Parallel Programming

All-Pairs Shortest Path in CUDA

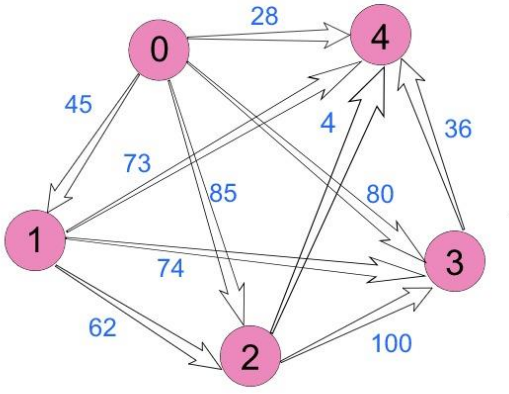
106062530 張原嘉

1. Implementation

Divide data、Communication、Configuration.

✓ Single GPU

Vertex 與 Weight 的相關資訊是以相鄰矩陣來表示，如下圖。有邊相連的則填邊上的值，自己到自己是 0，沒邊相連設為 INF (1e9)。



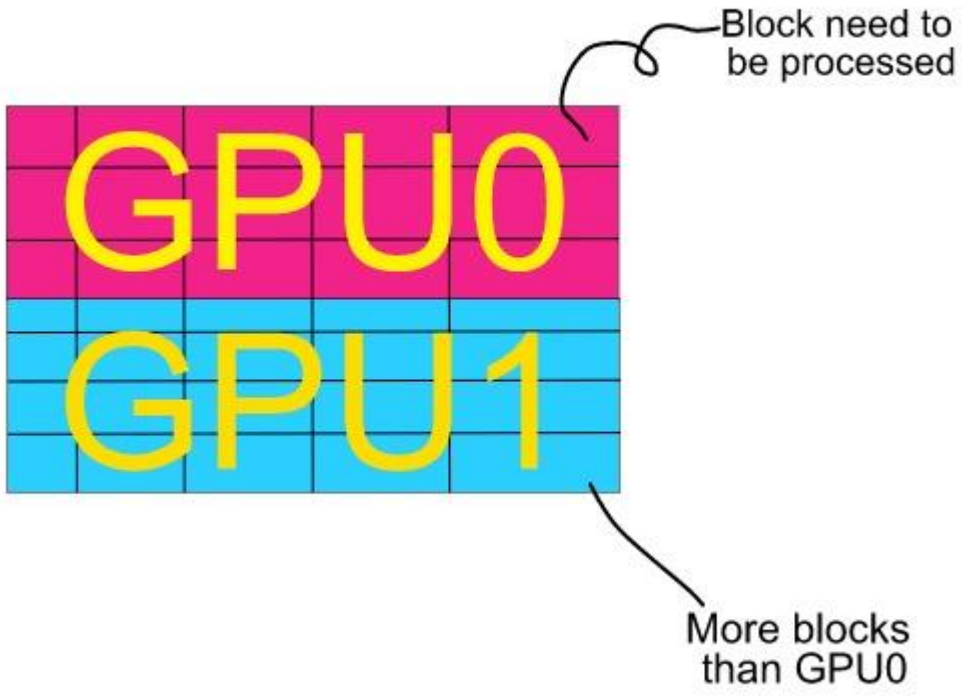
	0	1	2	3	4
0	0	45	85	80	28
1	INF	0	62	74	73
2	INF	INF	0	100	4
3	INF	INF	INF	0	36
4	INF	INF	INF	INF	0

但在程式的資料結構中，是用一維 Row-major 方式來儲存這些資訊，因為後續計算較方便。在 Grid Dimension 部分，由於 Block Factor 是使用者設定同時為了讓 GPU 的 thread 有更好的利用度，Block Size 的計算方式為 Block Factor * Block Factor。又，Block Factor 可用來計算 rounds (程式執行次數)，具體關係為：rounds = (vertexNum + Block Factor- 1) / Block Factor。而這個變數 rounds，亦能用來確立 3 個

phase，block 之 dimension。而 type 為 dim3 的變數 grid1、gird2、grid3，其 X 、 Y 兩向量分別為（1,1）、（rounds,2）、（rounds,rounds）。各變數關係整理如下：

✅ Multi GPU（OpenMP）

為了取得多 GPU 並行計算的效益，將資料都分配到參與運算的 GPU 是理所當然的。首先 cudaMalloc 把資料量為 vertexNum* vertexNum 皆分配給參與運算之 GPU，接著根據可用 GPU（此次作業固定為 2）、thread 數量（OpenMP 時）/process 數量（MPI 時）算出工作量與 Offset（用 location 函式定位），多出來的工作則丟給第二個 GPU，具體可能分配情況如下圖：



Host 會將切割好的資料大小分配給 GPU 0 、 1 (以 Row-major 方式)，待其計算完自己的部分後先暫存至自己的 device_matrix (二維陣列，可同時儲存 threadID 及資料)，待 GPU 全部完成再一次 copy 回 host (cudaMemcpyDeviceToHost)。

✅ **Multi GPU (MPI)**

MPI 版本與 OpenMP 雷同，惟 threadID 換成 rankID，以及溝通方面採用先前所用 MPI_Send/MPI_Recv，收取最後結果則用 MPI_Sendrecv。

2. Profiling Results

✅ **Single GPU**

```
==11669== Profiling application: ./HW4_cuda 3.in 3.out 32
==11669== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   97.27%  2.42191s       125  19.375ms  16.762ms  21.441ms  phase3(int, int, int*, int)
                    1.94%  48.340ms       125  386.72us  301.32us  1.6299ms  phase2(int, int, int*, int)
                    0.38%  9.3467ms         1  9.3467ms  9.3467ms  9.3467ms  [CUDA memcpy HtoD]
                    0.21%  5.3020ms       125  42.415us  40.256us  285.38us  phase1(int, int, int*, int)
                    0.20%  5.0862ms         1  5.0862ms  5.0862ms  5.0862ms  [CUDA memcpy DtoH]
      API calls:   81.10%  2.47211s         5  494.42ms  3.2780us  2.47202s  cudaEventSynchronize
                   18.15%  553.30ms         4  138.33ms  1.7070us  297.35ms  cudaEventCreate
                    0.53%  16.065ms         2  8.0325ms  6.5926ms  9.4725ms  cudaMemcpy
                    0.12%  3.5437ms       375  9.4490us  8.8220us  53.028us  cudaLaunch
                    0.04%  1.1824ms       188  6.2890us     314ns  262.61us  cuDeviceGetAttribute
                    0.03%  837.93us         1  837.93us  837.93us  837.93us  cudaMalloc
                    0.01%  398.44us         2  199.22us  99.030us  299.41us  cuDeviceTotalMem
                    0.01%  358.05us      1500     238ns     197ns  8.3530us  cudaSetupArgument
                    0.01%  190.17us         1  190.17us  190.17us  190.17us  cudaFree
                    0.00%  141.61us       375     377ns     355ns  1.7860us  cudaConfigureCall
                    0.00%  122.58us         2  61.289us  52.548us  70.030us  cuDeviceGetName
                    0.00%  89.460us        11  8.1320us  2.7710us  21.992us  cudaEventRecord
                    0.00%  19.518us         5  3.9030us  1.2390us  5.6620us  cudaEventElapsedTime
                    0.00%  4.9050us         3  1.6350us     394ns  4.0510us  cuDeviceGetCount
                    0.00%  3.2520us         1  3.2520us  3.2520us  3.2520us  cudaSetDevice
                    0.00%  2.7040us         4     676ns     358ns  1.4890us  cuDeviceGet
```

✓ Multi GPU (OpenMP)

```
==11908== Profiling application: ./HW4_openmp 3.in 3.out 32
==11908== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   81.30%  225.55ms       250  902.19us  825.70us  969.29us  FW_phase3(int*, int, int, int, int, int)
                    8.60%  23.865ms       252  94.701us  42.112us  5.2919ms  [CUDA memcpy HtoD]
                    5.37%  14.896ms       127  117.29us  38.496us  4.8209ms  [CUDA memcpy DtoH]
                    4.18%  11.589ms       250  46.357us  41.728us  50.336us  FW_phase2(int*, int, int, int, int)
                    0.56%  1.5402ms       250  6.1600us  5.4400us  6.6560us  FW_phase1(int*, int, int, int, int)
      API calls:   29.30%  265.93ms         1  265.93ms  265.93ms  265.93ms  cudaMallocHost
                   27.52%  249.84ms         2  124.92ms  510.99us  249.33ms  cudaMalloc
                   27.34%  248.15ms       379  654.76us  50.933us  5.6616ms  cudaMemcpy
                   13.23%  120.13ms       500  240.25us  6.5470us  2.3522ms  cudaDeviceSynchronize
                    1.50%  13.657ms         1  13.657ms  13.657ms  13.657ms  cudaFreeHost
                    0.65%  5.9189ms       750  7.8910us  4.7140us  43.859us  cudaLaunch
                    0.18%  1.5895ms         2  794.73us  787.52us  801.94us  cudaGetDeviceProperties
                    0.08%  730.69us       188  3.8860us     121ns  172.93us  cuDeviceGetAttribute
                    0.08%  729.71us         2  364.85us  249.85us  479.86us  cudaFree
                    0.06%  576.63us      4000     144ns      96ns  11.075us  cudaSetupArgument
                    0.02%  177.18us       750     236ns     118ns  1.1090us  cudaConfigureCall
                    0.02%  172.88us         2  86.439us  47.807us  125.07us  cuDeviceTotalMem
                    0.01%  74.819us         2  37.409us  30.390us  44.429us  cuDeviceGetName
                    0.00%  36.312us         8  4.5390us  2.6080us  7.6820us  cudaSetDevice
                    0.00%  2.8620us         3     954ns     145ns  2.4700us  cuDeviceGetCount
                    0.00%  1.9340us         1  1.9340us  1.9340us  1.9340us  cudaGetDeviceCount
                    0.00%  1.4710us         4     367ns     150ns     783ns  cuDeviceGet
```

✓ Multi GPU (MPI)

```
==14413== Profiling application: ./HW4_mpi 3.in 3.out 32
==14412== Profiling result:
==14413== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   77.17%  118.43ms       125  947.43us  883.08us  992.68us  FW_phase3(int*, int, int, int, int, int)
                    9.78%  15.015ms       126  119.17us  79.521us  5.0456ms  [CUDA memcpy HtoD]
                    8.47%  13.002ms        64  203.16us  75.904us  8.0869ms  [CUDA memcpy DtoH]
                    4.04%  6.2029ms       125  49.622us  45.857us  55.777us  FW_phase2(int*, int, int, int, int)
                    0.53%  819.30us       125  6.5540us  6.1440us  6.9760us  FW_phase1(int*, int, int, int, int)
 GPU activities:   77.14%  116.25ms       125  929.99us  869.16us  973.19us  FW_phase3(int*, int, int, int, int, int)
                    9.92%  14.956ms       126  118.70us  79.488us  4.9689ms  [CUDA memcpy HtoD]
                    8.23%  12.411ms        63  196.99us  76.032us  7.5825ms  [CUDA memcpy DtoH]
                    4.16%  6.2679ms       125  50.143us  45.409us  53.985us  FW_phase2(int*, int, int, int, int)
                    0.55%  824.93us       125  6.5990us  6.1120us  7.0400us  FW_phase1(int*, int, int, int, int)
      API calls:   39.87%  186.95ms         1  186.95ms  186.95ms  186.95ms  cudaMallocHost
                   31.55%  147.94ms       190  778.64us  87.862us  9.0594ms  cudaMemcpy
                   26.15%  122.62ms       250  490.47us  15.773us  1.0915ms  cudaDeviceSynchronize
                    1.57%  7.3574ms         1  7.3574ms  7.3574ms  7.3574ms  cudaFreeHost
                    0.40%  1.8884ms       375  5.0350us  4.0300us  33.248us  cudaLaunch
                    0.14%  645.91us         1  645.91us  645.91us  645.91us  cudaFree
                    0.13%  621.03us        94  6.6060us     114ns  246.59us  cuDeviceGetAttribute
                    0.07%  311.85us         1  311.85us  311.85us  311.85us  cudaMalloc
                    0.05%  243.02us      2000     121ns      90ns  4.7080us  cudaSetupArgument
                    0.05%  213.01us         1  213.01us  213.01us  213.01us  cuDeviceTotalMem
                    0.01%  65.326us       375     174ns     130ns     768ns  cudaConfigureCall
                    0.01%  31.115us         1  31.115us  31.115us  31.115us  cuDeviceGetName
                    0.00%  5.0170us         1  5.0170us  5.0170us  5.0170us  cudaSetDevice
                    0.00%  2.5720us         3     857ns     155ns  2.2490us  cuDeviceGetCount
                    0.00%  1.0840us         1  1.0840us  1.0840us  1.0840us  cudaGetDeviceProperties
                    0.00%     480ns         2     240ns     142ns     338ns  cuDeviceGet
      API calls:   41.32%  196.93ms         1  196.93ms  196.93ms  196.93ms  cudaMallocHost
                   30.51%  145.40ms       189  769.34us  88.211us  9.6228ms  cudaMemcpy
                   26.05%  124.16ms       250  496.63us  15.720us  1.1052ms  cudaDeviceSynchronize
                    1.30%  6.2173ms         1  6.2173ms  6.2173ms  6.2173ms  cudaFreeHost
                    0.40%  1.8930ms       375  5.0480us  4.1440us  21.871us  cudaLaunch
                    0.11%  546.28us        94  5.8110us     109ns  245.20us  cuDeviceGetAttribute
                    0.08%  360.51us         1  360.51us  360.51us  360.51us  cudaGetDeviceProperties
                    0.06%  299.93us         1  299.93us  299.93us  299.93us  cudaMalloc
                    0.05%  221.63us      2000     110ns      91ns  1.4380us  cudaSetupArgument
                    0.04%  213.04us         1  213.04us  213.04us  213.04us  cuDeviceTotalMem
                    0.04%  172.57us         1  172.57us  172.57us  172.57us  cudaFree
                    0.03%  120.18us         1  120.18us  120.18us  120.18us  cuDeviceGetName
                    0.01%  66.220us       375     176ns     145ns     463ns  cudaConfigureCall
                    0.00%  20.137us         1  20.137us  20.137us  20.137us  cudaSetDevice
                    0.00%  1.9590us         3     653ns     162ns  1.5840us  cuDeviceGetCount
                    0.00%     529ns         2     264ns     143ns     386ns  cuDeviceGet
```

3. Experiment & Analysis

System Spec、Weak Scalability & Time Distribution、Blocking Factor、Others

✅ System Spec（皆使用課程提供之設備）

GPU 資訊

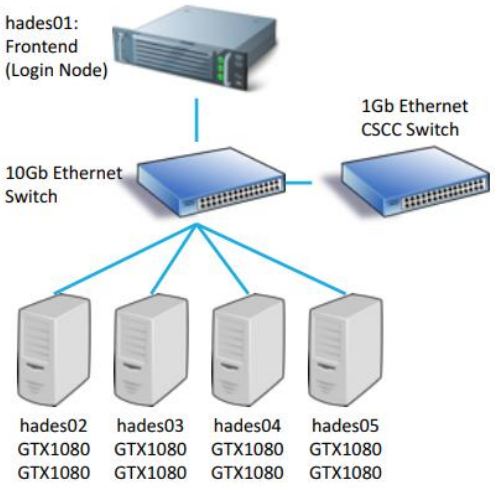
```
[p106062530@hades02 hw]$ nvidia-smi
Wed Dec 27 20:49:58 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.81                 Driver Version: 384.81                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    On   | 00000000:4B:00.0 Off |                  N/A |
| 57%   70C    P2   212W / 216W |   7956MiB /  8114MiB |     87%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    On   | 00000000:4D:00.0 Off |                  N/A |
|  1%   47C    P8    15W / 216W |      2MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      3175      C   python                                     7945MiB |
+-----------------------------------------------------------------------------+
```

CPU 資訊

```
[p106062530@hades02 hw]$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz
stepping        : 1
microcode       : 0xb00001d
cpu MHz         : 1710.421
cache size      : 15360 KB
physical id     : 0
siblings        : 12
core id         : 0
cpu cores       : 6
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 20
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon peb
s bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer
 aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch ida arat epb pln pts dtherm intel_pt tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm rds
eed adx smap xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local
bogomips        : 7199.73
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

Network 資訊

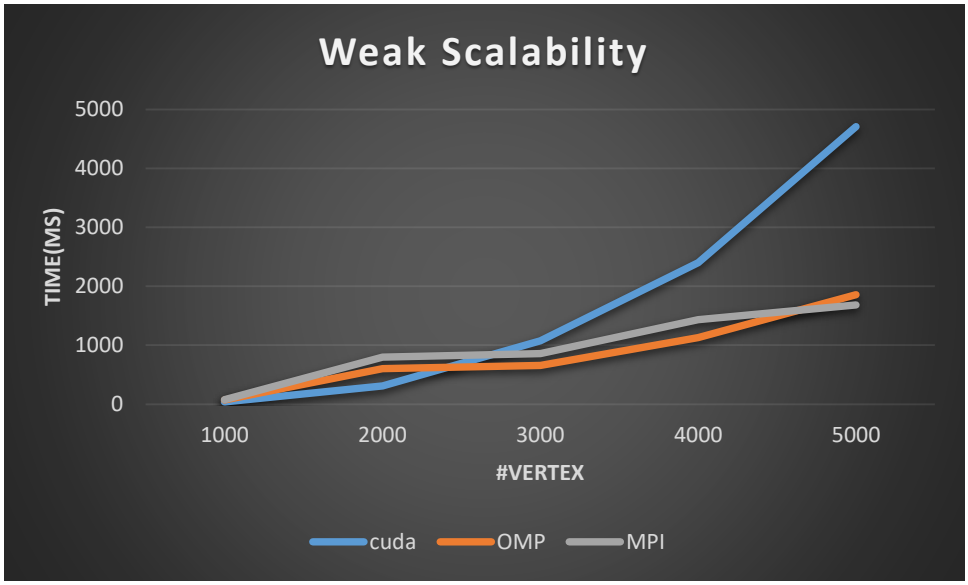


(引用自 PP2017_Lab3_tutorial slides)

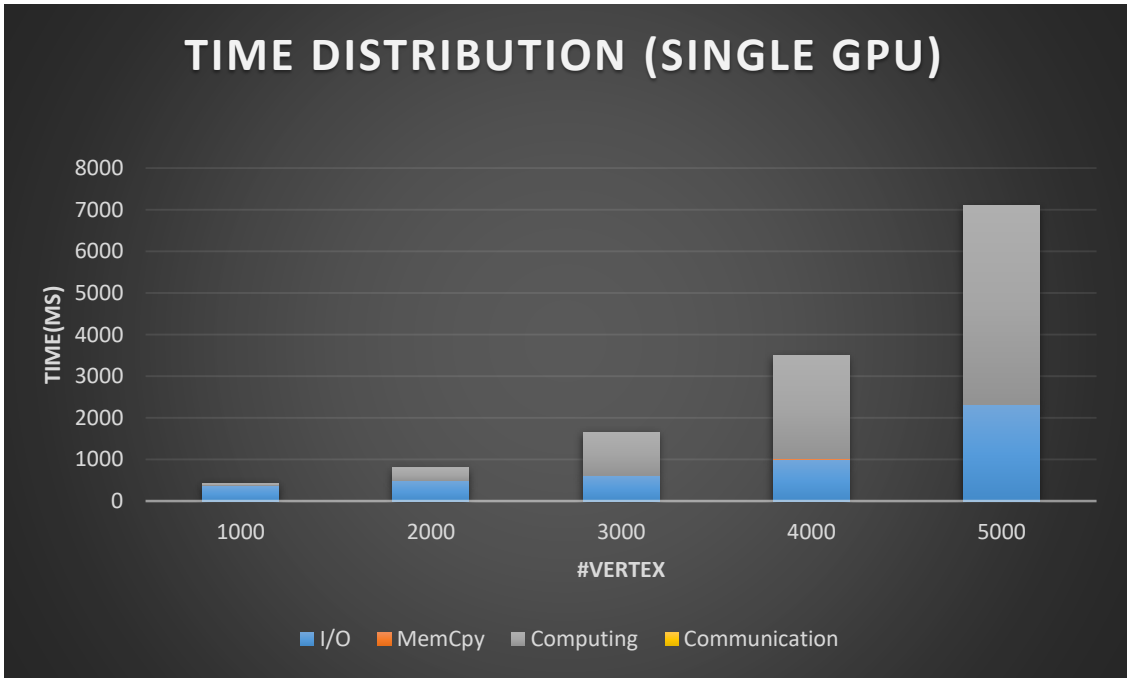
✓ Time Measurement

在需要測量時間的程式碼前後，插入 cudaEvent，再利用 cudaEventElapsedTime 得到所需時間，其餘測量方法將於稍後解釋圖時一併說明。

✅ Weak Scalability & Time Distribution

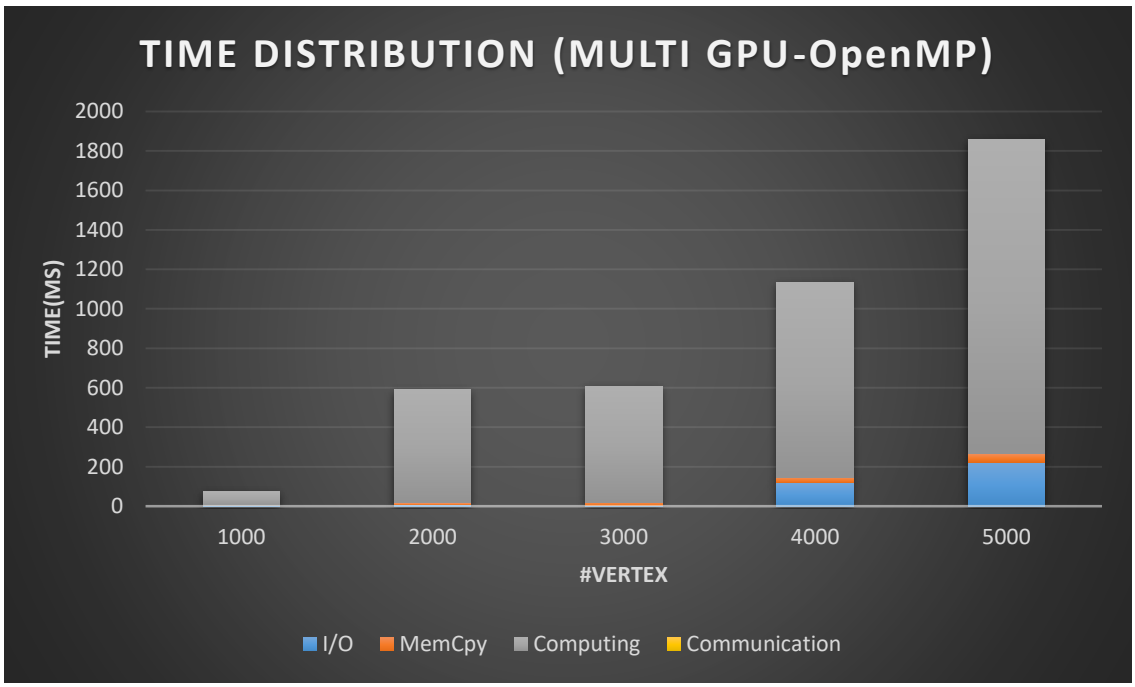


由於是小測資，若以宏觀來看其實三個版本的執行時間都很小，但由於 Multi-GPU 的關係，OMP 和 MPI 的執行時間還是會比 Single 版本快一些，OMP 又比 MPI 快，因為 MPI 需要跨 Node 的溝通。

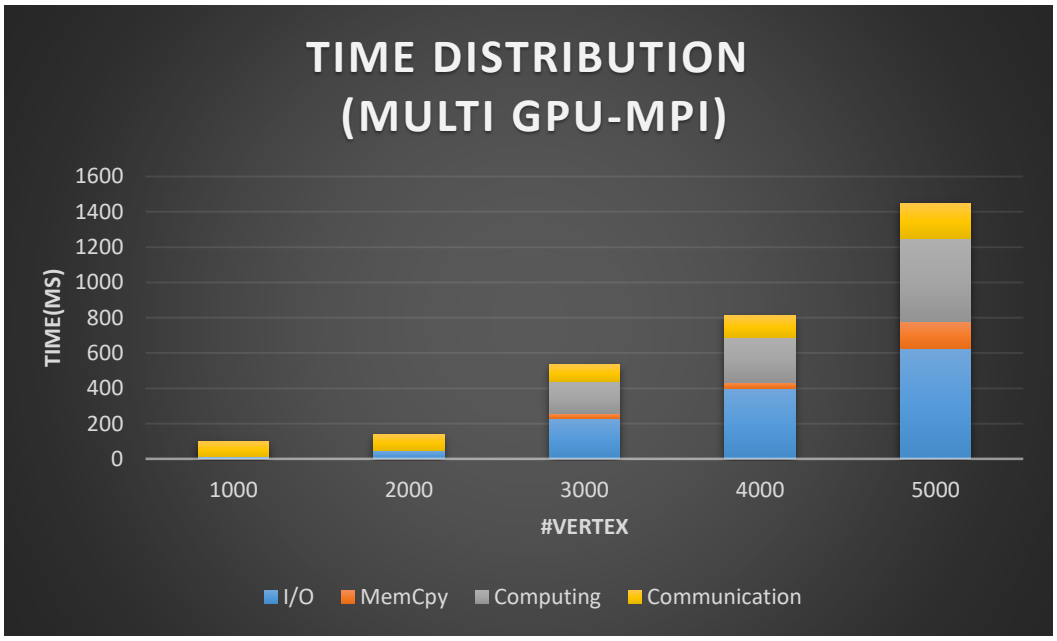


Single GPU 版本因不須溝通，所以 Communication Time 都是 0。接著由圖可看

出，當點的數目增加，Computing Time 和 I/O Time 會逐漸成為 Bottleneck。



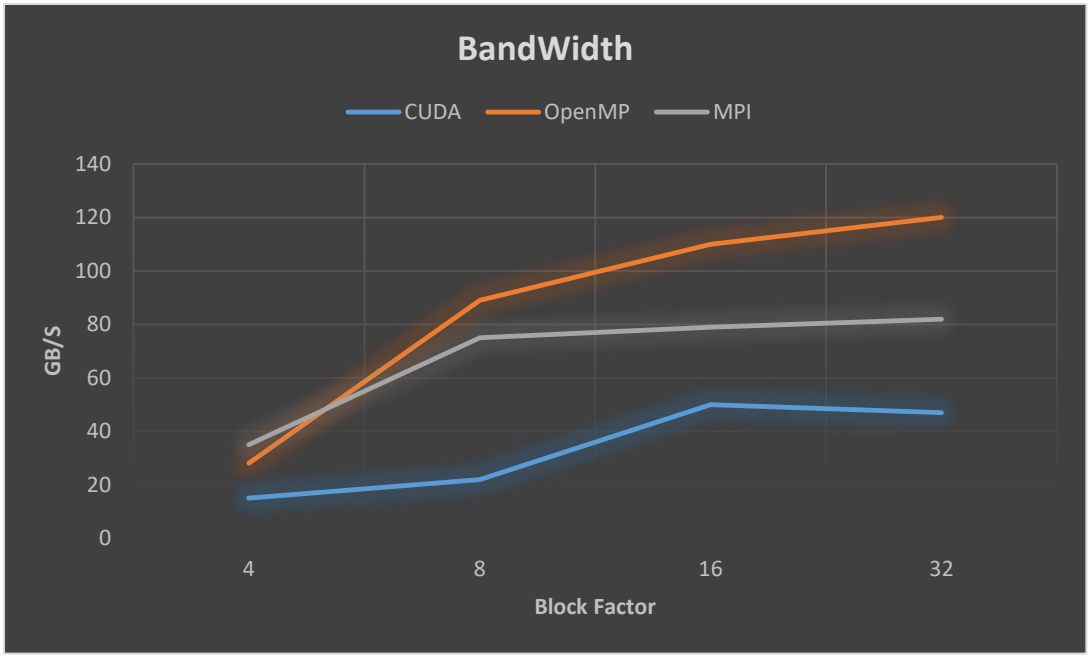
OMP 版本由於溝通主要是 CPU 內的 thread 溝通，所以 Comm Time 也是 0 。隨著時間增加，Computing 成為最大的 Bottleneck。

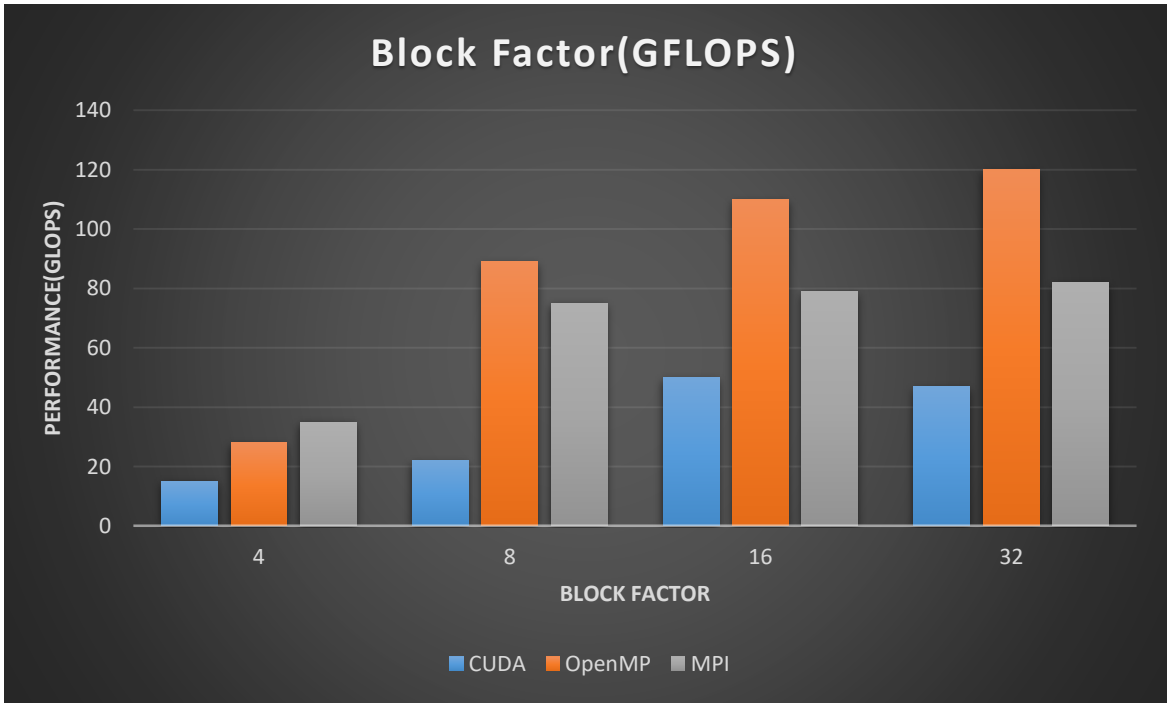


此版本多了跨 Node 的溝通，因此在小測資時，Communication Time 幾乎占了所有執行時間，然而隨著點的數量增加，跨Node、Multi GPU 的性質逐漸產生效益。因此次作業 Process 數最多兩個，Communication Time 不會成為主要的 Bottleneck，演算法的設計、時

做，才是此次作業執行時間的關鍵。

✓ Block Factor-Glops/Bandwidth





Cuda 中每個 thread 是以 32 為一單位 wrap 運作，因此 block size 若為 32 較能取得效率。由圖可看出三種版本最佳 Performance 的 blocking factor 分別為 16。32、16 （Cuda/OpenMP/MPI）。而 Bandwidth 差距有些大有些小，推測是因為 bank conflict 的問題所導致。

4. Experience & Conclusion

這次作業學到的是嶄新的東西 - Cuda，儘管學習過程跌跌撞撞，但著實了解到 GPU 在重複運算中的威力。期末將至，還有許多科目要顧，如果有更多時間，我想我的這份作業還有很大的改善空間。