CME213 Spring 2023 Homework 3

Ines Dormoy – idormoy@stanford.edu

May 11, 2023

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Problem 1

Question 1.3

```
Output from 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 (2427 ms)
[ RUN      ] RecurrenceTestFixture.InitalizeArrayTest_2
[       OK ] RecurrenceTestFixture.InitalizeArrayTest_2 (18525 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: 86975 error 1.19191e-07 expected 1.00015 and
    got 1.00015
Largest error found at pos: 941076 error 2.38392e-07 expected 2.00023 and
    got 2.00023
Largest error found at pos: 603501 error 5.61004e-07 expected 16.9994 and
    got 16.9994
Largest error found at pos: 603501 error 1.15797e-06 expected 289.897 and
    got 289.898
Largest error found at pos: 603501 error 2.324e-06 expected 84041.4 and
    got 84041.6
Largest error found at pos: 603501 error 4.63941e-06 expected 7.06296e+09
    and got 7.063e+09
Largest error found at pos: 603501 error 9.25711e-06 expected 4.98854e+19
    and got 4.98859e+19
```

```
Largest error found at pos: 581442 error 1.67619e-05 expected 2.14944e+22
    and got 2.14948e+22
Largest error found at pos: 503149 error 2.68556e-05 expected 1.29807e+35
    and got 1.29803e+35

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

Question 1.4

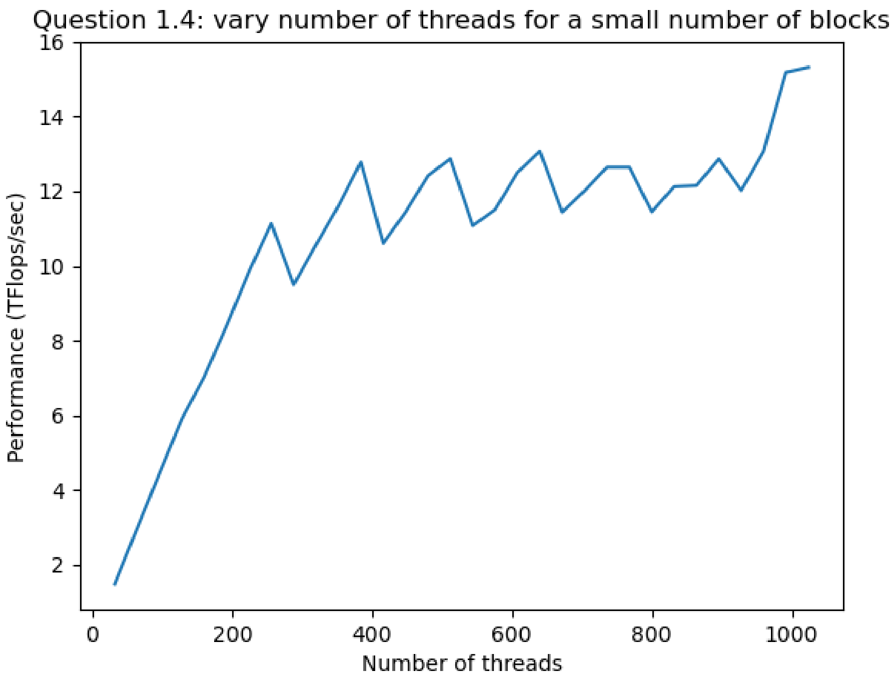

Figure 1: Question 1.4

We observe that the performance increases linearly until reaching 200 threads. After 200 threads, we exceed the total number of threads that can be handled by the GPU, and the performance stabilizes to $\sim$ 12 TFlops/sec with regular shrinks.
The period of the shrinks is 128. This is because an SM runs 4 warps at the same time, which makes it 128 threads (32 threads per warp). Whenever we add a thread after a 128 period, we have to run a whole new SM only for one thread, which makes the performance drop.

Question 1.5

.

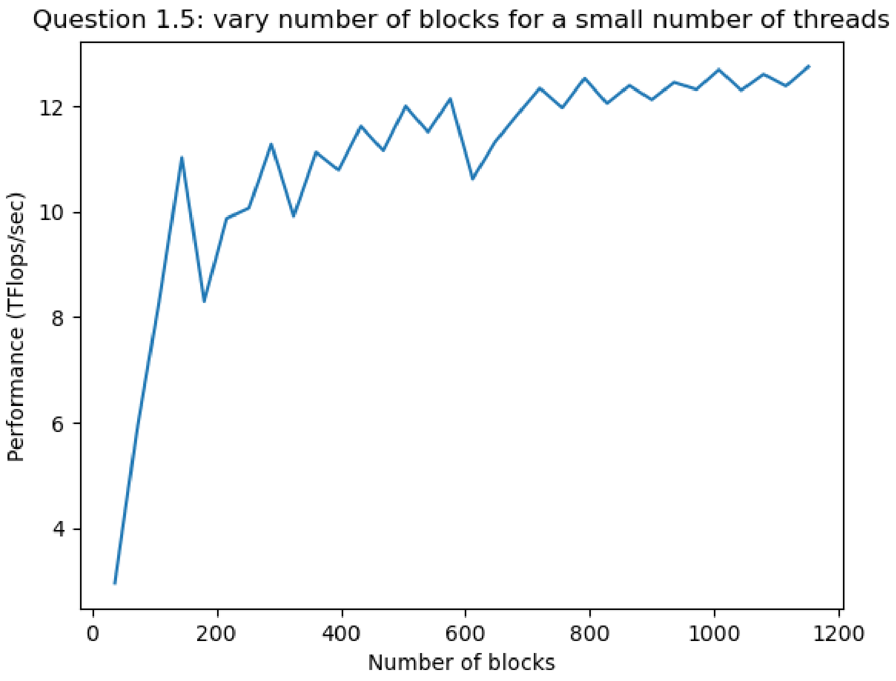

Figure 2: Question 1.5

We observe a performance increase as the number of blocks increases (the number of threads per block is fixed). The performance stagnates after 576 blocks (576 = 72 * 8, the maximum number of blocks the GPU can handle at once). We also observe regular shrinks as we increase the number of blocks. This is because we increase the number of blocks by 36 every time. When the number of blocks is divisible by 72, we get high performance. However, when this number is not divisible by 72, the workload is unevenly balanced between the SMs, which hinders performance.

Question 1.6

We observe that the performance increases until ~ 250 iterations, then stays steady. The asymptotic behavior is when we reach the maximum capacity of the GPU (12 TFlops/sec in this particular setting). We do not achieve the maximum compute capability before because there is a cost in allocating the blocks and threads which is not worth it if we do not execute a minimum of iterations. After approximately 1000 iterations, we reach maximum GPU capacity and are using all the threads at their maximum.

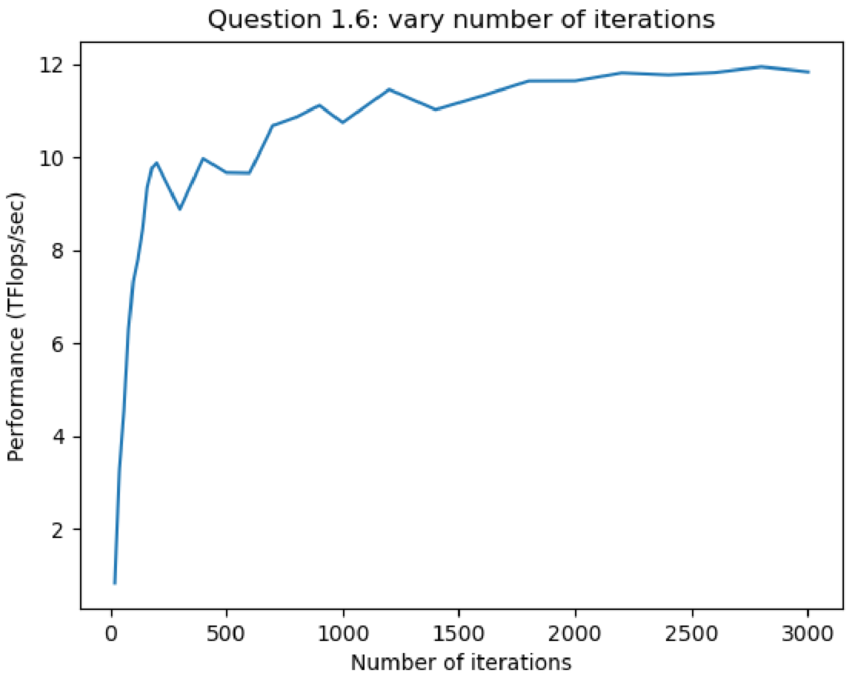

Figure 3: Question 1.6

Problem 2

Question 2.3

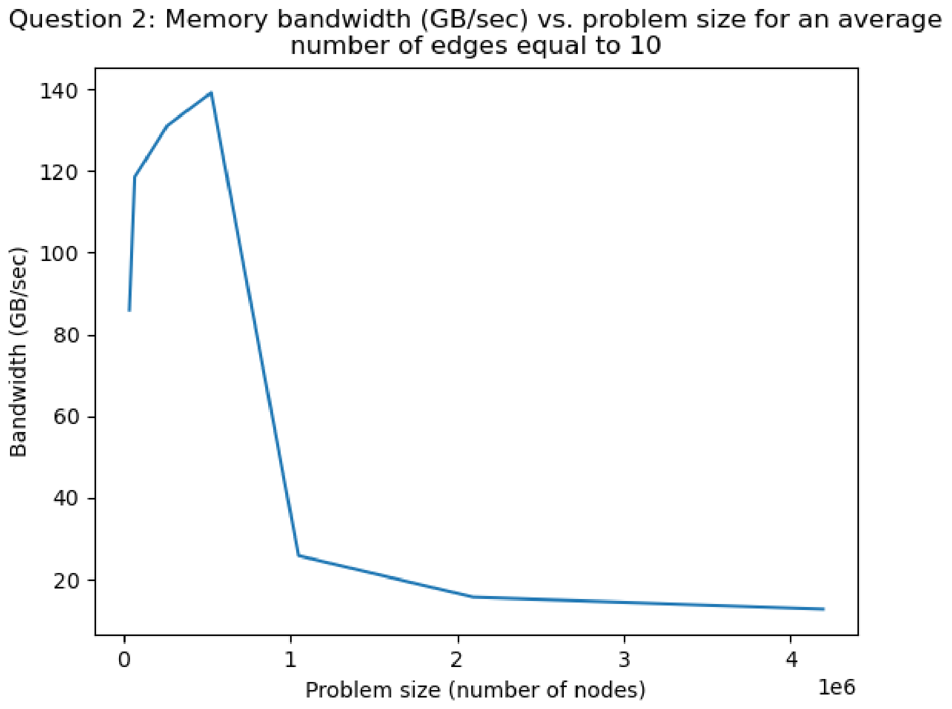

Figure 4: Question 2.3

Question 2.4

We observe that the memory bandwidth is at its maximum for approximately 500,000, and then drops. This is because when the problem size is too big, L2 memory is full, and bandwidth drops as L3/main memory accesses are needed. Those L3/main memory accesses are way slower than the L2 memory accesses.
We do not reach the maximum bandwidth of about 480 GB/sec because we are making irregular memory accesses because A is sparse. Therefore, the memory access is not contiguous and is slower, which prevents from reaching the theoretical bandwidth maximum.

Problem 3

Question 3.1

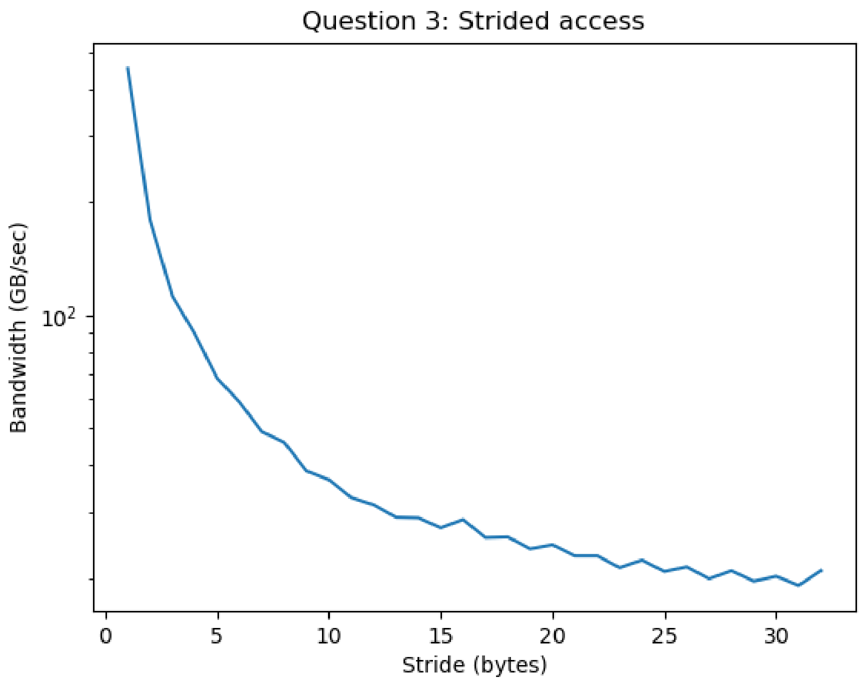

Figure 5: Question 3.1

Question 3.2

We observe that the bandwidth diminishes as the stride increases. This is because when the stride is 0, there are no cache misses when accessing the memory, and the maximum bandwidth is used. When we increase the stride, some cache misses start appearing, thus diminishing the bandwidth. Increasing the stride increases the number of cache misses and diminishes performance.