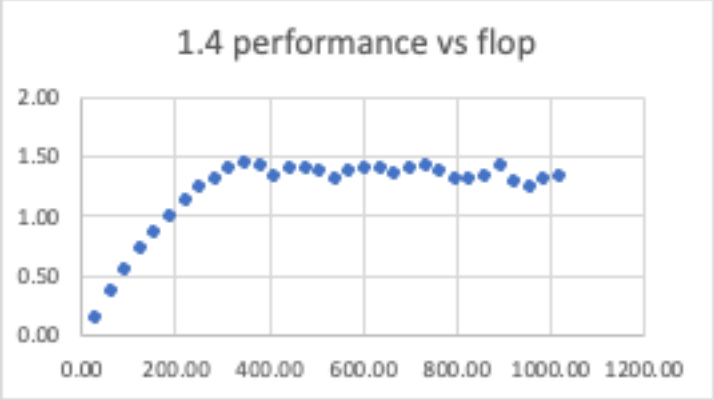
P1.1-1.3

```
[weihongh@icme-gpu:~/hw3/starter-code$ make run1
nvcc -O3 -std=c++11 -arch=compute_75 -code=sm_75 -o main_q1 main_q1.cu
main_q1.cu(216): warning: label "std" was declared but never referenced

srun --partition=CME --gres=gpu:1 ./main_q1
test
Largest error found at pos: 10 error 7.54233e-08 expected 1.58054 and got 1.58054
small test case pass
Largest error found at pos: 0 error 0 expected -0.9537 and got -0.9537
Largest error found at pos: 352873 error 1.19209e-07 expected 1 and got 1
Largest error found at pos: 866140 error 2.38414e-07 expected 2.00004 and got 2.00004
Largest error found at pos: 748818 error 5.46302e-07 expected 17.4569 and got 17.4569
Largest error found at pos: 386765 error 1.14615e-06 expected 319.515 and got 319.516
Largest error found at pos: 341942 error 2.33473e-06 expected 103732 and got 103732
Largest error found at pos: 341942 error 4.66302e-06 expected 1.07604e+10 and got 1.07604e+10
Largest error found at pos: 341942 error 9.3441e-06 expected 1.15786e+20 and got 1.15785e+20
Largest error found at pos: 209454 error 1.70229e-05 expected 4.92744e+21 and got 4.92736e+21
Largest error found at pos: 945175 error 2.59454e-05 expected 1.60256e+37 and got 1.6026e+37

Questions 1.1-1.3: your code passed all the tests!
```

P1.4

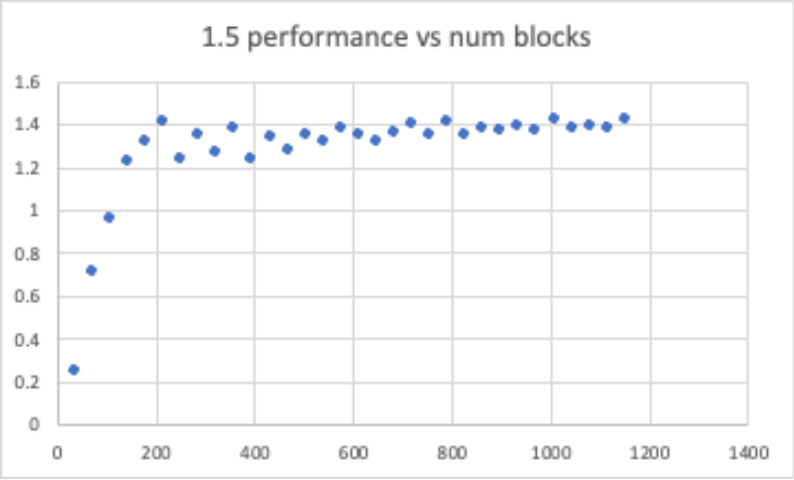
Q1.4	
Number of Threads	Performance TFlops/sec
32	0.146732
64	0.359105
96	0.531933
128	0.711131
160	0.852685
192	0.977394
224	1.11194
256	1.22225
288	1.30565
320	1.39815
352	1.43063
384	1.40817
416	1.31997
448	1.38848
480	1.39844
512	1.36517
544	1.2998
576	1.35874
608	1.39804
640	1.38564
672	1.33982
704	1.38499
736	1.4058
768	1.37063
800	1.30102
832	1.30531
864	1.32723
896	1.40964
928	1.28823
960	1.2243
992	1.29546
1024	1.33258



We can see the performance start to peak out at ~350 threads. We will fully utilize all resources then and additional threads will not improve performance.

Q1.5

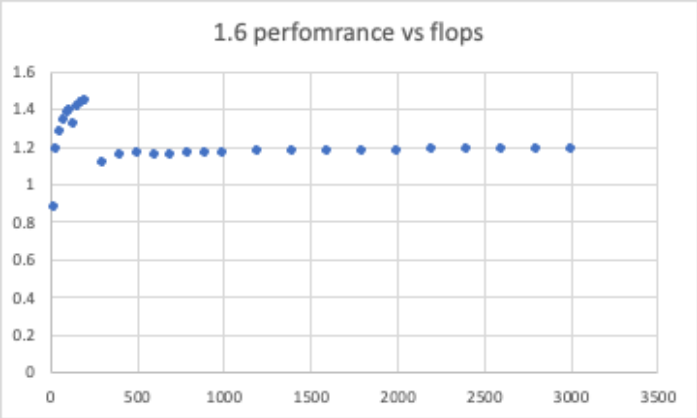
Number of Blocks	Performance TFlops/sec
36	0.24731
72	0.707427
108	0.961619
144	1.22622
180	1.31949
216	1.40577
252	1.23883
288	1.34489
324	1.26188
360	1.38267
396	1.23638
432	1.33458
468	1.27319
504	1.34866
540	1.32141
576	1.37922
612	1.34608
648	1.31378
684	1.35464
720	1.39769
756	1.34466
792	1.41098
828	1.34838
864	1.38149
900	1.3671
936	1.39442
972	1.37224
1008	1.41637
1044	1.38327
1080	1.39464
1116	1.37509
1152	1.42123



Using hint, we have 8 blocks/SM(8*128=1024). And 72 SMs, so we have 72*8=576 blocks. We can see performance peaking out around 550(less variance there compared to 200). This make sense as we have used all our resources. Doing more blocks will fight for resource with existing blocks

Q1.6

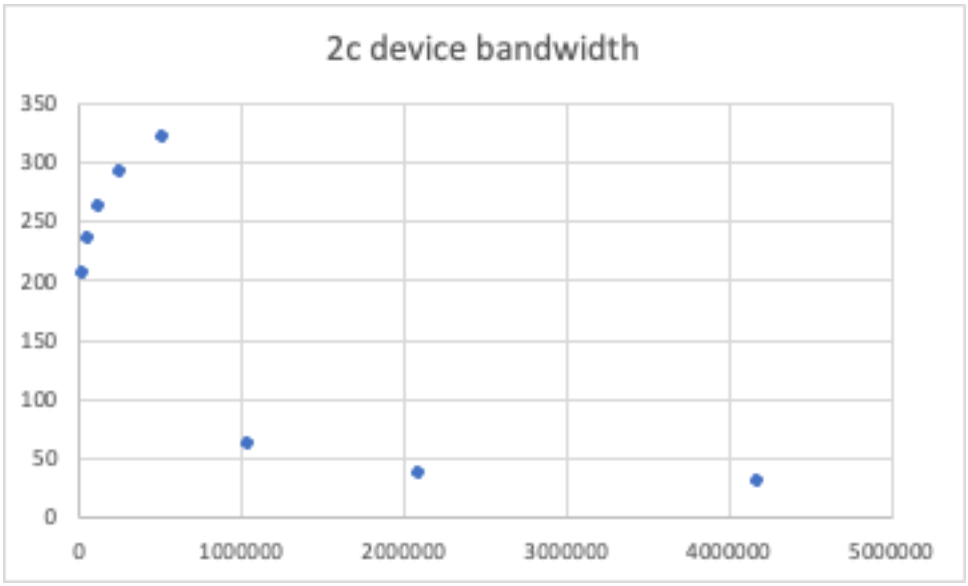
Number of Iters	Performance TFlops/sec
20	0.87535
40	1.18147
60	1.27378
80	1.33905
100	1.37726
120	1.39509
140	1.31362
160	1.41004
180	1.43002
200	1.44525
300	1.10999
400	1.15388
500	1.16331
600	1.15673
700	1.15725
800	1.15999
900	1.16411
1000	1.16398
1200	1.17187
1400	1.17668
1600	1.1765
1800	1.17721
2000	1.17846
2200	1.1814
2400	1.18192
2600	1.18213
2800	1.18374
3000	1.18195



On the performance vs iterations plot, we can see performance peaking out at around 250 iterations, and there's a break after that. I have no idea why, my best guess is resources are fully utilize up to 250, and break down after that.

Question 2

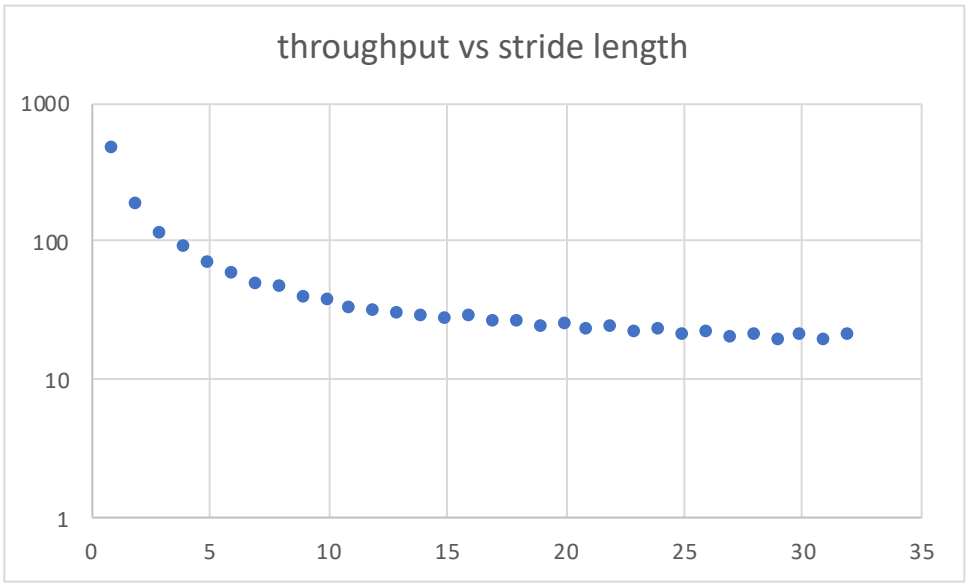
```
weihongh@icme-gpu:~/hw3/starter-code$ make run2
nvcc -O3 -std=c++11 -arch=compute_75 -code=sm_75 -o main_q2 main_q2.cu
srun --partition=CME --gres=gpu:1 ./main_q2
                                Device Bandwidth GB/sec

                                                Number of nodes
                      32768         65536        131072        262144        524288       1048576       2097152       4194304
 Avg. no. edges
                2        123.55        193.51        284.41        356.29        390.62        109.83         62.43         47.70
                3        122.27        203.11        275.47        334.12        377.08         95.15         52.42         40.37
                4        130.19        211.65        270.99        317.83        357.66         83.81         46.57         36.43
                5        134.88        221.06        276.95        316.68        348.50         80.44         43.76         35.09
                6        136.96        223.55        278.86        311.50        339.81         79.29         42.01         33.05
                7        152.90        226.99        264.75        309.17        334.94         72.93         41.85         31.83
                8        140.77        231.06        279.41        299.39        327.23         68.23         39.88         31.28
                9        193.23        290.14        342.49        372.25        317.98         63.55         37.61         30.27
               10        206.13        233.96        262.08        291.15        319.70         61.54         36.23         29.55
               11        174.79        241.22        264.35        297.67        316.38         57.85         35.25         28.93
               12        173.40        243.30        262.71        294.46        311.18         55.68         34.86         28.13
               13        176.40        242.32        258.39        289.56        310.15         53.94         35.07          0.85
               14        175.94        242.79        258.20        290.53        303.33         52.14         33.95          2.54
               15        182.48        240.31        261.28        288.20        296.98         50.96         33.05          3.95
               16        182.97        243.17        257.67        284.75        292.39         49.86         32.42          5.28
               17        186.86        245.98        255.63        282.51        280.24         50.52         32.20          6.35
               18        178.07        245.07        254.61        280.10        275.32         49.91         32.89          7.26
               19        190.23        243.50        254.89        281.28        265.81         48.54         32.79          8.25
```



2.4. The plot is increasing initially, but decreasing after that. It's increasing in the first part as there were idle resources not being fully make use of. However, as number of nodes increases too much. We are not able to fully store the constants properly in a SM. This result in more reading from further caches/memory.

3

```
[weihongh@icme-gpu:~/hw3/starter-code$ make run3
nvcc -O3 -std=c++11 -arch=compute_75 -code=sm_75 -o main_q3 main_q3.cu
srun --partition=CME --gres=gpu:1 ./main_q3
# Using device: Quadro RTX 6000
# stride     time [ms]   GB/sec
       1      1.3259     452.5
       2      3.3447     179.4
       3      5.3466     112.2
       4      6.6858      89.7
       5      8.8305      67.9
       6     10.2199      58.7
       7     12.3072      48.8
       8     13.0717      45.9
       9     15.6080      38.4
      10     16.4051      36.6
      11     18.2352      32.9
      12     19.1120      31.4
      13     20.6221      29.1
      14     20.8457      28.8
      15     22.1312      27.1
      16     21.0159      28.5
      17     23.6209      25.4
      18     23.5197      25.5
      19     25.2074      23.8
      20     24.5364      24.5
      21     26.1823      22.9
      22     25.7119      23.3
      23     28.1633      21.3
      24     26.4404      22.7
      25     29.2731      20.5
      26     27.8999      21.5
      27     30.2183      19.9
      28     29.0200      20.7
      29     31.2025      19.2
      30     29.3424      20.4
      31     31.4989      19.0
      32     28.4896      21.1
```



First of all, data is stored contiguously. And we might not be able to store all in a single SM. We have to go there further places to fetch data. When stride length is small, most of the data can be access “nearby”, nearest cache or shared memory. We can see as stride length increases, performance decreases.