/s
         79                     shared_store_throughput            Shared Memory Store Throughput  37.017GB/s  42.094GB/s  37.517GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
        156                             sm_efficiency                   Multiprocessor Activity      4.52%      95.83%      71.05%
        156                         achieved_occupancy                        Achieved Occupancy    0.496750    0.964880    0.771825
        156                             gld_throughput                  Global Load Throughput  1.3909GB/s  30.977GB/s  22.497GB/s
        156                             gst_throughput                  Global Store Throughput  712.14MB/s  15.489GB/s  11.248GB/s
        156                      shared_load_throughput             Shared Memory Load Throughput  133.87GB/s  2981.6GB/s  2165.3GB/s
        156                     shared_store_throughput            Shared Memory Store Throughput  45.899GB/s  1022.3GB/s  742.39GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
        156                             sm_efficiency                   Multiprocessor Activity      4.51%      94.51%      70.99%
        156                         achieved_occupancy                        Achieved Occupancy    0.496846    0.970496    0.772258
        156                             gld_throughput                  Global Load Throughput  1.3875GB/s  31.176GB/s  22.218GB/s
        156                             gst_throughput                  Global Store Throughput  710.42MB/s  15.588GB/s  11.109GB/s
        156                      shared_load_throughput             Shared Memory Load Throughput  133.55GB/s  3000.7GB/s  2138.5GB/s
        156                     shared_store_throughput            Shared Memory Store Throughput  45.789GB/s  1028.8GB/s  733.19GB/s
```

Experiment & Analysis

System Spec

使用hades來做實驗與測量。

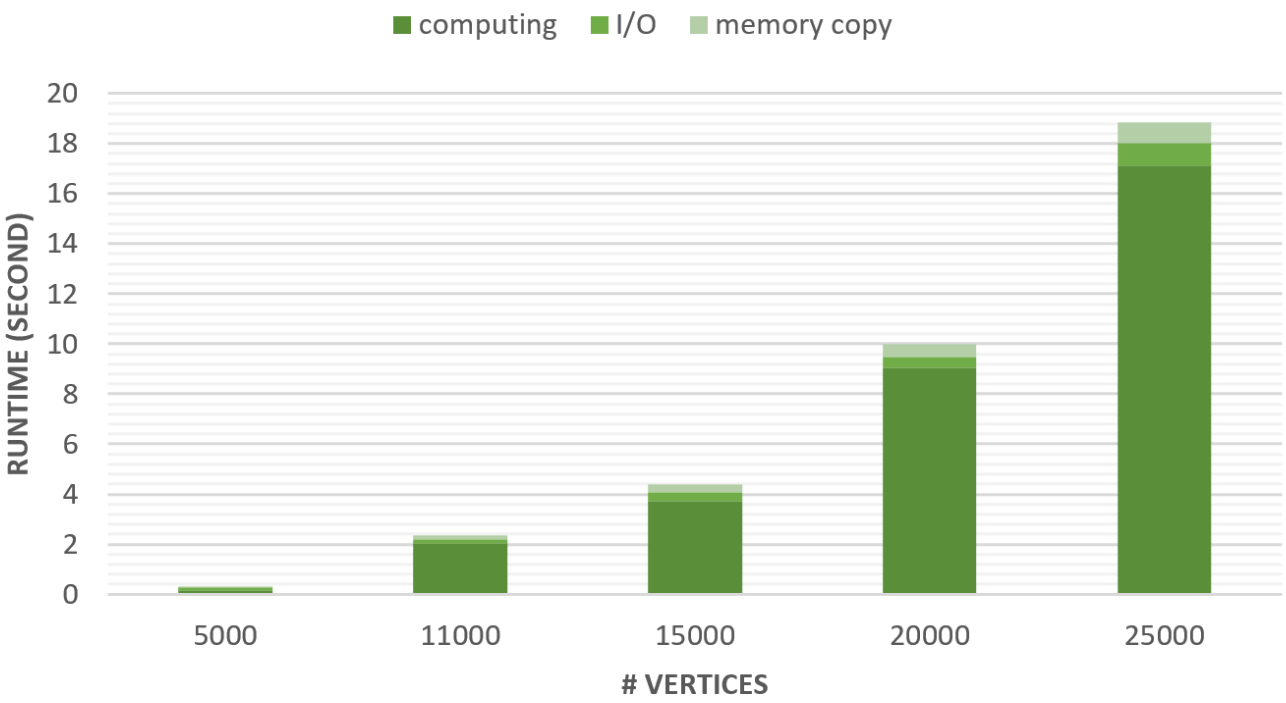
Time Distribution

computing time與memory copy time(H2D, D2H)都是透過`nvprof`來測量，其中computing time為phase1, phase2-1, phase2-2, 與phase3四者的時間總和。而I/O time則是透過以下的方式測量：

```
std::chrono::steady_clock::time_point t1 =
std::chrono::steady_clock::now();
/* doing I/O here */
std::chrono::steady_clock::time_point t2 =
std::chrono::steady_clock::now();
std::cout << "Reading(or writing) file took " <<
std::chrono::duration_cast<std::chrono::microseconds>(t2 - t1).count() <<
"us.\n";
```

testcase	n	m	input (second)	output (second)	computing (second)	H2D (second)	D2H (second)
p25k1	25000	5780158	0.0323	0.9038	17.1074	0.4167	0.3790
p20k1	20000	264275	0.0029	0.4353	9.0497	0.2473	0.2434
p15k1	15000	5591272	0.0342	0.3328	3.7257	0.1509	0.1370
p11k1	11000	505586	0.0048	0.1815	2.0307	0.0808	0.0733

testcase	n	m	input (second)	output (second)	computing (second)	H2D (second)	D2H (second)
c21.1	5000	10723117	0.0627	0.0407	0.1594	0.0170	0.0154



可發現做input的時間與m為正相關(因為要讀取m這麼多個距離的pair進來)，而output、computing、memory copy則都是與n為正相關。

Blocking Factor

The results below is based on running testcase c21.1.

global memory(shared memory memory) bandwidth的測量方式是使用`nvprof`加上`-m gld_throughput,gst_throughput`(`-m shared_load_throughtput,shared_store_throughput`)獲得，這邊取phase3的average來做比較，獲取方式如下圖:

```
[pp20s61@hades01 block_16]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m gld_throughput,gst_throughput,shared_load_throughput,shared_
ome/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
srun: job 113029 queued and waiting for resources
srun: job 113029 has been allocated resources
==736105== NVPROF is profiling process 736105, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736105== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
Execution took 230133422us.
==736105== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736105== Profiling result:
==736105== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
       1246                            gld_throughput                  Global Load Throughput  806.63MB/s  239.90GB/s  230.08GB/s
       1246                            gst_throughput                 Global Store Throughput  537.75MB/s  159.93GB/s  153.38GB/s
       1246                     shared_load_throughput           Shared Memory Load Throughput  6.3018GB/s  1919.2GB/s  1840.6GB/s
       1246                    shared_store_throughput          Shared Memory Store Throughput  1.0503GB/s  319.87GB/s  306.77GB/s
    Kernel: phase1(int, int, int, int, int, int*)
        313                            gld_throughput                  Global Load Throughput  157.31MB/s  175.39MB/s  165.75MB/s
        313                            gst_throughput                 Global Store Throughput  314.61MB/s  350.78MB/s  331.50MB/s
        313                     shared_load_throughput           Shared Memory Load Throughput  12.520GB/s  13.959GB/s  13.192GB/s
        313                    shared_store_throughput          Shared Memory Store Throughput  5.2231GB/s  5.8234GB/s  5.5034GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
        624                            gld_throughput                  Global Load Throughput  346.79MB/s  46.411GB/s  32.944GB/s
        624                            gst_throughput                 Global Store Throughput  346.79MB/s  46.411GB/s  32.944GB/s
        624                     shared_load_throughput           Shared Memory Load Throughput  13.801GB/s  1891.2GB/s  1342.5GB/s
        624                    shared_store_throughput          Shared Memory Store Throughput  6.0959GB/s  835.40GB/s  593.00GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
        624                            gld_throughput                  Global Load Throughput  428.32MB/s  33.720GB/s  27.839GB/s
        624                            gst_throughput                 Global Store Throughput  428.32MB/s  33.720GB/s  27.839GB/s
        624                     shared_load_throughput           Shared Memory Load Throughput  17.045GB/s  1374.1GB/s  1134.5GB/s
        624                    shared_store_throughput          Shared Memory Store Throughput  7.5290GB/s  606.96GB/s  501.11GB/s
```

而測量GOPS的方式則是先使用`nvprof`加上`-m inst_integer`，拿phase3的average乘以phase3總共跑的次數，以獲得phase3總共的integer instructions次數，再使用`nvprof`去獲取執行phase3的時間去除，就能獲得GOPS。

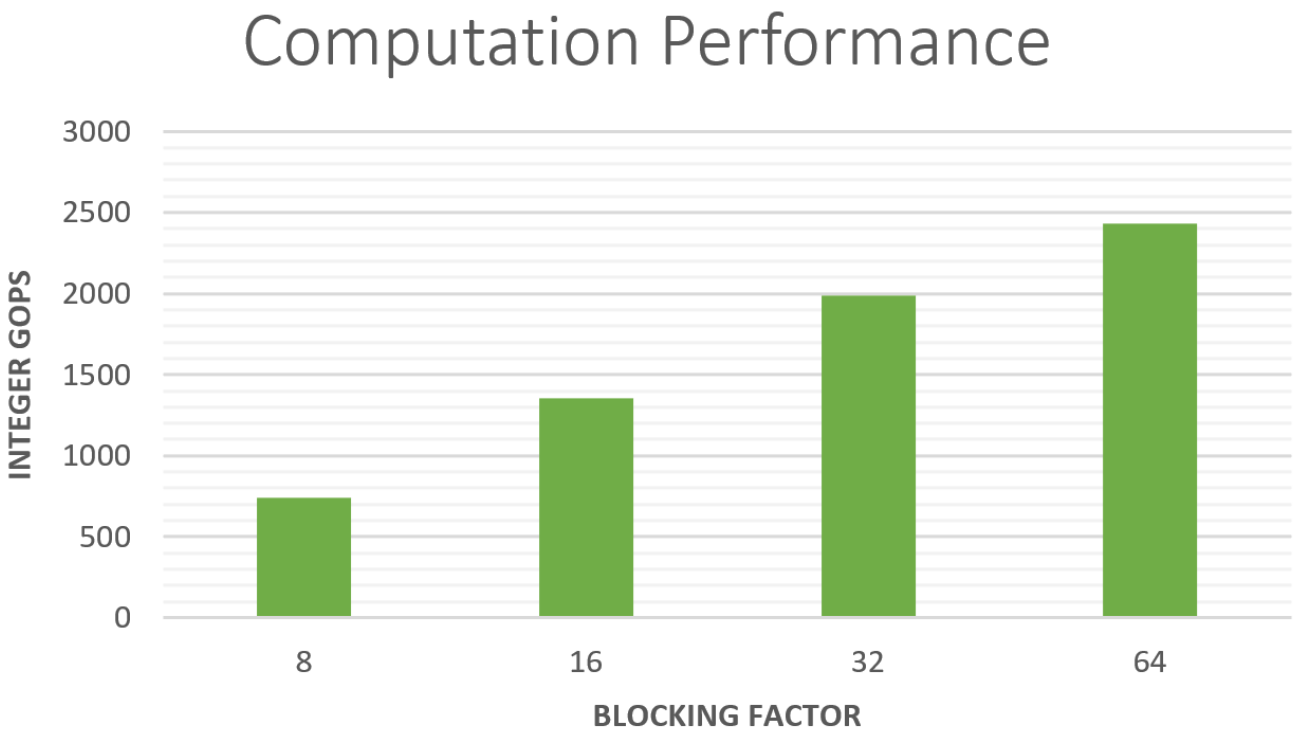
```
[pp20s61@hades02 block_16]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m inst_integer ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==746213== NVPROF is profiling process 746213, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
Execution took 82775907us.
==746213== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==746213== Profiling result:
==746213== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
       1246                            inst_integer                    Integer Instructions       18176  1769324544   444461141
    Kernel: phase1(int, int, int, int, int, int*)
        313                            inst_integer                    Integer Instructions       14912       14912       14912
    Kernel: phase2_2(int, int, int, int*, int, int)
        624                            inst_integer                    Integer Instructions       17344     5411328     2714336
    Kernel: phase2_1(int, int, int, int*, int, int)
        624                            inst_integer                    Integer Instructions       18944     5910528     2964736
```

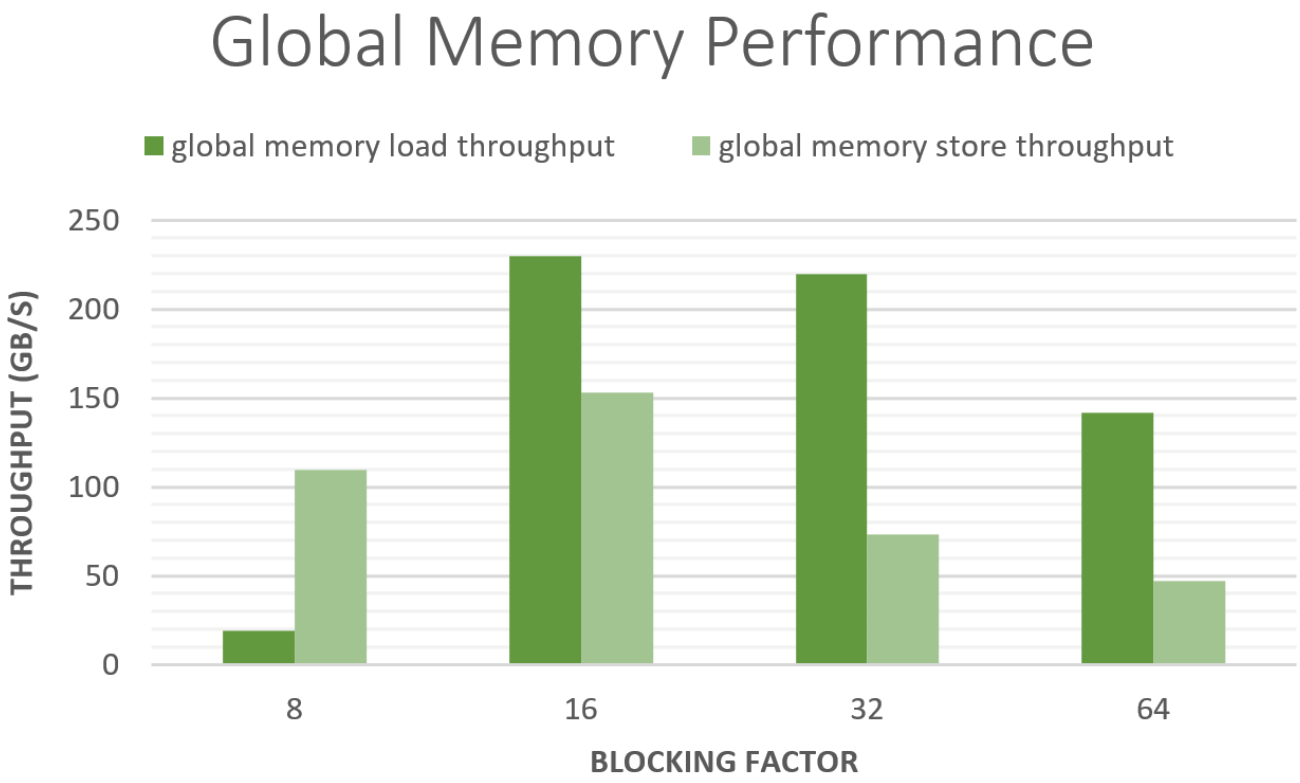
```
[pp20s61@hades02 block_16]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==746606== NVPROF is profiling process 746606, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
Execution took 832855us.
==746606== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==746606== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   89.94%  381.11ms      1246  305.87us  2.0480us  1.1762ms  phase3(int, int, int, int*, int, int)
                    3.65%  15.462ms         1  15.462ms  15.462ms  15.462ms  [CUDA memcpy HtoD]
                    3.59%  15.214ms         1  15.214ms  15.214ms  15.214ms  [CUDA memcpy DtoH]
                    1.30%  5.5265ms       624  8.8560us  2.8800us  16.288us  phase2_1(int, int, int, int*, int, int)
                    1.13%  4.7792ms       624  7.6580us  3.7760us  13.504us  phase2_2(int, int, int, int*, int, int)
                    0.39%  1.6521ms       313  5.2780us  4.4800us  6.7200us  phase1(int, int, int, int, int, int*)
      API calls:   43.86%  250.70ms      2817  88.993us     127ns  1.1585ms  cudaLaunchKernel
                   30.39%  173.71ms         2  86.853ms  15.495ms  158.21ms  cudaMemcpy
                   20.46%  116.94ms         1  116.94ms  116.94ms  116.94ms  cudaDeviceSetCacheConfig
                    3.16%  18.039ms         1  18.039ms  18.039ms  18.039ms  cudaHostAlloc
                    1.98%  11.337ms         1  11.337ms  11.337ms  11.337ms  cudaFreeHost
                    0.05%  277.15us         1  277.15us  277.15us  277.15us  cudaMalloc
                    0.04%  218.24us       101  2.1600us     158ns  93.315us  cuDeviceGetAttribute
                    0.04%  217.24us         1  217.24us  217.24us  217.24us  cuDeviceTotalMem
                    0.03%  145.75us         1  145.75us  145.75us  145.75us  cudaFree
                    0.00%  27.156us         1  27.156us  27.156us  27.156us  cuDeviceGetName
                    0.00%  3.0070us         1  3.0070us  3.0070us  3.0070us  cuDeviceGetPCIBusId
                    0.00%  1.7530us         3     584ns     304ns     934ns  cuDeviceGetCount
                    0.00%     962ns         2     481ns     225ns     737ns  cuDeviceGet
                    0.00%     368ns         1     368ns     368ns     368ns  cuDeviceGetUuid
```

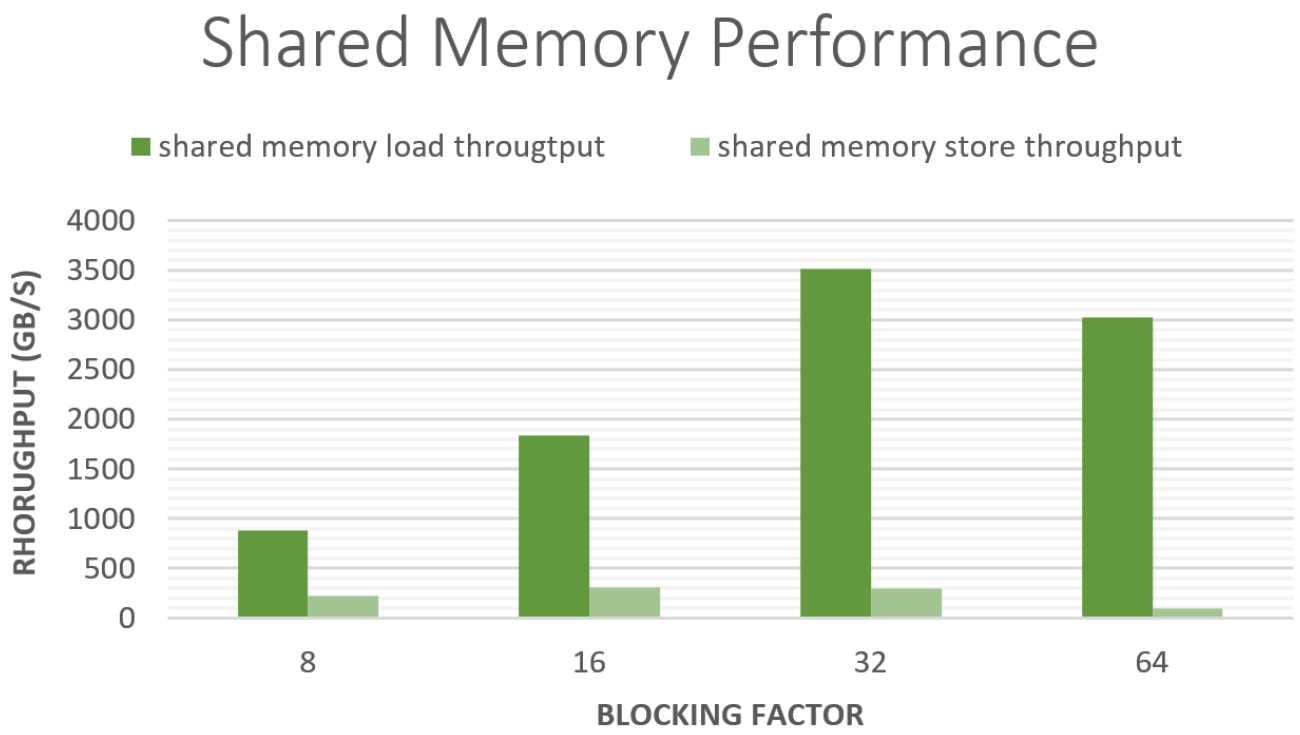
以上面兩張圖(block size = 16)為例，GOPS = 444461141 * 1246 / 0.38111 / 1024 / 1024 / 1024 ~= 1353.3235。

這幾個metrics都是取phase3的結果來比較(3個phase的結果放在report最後面的附件供參考)，而沒有跑更大一點的測資是因為block size = 8時，效能不是太好，會超過TIME LIMIT，導致hades會砍掉超時的nvprof，因此只有跑c21.1做代表。

Blocking factor	gld throughput (GB/s)	gst throughput (GB/s)	shared load throughtput (GB/s)	shared store throughput (GB/s)	GOPS
8	19.05	109.8	878.39	219.69	740.0995
16	230.08	153.28	1840.6	306.77	1353.3235
32	219.64	73.215	3514.3	292.86	1985.4961
64	141.77	47.257	3024.5	94.514	2431.1053







可發現global memory的performance在blocking factor = 16時是最好的，而shared memory的performance在blocking factor = 32時是大約最好的，而computation performance則是在blocking factor = 64時是最好的。

Optimization

The results below is based on running testcase p12k1.

1. GPU baseline: 將一開始的seq.cc改成可以在GPU跑。
2. Coalesced memory access: 修改code，使讀取global memory時是Coalesced地讀取。
3. Unroll: 使用`#pragma unroll`去unroll迴圈。
4. Shared memory: 將三個phase分開寫function做處理，根據不同的phase索取不同的shared memory大小，並將重複利用到的global memory load進shared memory中。
5. Modify if-branch to min: 把使用if判斷兩個值的大小去決定是否要修改最小值，改成取兩者之間的min。如下方程式碼所示。這樣的話可以減少diversity，讓wrap中的所有人都做同樣的事。

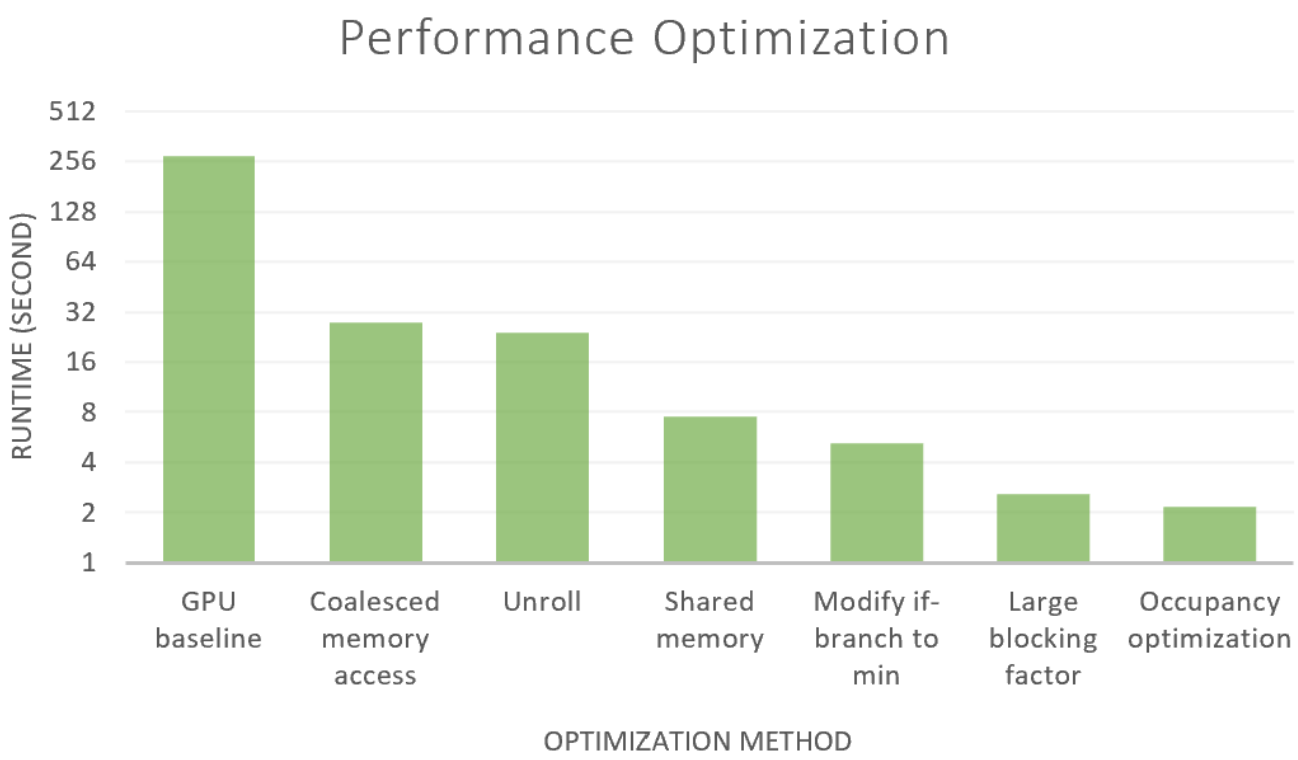
```
int diff = k_max - k_min;
int dist_i_j = Dist_ij[threadIdx.y][threadIdx.x];

/* original version */
#pragma unroll
for (int k = 0; k < diff; ++k){
    int val = Dist_ik[threadIdx.y][k] + Dist_kj[k][threadIdx.x];
    dist_i_j = val * (val < dist_i_j) + dist_i_j * (val >= dist_i_j);
}

/* optimized version */
#pragma unroll
for (int k = 0; k < diff; ++k){
    int val = Dist_ik[threadIdx.y][k]+Dist_kj[k][threadIdx.x];
    dist_i_j = min(val, dist_i_j);
}
```

6. Large blocking factor: 原先的block size是取32，將block size擴大成64，而thread大小維持(32, 32)不變，只是改成要做4遍的工作。
7. Occupancy optimization: 在Makefile中調整-maxrregcount="22"。

Optimization method	time (second)
GPU baseline	273.616392
Coalesced memory access	27.610206
Unroll	23.837973
Shared memory	7.482059
Modify if-branch to min	5.238779
Large blocking factor	2.582597
Occupancy optimization	2.177445



Experience & conclusion

What have you learned from this homework?

這次是除了lab以外，第一次寫cuda的程式，發現cuda真的有很多深奧的優化技巧，上課的時候聽老師講說不同的寫法，效能甚至可以差到幾十倍，那時候還覺得有點浮誇，真的會差那麼多嗎？直到自己實際跑了之後才發現光有沒有做coalesced memory access，就可以差到好幾倍了。

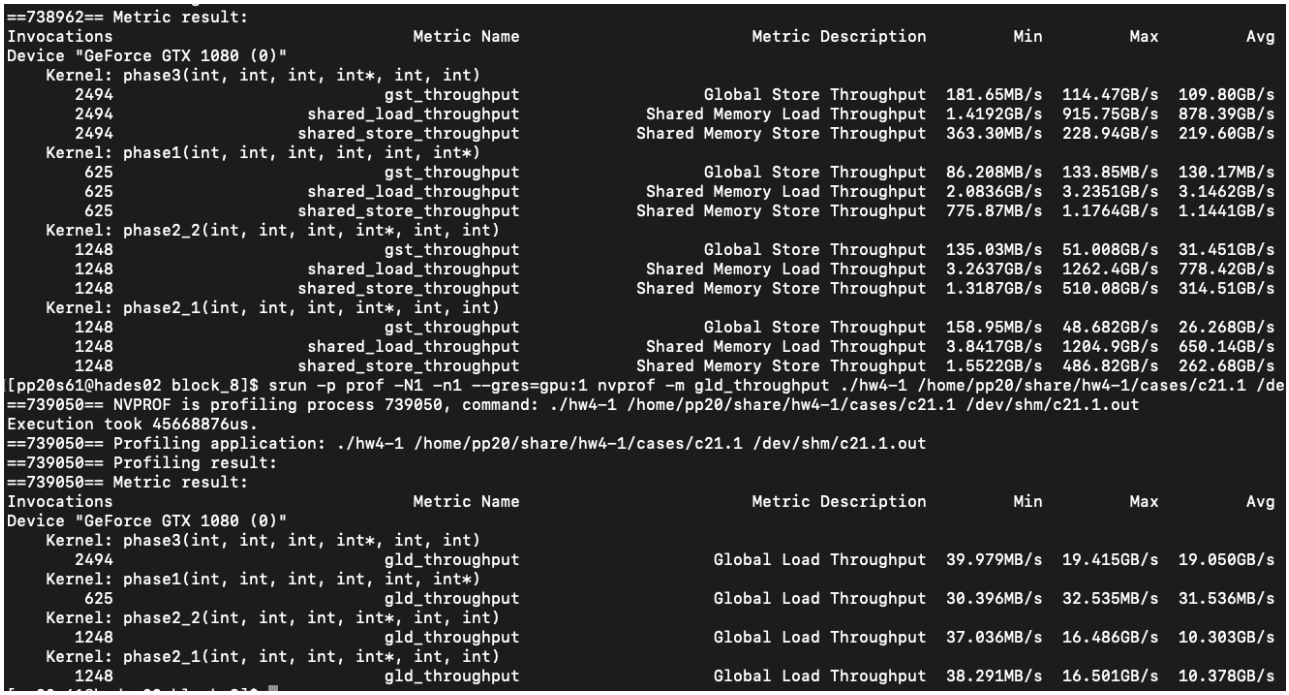
中間一度卡在block size = 32，那時2D API、Padding、Coalesced memory、Shared memory、Unroll都做了，卻一直卡在p24k1，直到後來聽到有人說將block size改成64，一次做4遍，會使效能變好，才又抱著希望去嘗試看看，幸好到最後還有繼續堅持寫下去(但大部分是抱持著這次沒有寫出來的話，hw4-2不知道會不會悲劇，所以拼命也要寫過XD)，才能打破很多以前的想法，學到新的東西。

就像老師說的，cuda的程式碼真的會隨著優化而使得可讀性變差，就算是前一天才打完的程式碼，過了一天去看還是要想一下自己到底在寫什麼，真的要保持思緒清楚才不會卡在奇怪的地方。

附件

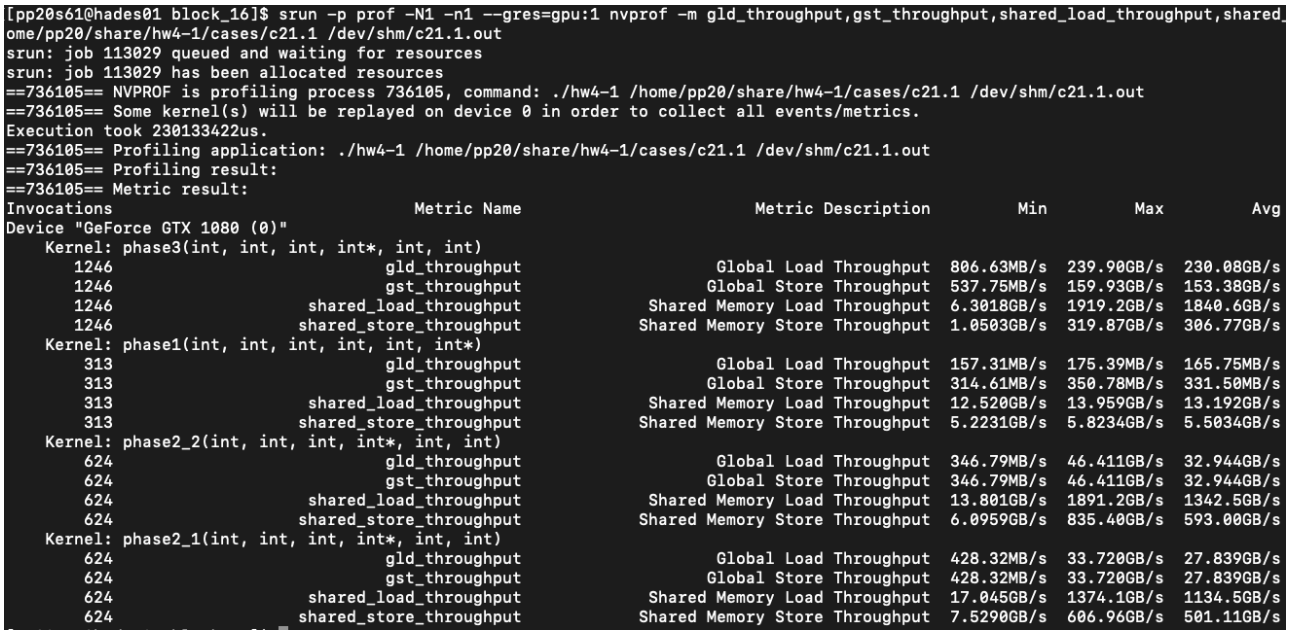
Global and Shared Memory Performance

- blocking factor = 8



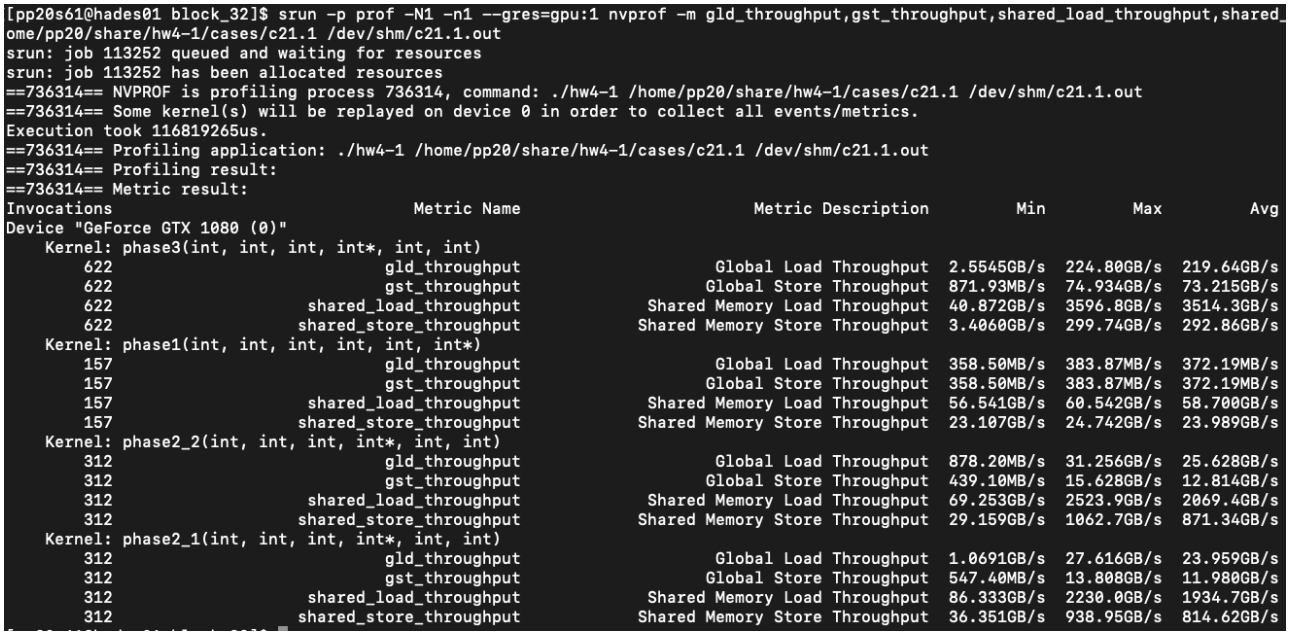
```
==738962== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
       2494                            gst_throughput                Global Store Throughput  181.65MB/s  114.47GB/s  109.80GB/s
       2494                     shared_load_throughput         Shared Memory Load Throughput  1.4192GB/s  915.75GB/s  878.39GB/s
       2494                    shared_store_throughput        Shared Memory Store Throughput  363.30MB/s  228.94GB/s  219.60GB/s
    Kernel: phase1(int, int, int, int, int, int*)
        625                            gst_throughput                Global Store Throughput  86.208MB/s  133.85MB/s  130.17MB/s
        625                     shared_load_throughput         Shared Memory Load Throughput  2.0836GB/s  3.2351GB/s  3.1462GB/s
        625                    shared_store_throughput        Shared Memory Store Throughput  775.87MB/s  1.1764GB/s  1.1441GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
       1248                            gst_throughput                Global Store Throughput  135.03MB/s  51.008GB/s  31.451GB/s
       1248                     shared_load_throughput         Shared Memory Load Throughput  3.2637GB/s  1262.4GB/s  778.42GB/s
       1248                    shared_store_throughput        Shared Memory Store Throughput  1.3187GB/s  510.08GB/s  314.51GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
       1248                            gst_throughput                Global Store Throughput  158.95MB/s  48.682GB/s  26.268GB/s
       1248                     shared_load_throughput         Shared Memory Load Throughput  3.8417GB/s  1204.9GB/s  650.14GB/s
       1248                    shared_store_throughput        Shared Memory Store Throughput  1.5522GB/s  486.82GB/s  262.68GB/s
[pp20s61@hades02 block_8]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m gld_throughput ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /de
==739050== NVPROF is profiling process 739050, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
Execution took 45668876us.
==739050== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==739050== Profiling result:
==739050== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
       2494                            gld_throughput                Global Load Throughput  39.979MB/s  19.415GB/s  19.050GB/s
    Kernel: phase1(int, int, int, int, int, int*)
        625                            gld_throughput                Global Load Throughput  30.396MB/s  32.535MB/s  31.536MB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
       1248                            gld_throughput                Global Load Throughput  37.036MB/s  16.486GB/s  10.303GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
       1248                            gld_throughput                Global Load Throughput  38.291MB/s  16.501GB/s  10.378GB/s
```

- blocking factor = 16



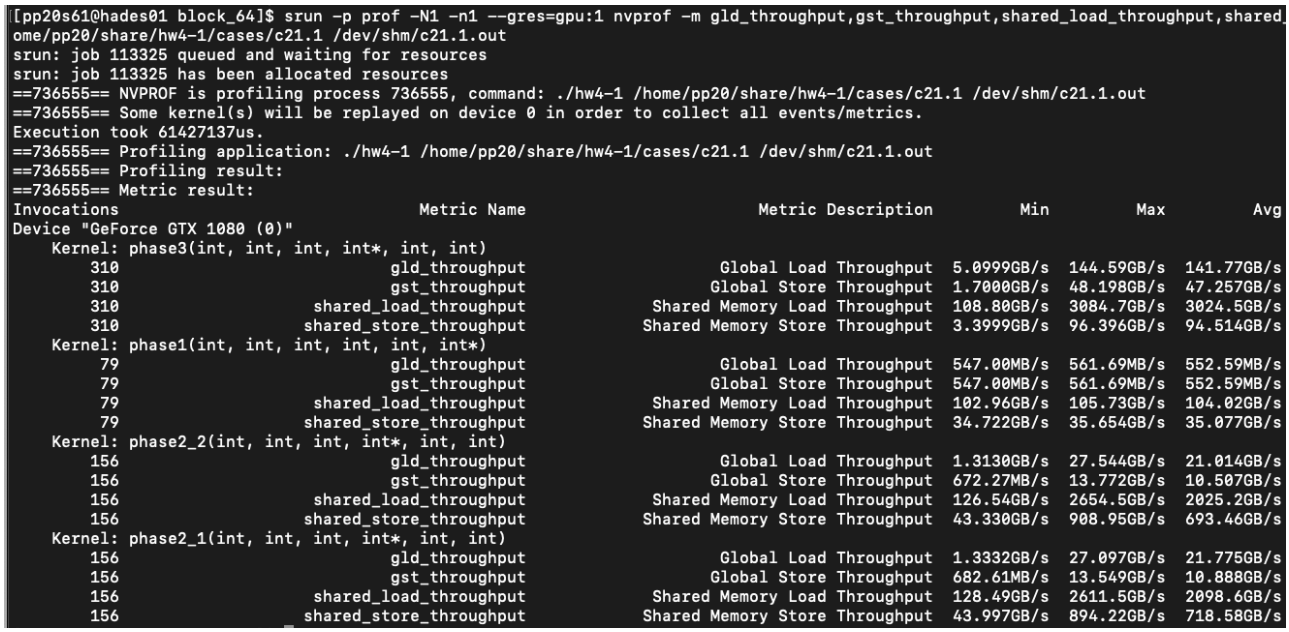
```
[pp20s61@hades01 block_16]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m gld_throughput,gst_throughput,shared_load_throughput,shared_
ome/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
srun: job 113029 queued and waiting for resources
srun: job 113029 has been allocated resources
==736105== NVPROF is profiling process 736105, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736105== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
Execution took 230133422us.
==736105== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736105== Profiling result:
==736105== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
       1246                            gld_throughput                Global Load Throughput  806.63MB/s  239.90GB/s  230.08GB/s
       1246                            gst_throughput               Global Store Throughput  537.75MB/s  159.93GB/s  153.38GB/s
       1246                     shared_load_throughput         Shared Memory Load Throughput  6.3018GB/s  1919.2GB/s  1840.6GB/s
       1246                    shared_store_throughput        Shared Memory Store Throughput  1.0503GB/s  319.87GB/s  306.77GB/s
    Kernel: phase1(int, int, int, int, int, int*)
        313                            gld_throughput                Global Load Throughput  157.31MB/s  175.39MB/s  165.75MB/s
        313                            gst_throughput               Global Store Throughput  314.61MB/s  350.78MB/s  331.50MB/s
        313                     shared_load_throughput         Shared Memory Load Throughput  12.520GB/s  13.959GB/s  13.192GB/s
        313                    shared_store_throughput        Shared Memory Store Throughput  5.2231GB/s  5.8234GB/s  5.5034GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
        624                            gld_throughput                Global Load Throughput  346.79MB/s  46.411GB/s  32.944GB/s
        624                            gst_throughput               Global Store Throughput  346.79MB/s  46.411GB/s  32.944GB/s
        624                     shared_load_throughput         Shared Memory Load Throughput  13.801GB/s  1891.2GB/s  1342.5GB/s
        624                    shared_store_throughput        Shared Memory Store Throughput  6.0959GB/s  835.40GB/s  593.00GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
        624                            gld_throughput                Global Load Throughput  428.32MB/s  33.720GB/s  27.839GB/s
        624                            gst_throughput               Global Store Throughput  428.32MB/s  33.720GB/s  27.839GB/s
        624                     shared_load_throughput         Shared Memory Load Throughput  17.045GB/s  1374.1GB/s  1134.5GB/s
        624                    shared_store_throughput        Shared Memory Store Throughput  7.5290GB/s  606.96GB/s  501.11GB/s
```

- blocking factor = 32



```
[pp20s61@hades01 block_32]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m gld_throughput,gst_throughput,shared_load_throughput,shared_
ome/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
srun: job 113252 queued and waiting for resources
srun: job 113252 has been allocated resources
==736314== NVPROF is profiling process 736314, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736314== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
Execution took 116819265us.
==736314== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736314== Profiling result:
==736314== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
        622                            gld_throughput                  Global Load Throughput  2.5545GB/s  224.80GB/s  219.64GB/s
        622                            gst_throughput                 Global Store Throughput  871.93MB/s  74.934GB/s  73.215GB/s
        622                     shared_load_throughput           Shared Memory Load Throughput  40.872GB/s  3596.8GB/s  3514.3GB/s
        622                    shared_store_throughput          Shared Memory Store Throughput  3.4060GB/s  299.74GB/s  292.86GB/s
    Kernel: phase1(int, int, int, int, int, int*)
        157                            gld_throughput                  Global Load Throughput  358.50MB/s  383.87MB/s  372.19MB/s
        157                            gst_throughput                 Global Store Throughput  358.50MB/s  383.87MB/s  372.19MB/s
        157                     shared_load_throughput           Shared Memory Load Throughput  56.541GB/s  60.542GB/s  58.700GB/s
        157                    shared_store_throughput          Shared Memory Store Throughput  23.107GB/s  24.742GB/s  23.989GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
        312                            gld_throughput                  Global Load Throughput  878.20MB/s  31.256GB/s  25.628GB/s
        312                            gst_throughput                 Global Store Throughput  439.10MB/s  15.628GB/s  12.814GB/s
        312                     shared_load_throughput           Shared Memory Load Throughput  69.253GB/s  2523.9GB/s  2069.4GB/s
        312                    shared_store_throughput          Shared Memory Store Throughput  29.159GB/s  1062.7GB/s  871.34GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
        312                            gld_throughput                  Global Load Throughput  1.0691GB/s  27.616GB/s  23.959GB/s
        312                            gst_throughput                 Global Store Throughput  547.40MB/s  13.808GB/s  11.980GB/s
        312                     shared_load_throughput           Shared Memory Load Throughput  86.333GB/s  2230.0GB/s  1934.7GB/s
        312                    shared_store_throughput          Shared Memory Store Throughput  36.351GB/s  938.95GB/s  814.62GB/s
```

- blocking factor = 64



```
[pp20s61@hades01 block_64]$ srun -p prof -N1 -n1 --gres=gpu:1 nvprof -m gld_throughput,gst_throughput,shared_load_throughput,shared_
ome/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
srun: job 113325 queued and waiting for resources
srun: job 113325 has been allocated resources
==736555== NVPROF is profiling process 736555, command: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736555== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
Execution took 61427137us.
==736555== Profiling application: ./hw4-1 /home/pp20/share/hw4-1/cases/c21.1 /dev/shm/c21.1.out
==736555== Profiling result:
==736555== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1080 (0)"
    Kernel: phase3(int, int, int, int*, int, int)
        310                            gld_throughput                  Global Load Throughput  5.0999GB/s  144.59GB/s  141.77GB/s
        310                            gst_throughput                 Global Store Throughput  1.7000GB/s  48.198GB/s  47.257GB/s
        310                     shared_load_throughput           Shared Memory Load Throughput  108.80GB/s  3084.7GB/s  3024.5GB/s
        310                    shared_store_throughput          Shared Memory Store Throughput  3.3999GB/s  96.396GB/s  94.514GB/s
    Kernel: phase1(int, int, int, int, int, int*)
         79                            gld_throughput                  Global Load Throughput  547.00MB/s  561.69MB/s  552.59MB/s
         79                            gst_throughput                 Global Store Throughput  547.00MB/s  561.69MB/s  552.59MB/s
         79                     shared_load_throughput           Shared Memory Load Throughput  102.96GB/s  105.73GB/s  104.02GB/s
         79                    shared_store_throughput          Shared Memory Store Throughput  34.722GB/s  35.654GB/s  35.077GB/s
    Kernel: phase2_2(int, int, int, int*, int, int)
        156                            gld_throughput                  Global Load Throughput  1.3130GB/s  27.544GB/s  21.014GB/s
        156                            gst_throughput                 Global Store Throughput  672.27MB/s  13.772GB/s  10.507GB/s
        156                     shared_load_throughput           Shared Memory Load Throughput  126.54GB/s  2654.5GB/s  2025.2GB/s
        156                    shared_store_throughput          Shared Memory Store Throughput  43.330GB/s  908.95GB/s  693.46GB/s
    Kernel: phase2_1(int, int, int, int*, int, int)
        156                            gld_throughput                  Global Load Throughput  1.3332GB/s  27.097GB/s  21.775GB/s
        156                            gst_throughput                 Global Store Throughput  682.61MB/s  13.549GB/s  10.888GB/s
        156                     shared_load_throughput           Shared Memory Load Throughput  128.49GB/s  2611.5GB/s  2098.6GB/s
        156                    shared_store_throughput          Shared Memory Store Throughput  43.997GB/s  894.22GB/s  718.58GB/s
```