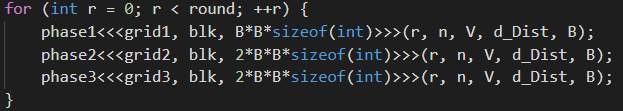
HW4 – Report

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

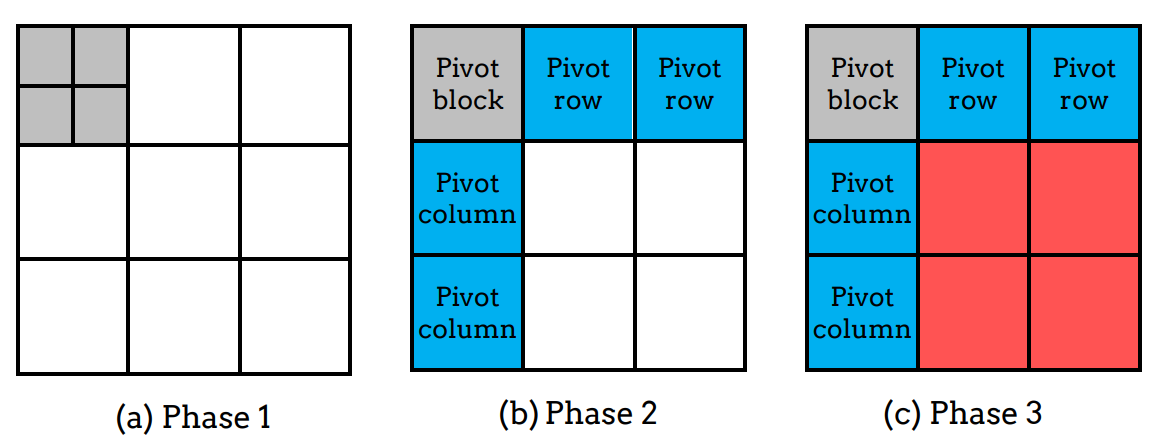
* **Divide Data:** 如同這次作業spec的切割方法，將data切成blocks，以下為configuration:
  1. **Blocking Factor:** 32
  2. **Blocks:** (data量/32)2
  3. **Threads:** 32\*32
* **Implementation:** 如同這次作業spec的實作方法，將過程分為好幾rounds，每round分為3個phases:



(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

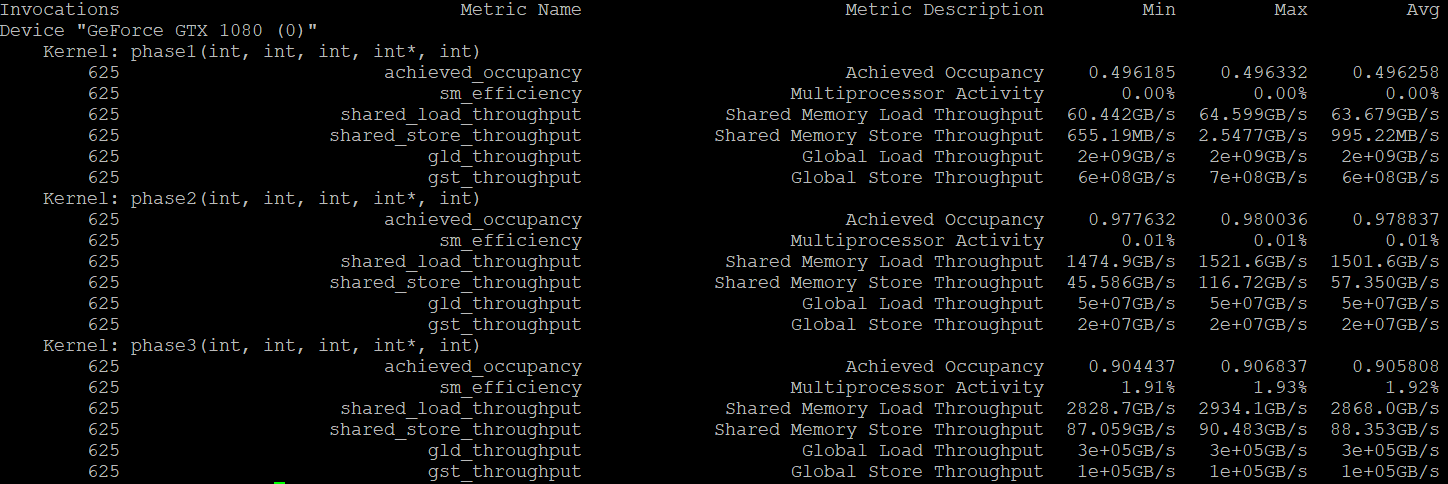


1. **Profiling Results**

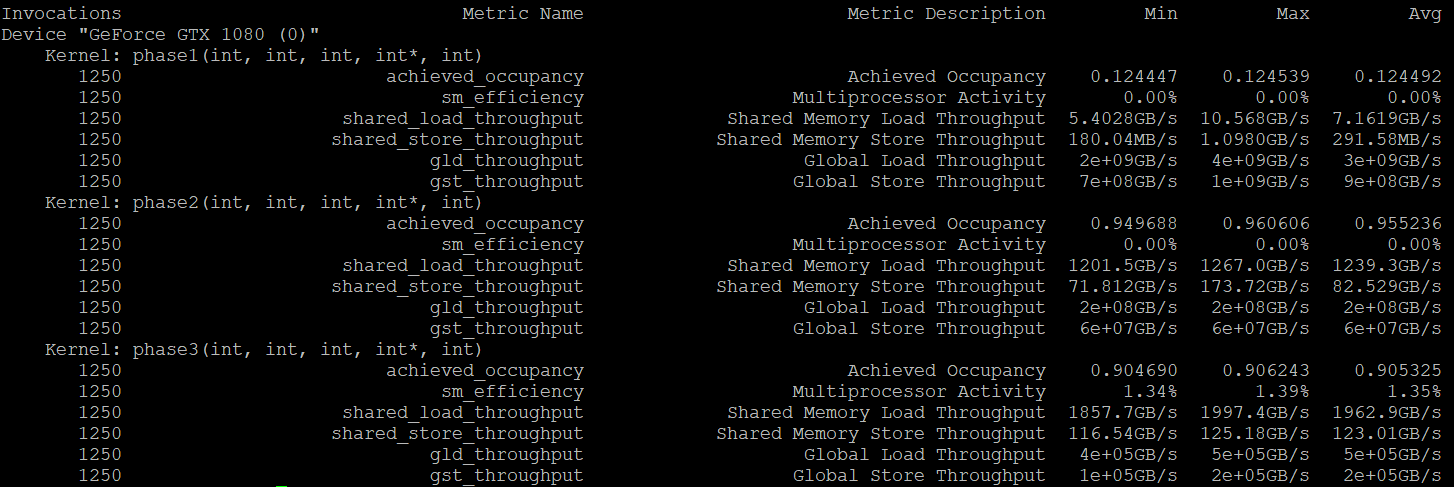
*使用p20k1做measurement*

以下為Occupancy, sm efficiency, shared memory load/store throughput, global load/store throughput測量之結果:

* **Block dimension: (32, 32)**



* **Block dimension: (16, 16)**



**結論:** 在occupancy中，block dimension 16x16明顯比block dimension 32x32來的小很多，代表同時active的warp數比較少。由於GeForce GTX 1080每個block一次最多只能launch 1024個threads，所以blocking factor最多只能設置32，此為bottleneck所在。

1. **Experiment & Analysis**

* **Time Distribution:**

*使用以下test cases做measurement:*

*p11k1: vector ->11000, edge->505586*

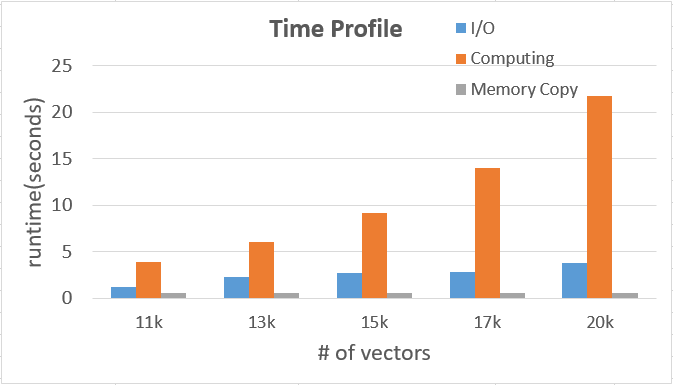
*p13k1: vector ->13000, edge->1829967*

*p15k1: vector ->15000, edge->5591272*

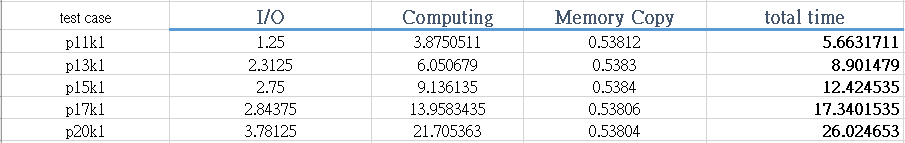
*p17k1: vector ->17000, edge->4326829*

*p20k1: vector ->20000, edge->264275*

1. Time Distribution

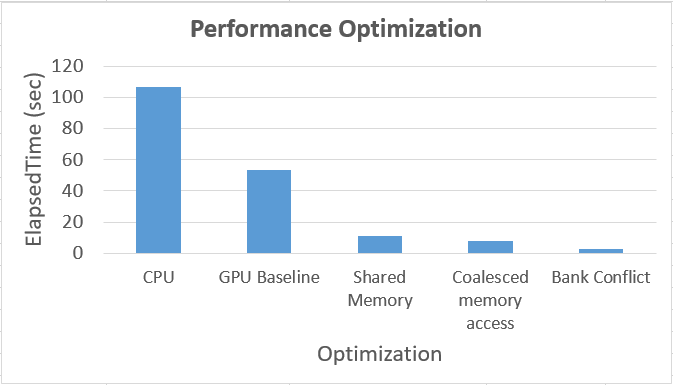


數值:



1. Optimization

*使用c21.1做measurement*



1. **Conclusion**

這次的作業我寫了好幾個版本，從一開始只用global memory做存取、copy至shared memory到最後解決bank conflict的問題，一步一步慢慢加速，也從中了解到記憶體存取方法及位置對於GPU計算的重要性。