CME 213 - Parallel Computing
homework 3

Leon H. Kloker
Institute for Computational and Mathematical Engineering
CME 213, Stanford University
leonkl@stanford.edu

Problem 1

Output from main_q1:

```
$ make run1
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -c main_q1.cu
nvcc -isystem ./googletest-main/googletest/include -I./googletest-main/googletest
-O3 -std=c++14 -arch=compute_75 -code=sm_75 -c
./googletest-main/googletest/src/gtest-all.cc
nvcc -isystem ./googletest-main/googletest/include -I./googletest-main/googletest
-O3 -std=c++14 -arch=compute_75 -code=sm_75 -c
./googletest-main/googletest/src/gtest_main.cc
ar rv gtest_main.a gtest-all.o gtest_main.o
r - gtest-all.o
r - gtest_main.o
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -lpthread main_q1.o gtest_main.a -o main_q1
srun -partition=CME -qos=cme -gres=gpu:1 ./main_q1
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 6 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 6 tests from RecurrenceTestFixture
[ RUN ] RecurrenceTestFixture.GPUAllocationTest_1
[ OK ] RecurrenceTestFixture.GPUAllocationTest_1 (2411 ms)
[ RUN ] RecurrenceTestFixture.InitalizeArrayTest_2
[ OK ] RecurrenceTestFixture.InitalizeArrayTest_2 (18540 ms)
[ RUN ] RecurrenceTestFixture.RecurrenceKernelTest_3
Largest error found at pos:  10 error 5.96046e-08 expected 0.90869 and got
0.90869
Largest error found at pos:  681885 error 1.19199e-07 expected 1.00008 and
got 1.00008
Largest error found at pos:  941076 error 2.38392e-07 expected 2.00023 and
got 2.00023
Largest error found at pos:  637665 error 5.56931e-07 expected 17.1237 and
got 17.1237
Largest error found at pos:  881942 error 1.13313e-06 expected 323.185 and
got 323.185
Largest error found at pos:  44313 error 2.25644e-06 expected 90020.1 and
got 90020.3
Largest error found at pos:  44313 error 4.4859e-06 expected 8.10362e+09
```

```
and got 8.10366e+09
Largest error found at pos:  44313 error 8.97442e-06 expected 6.56687e+19
and got 6.56692e+19
Largest error found at pos:  116328 error 1.59202e-05 expected 2.9986e+22
and got 2.99865e+22
Largest error found at pos:  979398 error 2.84008e-05 expected 5.82338e+34
and got 5.82321e+34

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

[ OK ] RecurrenceTestFixture.RecurrenceKernelTest_3 (19123 ms)
[ RUN ] RecurrenceTestFixture.RecurrenceThreadsTest_4
Q1.4
---------------------
Number of Threads Performance TFlops/sec
32 1.36972
64 2.95946
96 4.4248
128 5.88215
160 6.73143
192 8.17747
224 9.63079
256 10.8182
288 9.1562
320 10.1885
352 11.1924
384 12.09
416 9.83818
448 10.7701
480 11.5155
512 11.9238
544 10.3093
576 10.7227
608 11.6768
640 12.294
672 10.7724
704 11.3089
736 11.8948
768 11.8954
800 10.7738
832 11.4095
864 11.4613
896 12.1027
928 11.2977
960 11.3125
992 13.9794
1024 14.3187

[ OK ] RecurrenceTestFixture.RecurrenceThreadsTest_4 (90125 ms)
[ RUN ] RecurrenceTestFixture.RecurrenceBlocksTest_5
Q1.5
---------------------
Number of Blocks Performance TFlops/sec
36 2.5764
72 5.81977
108 8.00082
144 10.5774
```

```
180 7.91109
216 9.41747
252 9.50881
288 10.7374
324 9.39737
360 10.5574
396 10.2037
432 10.9089
468 10.381
504 11.3658
540 10.8814
576 11.4564
612 10.1016
648 10.764
684 11.2881
720 11.734
756 11.3642
792 11.951
828 11.4508
864 11.6861
900 11.5139
936 11.7303
972 11.683
1008 11.9781
1044 11.6651
1080 11.89
1116 11.747
1152 12.0158

[ OK ] RecurrenceTestFixture.RecurrenceBlocksTest_5 (89893 ms)
[ RUN ] RecurrenceTestFixture.RecurrenceItersTest_6
Q1.6
---------------------
Number of Iters Performance TFlops/sec
20 0.907771
40 3.34225
60 4.6875
80 6.33714
100 6.97545
120 7.32422
140 8.54492
160 8.82613
180 9.30521
200 9.52744
300 8.37054
400 9.26612
500 8.91329
600 10.0563
700 10.2029
800 10.5064
900 10.518
1000 10.4989
1200 10.8507
1400 10.8547
1600 10.7863
1800 10.9863
2000 10.9726
2200 11.1318
```

```
2400 11.1607
2600 11.2847
2800 11.3151
3000 11.2734

[ OK ] RecurrenceTestFixture.RecurrenceItersTest_6 (66612 ms)
[-----] 6 tests from RecurrenceTestFixture (286707 ms total)

[-----] Global test environment tear-down
[==========] 6 tests from 1 test suite ran.  (286707 ms total)
[ PASSED ] 6 tests.
```

Question 1.4

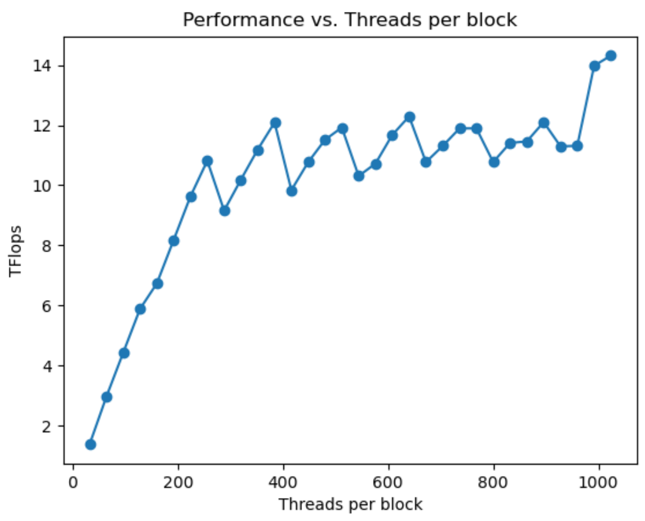


Figure 1: As the GPU has 72 streaming multiprocessors, each of the 72 blocks is assigned to a single SM. Moreover, we increase the amount of threads per block in steps of 32, meaning we add one warp to every block at every step. In the plot, we can see that the performance increases until 8 warps per SM are reached. From then on, the performance oscillates (while the peaks are still slightly increasing) with a wavelength (loosely speaking) of 4 warps. As a single SM on the ICME GPU can only run 4 warps in parallel, adding another warp means a warp is idle which increases the runtime, thus the performance is optimal when number of warps per block is a multiple of 4.

Question 1.5

Question 1.6

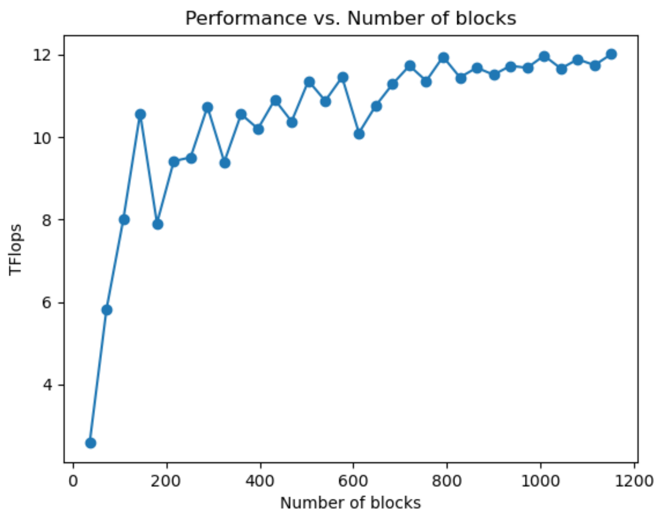


Figure 2: The performance increases initially and then starts oscillating with a wavelength of 72 blocks while performance still increases slightly. The oscillation is due to the fact that we have 72 SMs, meaning when the number of blocks is a multiple of 72, the same amount of blocks is mapped to each SM leading to a minimization of the time where one SM is idle. Moreover, when we reach 8*72 blocks, each SM reaches its maximal capacity of 8 blocks and the performance basically peaks as the entire GPU is maxed out.

Problem 2

Output from main_q2:

```
$ make run2
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -c main_q2.cu
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -lpthread main_q2.o gtest_main.a -o main_q2
srun -partition=CME -qos=cme -gres=gpu:1 ./main_q2
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from testQ2
[ RUN ] testQ2.testPageRank
Device Bandwidth GB/sec


 Number of nodes
32768 65536 131072 262144 524288 1048576 2097152 4194304
Avg. no. edges
2 136.05 239.93 333.48 415.85 473.71 118.02 72.93 57.04
3 142.93 267.64 355.40 441.12 491.06 105.85 66.70 53.23
4 177.74 271.79 368.29 435.38 478.48 96.75 62.44 50.84
5 194.18 314.07 373.42 428.19 411.60 89.95 59.95 50.20
6 205.48 321.86 373.97 435.30 320.21 86.52 57.89 48.33
7 254.43 380.49 445.59 517.92 257.43 80.10 57.52 1.43
8 272.22 412.21 477.90 542.82 206.03 75.09 55.33 6.84
9 283.71 420.68 487.39 535.51 171.88 71.41 53.18 10.86
10 239.35 343.53 386.46 398.66 152.66 68.83 51.85 14.01
11 248.21 347.08 382.97 346.12 139.16 66.69 50.93 16.53
12 246.26 348.06 383.55 299.38 128.79 65.03 50.39 18.47
```

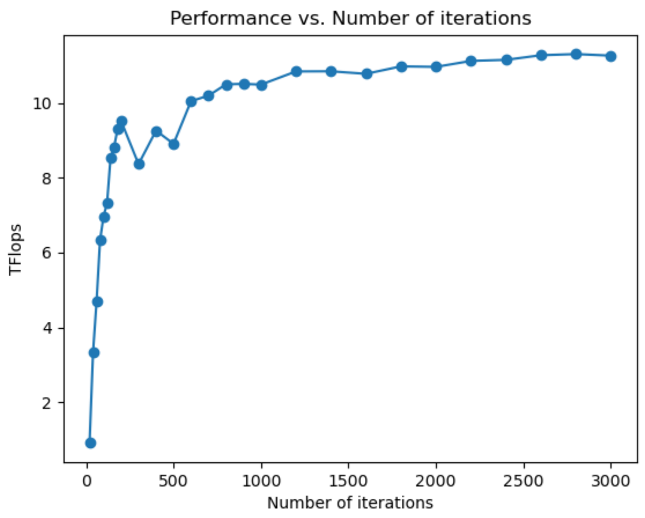
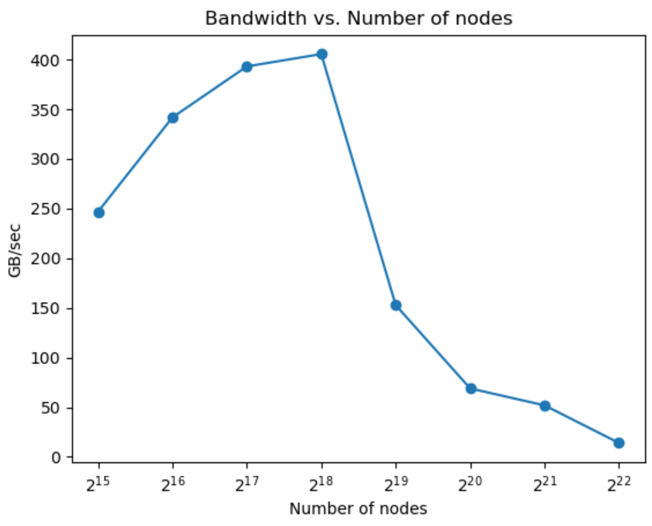


Figure 3: It is visible, that the performance increases sharply until around 200 iterations are executed. From then on, the performance plateaus and converges against around 11-12 TFlops. This is due to the fact that increasing the amount of iterations also increases the ratio of pure computational work of each thread to read and write operations, such as reading the constant c from the input array and writing the result of the iteration to the output array.

```
13 255.80 352.06 382.04 267.97 119.91 63.51 50.40 20.28
14 262.10 351.11 375.86 243.05 113.74 62.27 0.38 0.33
15 268.53 358.06 347.80 220.23 109.86 61.14 3.51 3.07
16 271.62 356.78 319.60 205.27 105.56 60.21 6.18 5.49
17 275.75 356.27 291.30 194.82 101.64 60.00 8.51 7.53
18 262.59 356.88 269.53 186.36 98.32 59.50 10.70 9.29
19 275.92 354.87 252.72 180.06 92.23 58.41 12.53 11.00
[ OK ] testQ2.testPageRank (103566 ms)
[-----] 1 test from testQ2 (103566 ms total)

[-----] Global test environment tear-down
[==========] 1 test from 1 test suite ran.  (103566 ms total)
[ PASSED ] 1 test.
```

Question 2.3



Question 2.4

Even though there is no stride when each thread reads from graph_indices or writes into graph_nodes_out, reading from graph_nodes_in and inv_edges_per_node when performing a pagerank update is done according to the adjacency structure of the graph. Hence, generally speaking, each threads accesses on average μ random entries which might need to be loaded from memory instead of the L2 cache, as they are not stored next to each other. In the plot, this is visible at 2^{18} nodes. The 5 arrays that are needed for pagerank have a size of 2^{20} bytes, meaning about 5MB of storage overall, which can fit into the L2 cache of the ICME cluster GPU. When doubling the number of nodes another time, not all the required arrays fit into the L2 cache, which means that, since the addresses each thread accesses are random, new data has to be loaded constantly from L3 to L2 cache. This increases the runtime significantly. For an even larger number of nodes, the frequency of capacity misses in L2 increases further, leading to further decreases in runtime and thus bandwidth.

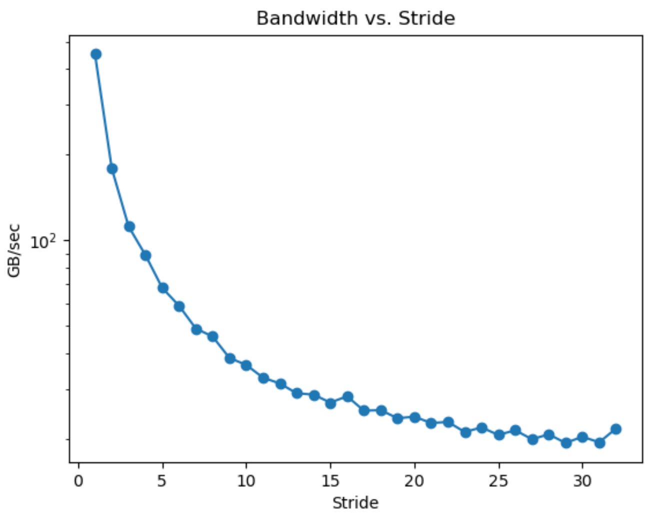
Problem 3

Output from main_q3:

```
$ make run3
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -c main_q3.cu
nvcc -isystem ./googletest-main/googletest/include -O3 -std=c++14
-arch=compute_75 -code=sm_75 -lpthread main_q3.o gtest_main.a -o main_q3
srun -partition=CME -qos=cme -gres=gpu:1 ./main_q3
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from testQ3
[ RUN ] testQ3.StrideTest
# Using device:  Quadro RTX 6000
# stride time [ms] GB/sec
1 0.0180 33363.0
2 0.0201 29856.7
3 0.0208 28890.6
4 0.0228 26334.3
5 0.0260 23034.4
6 0.0321 18712.6
7 0.0384 15638.0
8 0.0834 7194.9
9 0.1021 5877.7
10 0.1075 5583.7
11 0.1220 4920.0
12 0.1338 4484.6
13 0.1420 4224.9
14 0.1495 4013.3
15 0.1579 3800.9
16 0.1624 3695.3
17 0.1740 3449.2
18 0.1760 3409.7
19 0.1852 3240.0
20 0.1879 3193.1
21 0.1958 3064.7
22 0.1995 3007.7
23 0.2121 2829.3
24 0.2094 2865.7
25 0.2205 2720.5
26 0.2144 2798.9
27 0.2273 2639.4
28 0.2271 2642.0
29 0.2377 2524.6
30 0.2394 2506.7
31 0.2432 2467.4
32 0.2308 2599.8
[ OK ] testQ3.StrideTest (398 ms)
[-----] 1 test from testQ3 (398 ms total)


[-----] Global test environment tear-down
[==========] 1 test from 1 test suite ran.  (398 ms total)
[ PASSED ] 1 test.
```

Question 3.1



Question 3.2

The bandwidth is hyperbolic in relation to the stride as the array is too big to completely fit into the cache. For example, when the stride is 2, about half of the threads can't find the array entries in the cache and have to fetch them from memory. Thus, the bandwidth reduction is roughly proportional to the stride. Moreover, at stride equal to 1, we basically reach the maximal bandwidth of 480GB/sec for the ICME GPU.