HW5 – Report

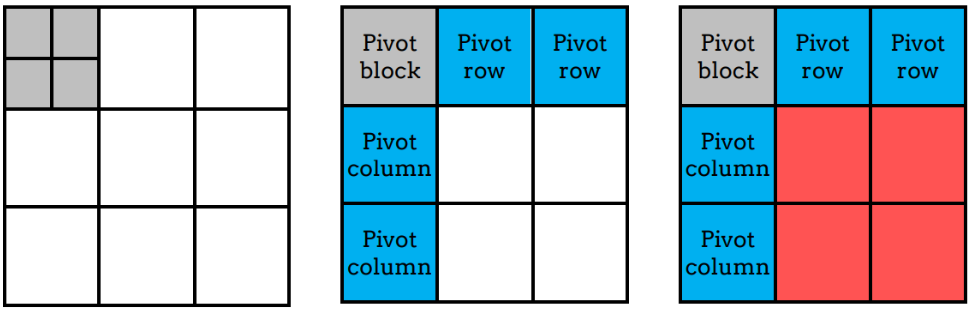
1. **Name:** 徐嘉駿, **Institute:** 資應所, **Student ID:** 107065528

2. Implement

- **Divide Data:** 如同這次作業 spec 的切割方法，將 data 切成 blocks，以下為 configuration:
  (1) **Blocking Factor:** 32
  (2) **Blocks:** (data 量/32)^2
  (3) **Threads:** 32*32
- **Implementation:** 如同這次作業 spec 的實作方法，將過程分為好幾 rounds，每 round 分為 3 個 phases:

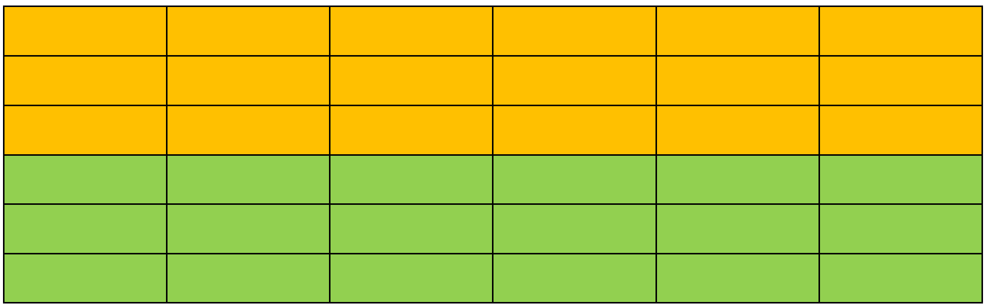
```
for (int r = 0; r < round; ++r) {
    phase1<<<grid1, blk, B*B*sizeof(int)>>>(r, n, V, d_Dist, B);
    phase2<<<grid2, blk, 2*B*B*sizeof(int)>>>(r, n, V, d_Dist, B);
    phase3<<<grid3, blk, 2*B*B*sizeof(int)>>>(r, n, V, d_Dist, B);
}
```

(1) **Phase 1:** 運算 pivot block
(2) **Phase 2:** 運算 pivot row blocks & pivot column blocks
(3) **Phase 3:** 運算剩下的 blocks



(a) Phase 1 (b) Phase 2 (c) Phase 3

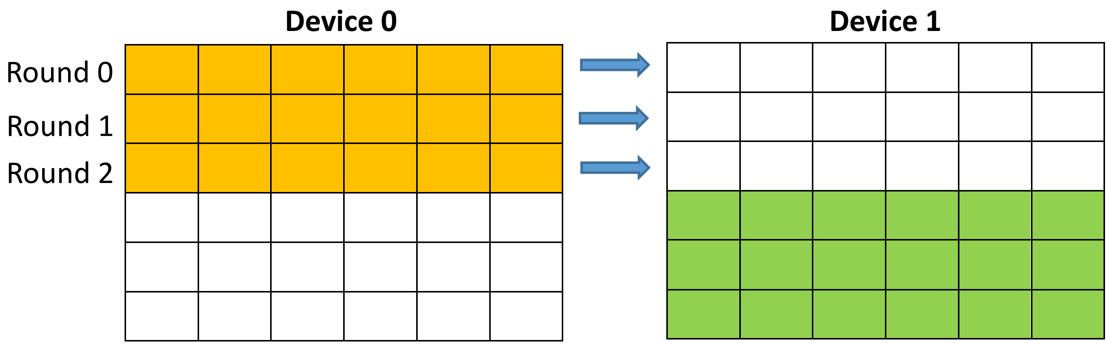
Multi-GPUs: 由於透過 single GPU 實驗時得知在 phase3 時要做最久，所以將 phase3 的 n * n 矩陣切成上下兩塊，device 0 做上半部，device 1 做下半部，如圖所示(假設 6x6 matrix):



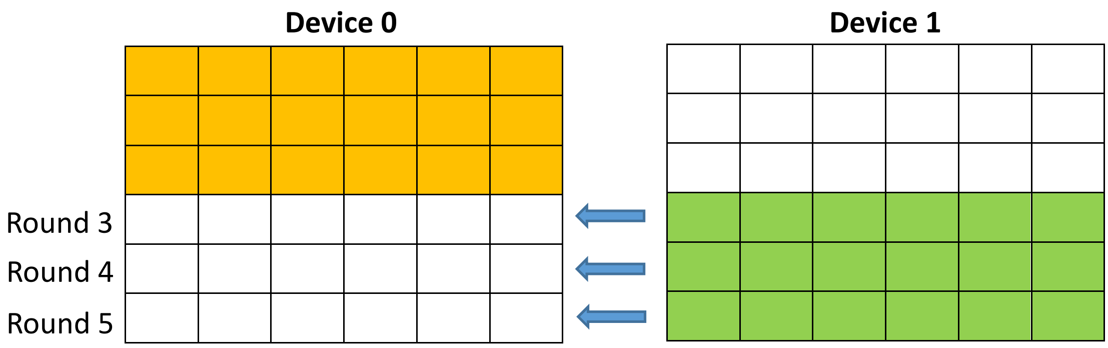
Device 0 做黃色部分的 data，device 1 做綠色部分。

```
dim3 grid1(1, 1);
dim3 grid2(round, 2);
dim3 grid3((round/2)+1, round);
```

Communication: 在每一 round 開始之前，先將 pivot row 傳給需要更新的 device，如上例所示(6x6 matrix)，在前 3 round 時，device 1 只需要 device 0 的 pivot row data，所以每 round 將 pivot row 由 device 0 傳給 device 1，如下圖所示:



後 3 round 時，device 0 需要 device 1 的 pivot row data，所以每 round 將 pivot row 由 device 1 傳給 device 0，如下圖所示:



3. Profiling Results

使用 p20k1 做 measurement

以下為 *nvprof* 測量之各項結果:

- **Single GPU:**

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	96.69%	21.7946s	625	34.871ms	33.840ms	35.213ms	phase3(int, int, int, int*, int)
	1.36%	305.92ms	2	152.96ms	544ns	305.92ms	[CUDA memcpy HtoD]
	1.33%	299.15ms	1	299.15ms	299.15ms	299.15ms	[CUDA memcpy DtoH]
	0.61%	137.67ms	625	220.27us	212.00us	224.19us	phase2(int, int, int, int*, int)
	0.01%	3.2451ms	625	5.1920us	4.9920us	9.2480us	phase1(int, int, int, int*, int)
API calls:	55.50%	12.5968s	3	4.19892s	130.00us	12.2907s	cudaMemcpy
	43.81%	9.94271s	1875	5.3028ms	3.6400us	35.209ms	cudaLaunch
	0.68%	154.66ms	2	77.330ms	147.04us	154.51ms	cudaMalloc
	0.00%	1.1177ms	188	5.9450us	161ns	246.64us	cuDeviceGetAttribute
	0.00%	980.61us	9375	104ns	83ns	12.798us	cudaSetupArgument
	0.00%	327.83us	2	163.91us	163.38us	164.45us	cuDeviceTotalMem
	0.00%	263.35us	1875	140ns	115ns	3.5860us	cudaConfigureCall
	0.00%	110.91us	2	55.455us	54.243us	56.668us	cuDeviceGetName
	0.00%	1.9790us	3	659ns	204ns	1.4930us	cuDeviceGetCount
	0.00%	1.3710us	4	342ns	175ns	739ns	cuDeviceGet

- **Multiple GPUs:**

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	92.69%	22.4057s	1250	17.925ms	17.388ms	19.335ms	phase3(int, int, int, int*, int, int, int)
	3.71%	897.08ms	629	1.4262ms	768ns	361.39ms	[CUDA memcpy HtoD]
	2.43%	588.11ms	627	937.98us	402.98us	163.70ms	[CUDA memcpy DtoH]
	1.14%	276.00ms	1250	220.80us	211.43us	242.91us	phase2(int, int, int, int*, int)
	0.03%	6.5347ms	1250	5.2270us	4.9600us	12.928us	phase1(int, int, int, int*, int)
API calls:	55.34%	8.99349s	625	14.390ms	18.323us	2.75211s	cudaMemcpyPeer
	41.39%	6.72551s	6	1.12092s	9.6970us	3.04213s	cudaMemcpy
	3.07%	499.52ms	4	124.88ms	252.03us	249.53ms	cudaMalloc
	0.15%	24.782ms	3750	6.6080us	3.8350us	428.90us	cudaLaunch
	0.02%	3.7812ms	21250	177ns	84ns	389.80us	cudaSetupArgument
	0.01%	2.2733ms	3750	606ns	132ns	15.530us	cudaConfigureCall
	0.01%	1.1182ms	188	5.9470us	163ns	248.18us	cuDeviceGetAttribute
	0.00%	327.90us	2	163.95us	162.91us	164.99us	cuDeviceTotalMem
	0.00%	106.46us	2	53.232us	50.049us	56.415us	cuDeviceGetName
	0.00%	21.112us	2	10.556us	4.9160us	16.196us	cudaSetDevice
	0.00%	1.8940us	3	631ns	195ns	1.4690us	cuDeviceGetCount
	0.00%	1.2090us	4	302ns	156ns	664ns	cuDeviceGet

結論: 在 single GPU & multiple GPUs 裡，可以看到 multiple GPUs 在 phase3 明顯比 single GPU 少(multiple GPUs phase3 time 必須除以 2，因為 nvprof 會將兩個 device time 相加)，所以 computing time 明顯下降許多。此減少的時間變為增加 communication (CudaMemcpyPeer)的時間。

4. Experiment & Analysis

- **Time Distribution:**

使用以下 test cases 做 measurement:

p11k1: vector ->11000, edge->505586

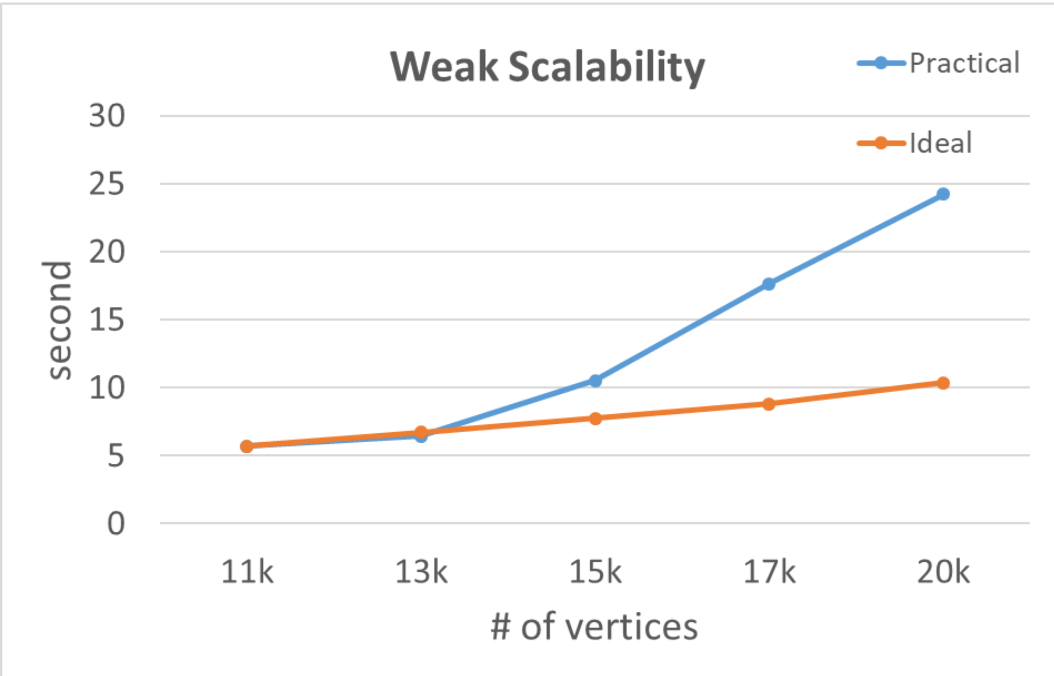
p13k1: vector ->13000, edge->1829967

p15k1: vector ->15000, edge->5591272

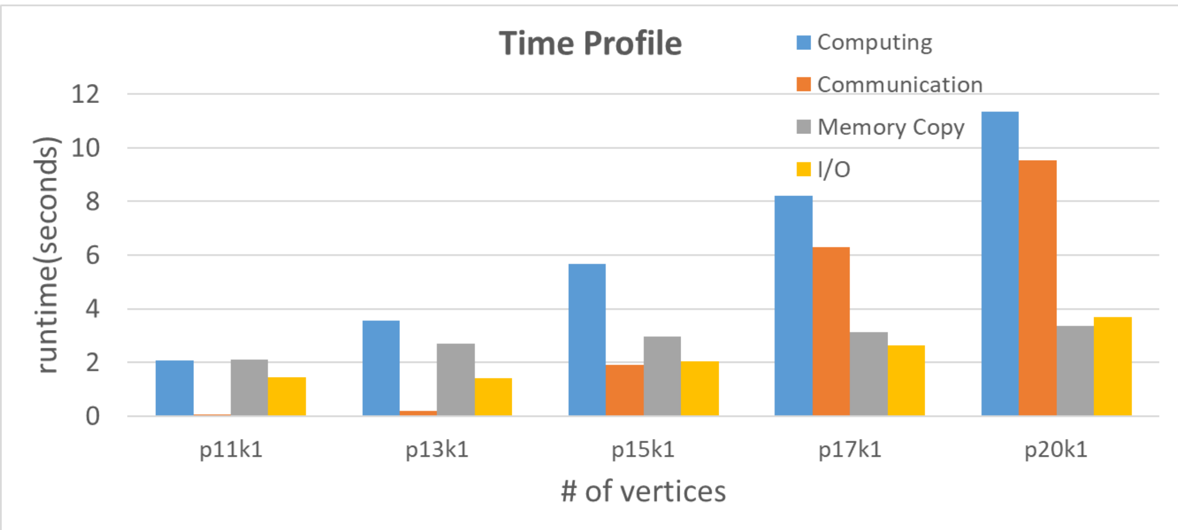
p17k1: vector ->17000, edge->4326829

p20k1: vector ->20000, edge->264275

(1) Weak Scalability



(2) Time Distribution



數值:

test case	Computing	Communication	Memory Copy	I/O
p11k1	2.0810008	0.059522	2.105695	1.4375
p13k1	3.552246	0.18143	2.681995	1.421875
p15k1	5.65249955	1.90014	2.95735	2.03125
p17k1	8.19671845	6.30011	3.13489	2.625
p20k1	11.34411735	9.52633	3.362755	3.703125

5. Conclusion

這次的作業重點為 GPU communication，一開始本來打算直接把整個 vertex matrix 互傳，但發現 communication overhead 會變得超級高，甚至比 single GPU 還差，所以只需要用的 data 變的十分重要，了解到這會直接嚴重影響 performance。