HW4-2 milti-card

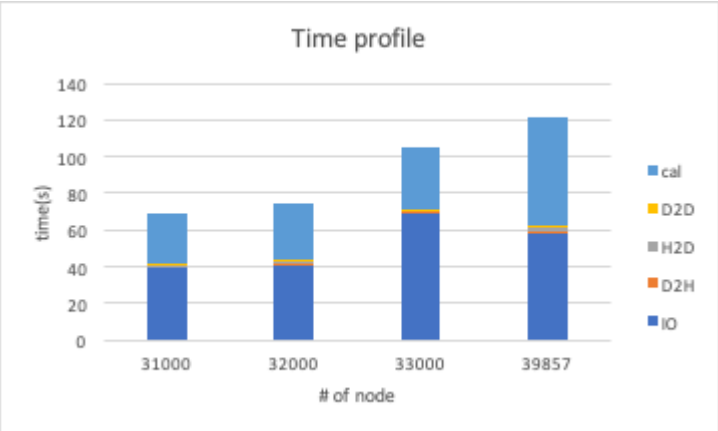
implementation

1. 一開始資料先透過Host傳給GPU0和GPU1
2. 在phase1和phase2當中，兩張GPU分別對他們自己的data做運算
3. phase3當中，他們分別計算一半的data
4. 在第r回合開始時，從負責第r回合的dev複製第r列給對方

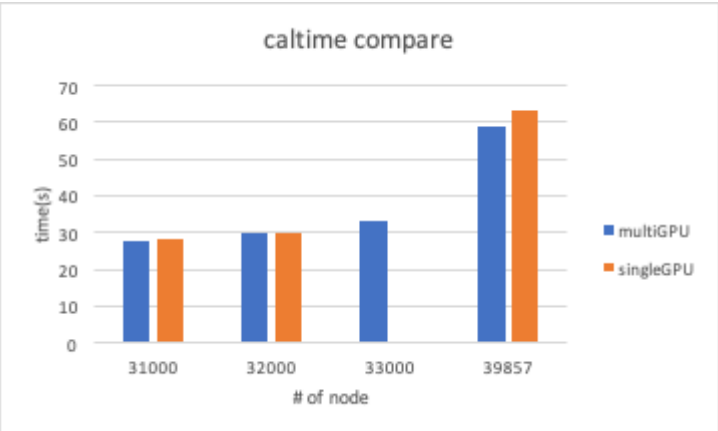
Experiment&Analysis

1. System Spec:hades

2. Weak Scalability:



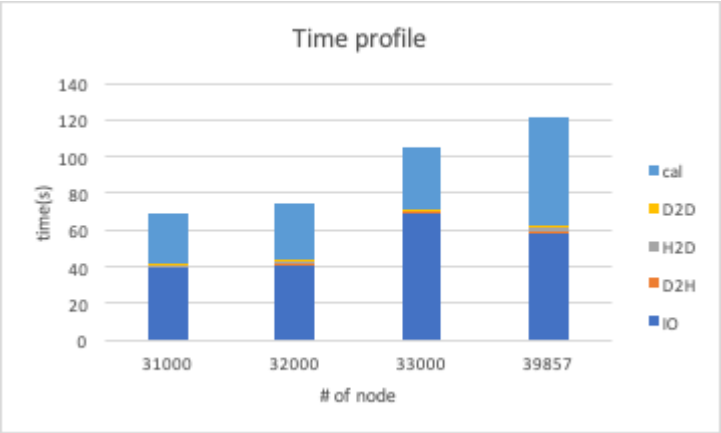
隨著資料量變大P2P,H2D和D2H都有些微的改變，這是因為兩張GPU之間需要溝通，但是全部的溝通也僅僅需要將半個matrix給彼此，因此幅度不會增加的太明顯



比較singe-GPU和multi-GPU得calculate time，會發現其實使用multi-GPU並不會加速太多，可能的原因是因為資料量不夠大，但是memory已經接近極限了，因此我認為multi-GPU的用途或許較適合用於partition，而非計算

3. Time Distribution:

n	e	IO	D2H	H2D	D2D	cal
31000	15125277	39.138507	0.58448	1.19287	0.85612	27.4063
32000	8281294	41.078747	0.62119	1.26783	0.91056	30.0828
33000	9027729	69.204016	0.66509	0.67629	0.97	33.3782
39857	4232291	58.44732	0.96963	1.97183	1.41344	58.9442



從圖表來看可以得知bottleneck在IO和calculate，相對下來D2H,H2D,P2P的時間都非常少

Experience & conclusion

1. 透過這次的實作經驗學會了如何寫multi-card得技巧
2. 使用omp一個thread控制一個GPU時，communication時需要block住，因為dev1不會知道dev0有傳送資料給他
3. 使用cudaMemcpy2D的速度會比cudaMemcpy還來得快，因為communication的時間會overlap