Sergi Aliaga

Northeastern University

Homework 5

Question 2

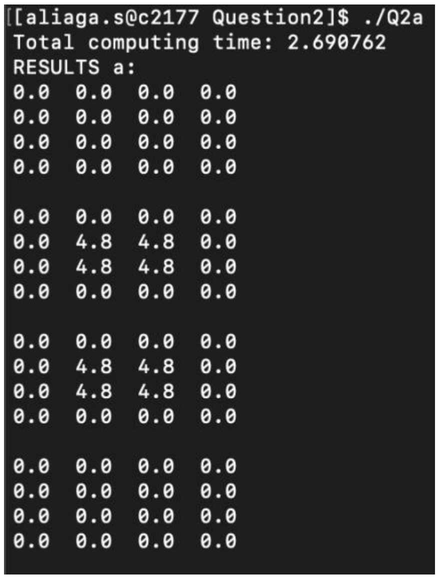
The code below carries out a "nearest neighbor" or "stencil" computation. This class of algorithm appears frequently in image processing applications. The memory reference pattern for matrix b exhibits reuse in 3 dimensions. Your task is to develop a C/CUDA version of this code that initializes b on the host and then uses tiling on theGPU to exploit locality in shared memory across the 3 dimensions of b:

```
#define n 64
float a[n][n][n], b[n][n][n];
for (i=1; i<n-1; i++)
   for (j=1; j<n-1; j++)
      for (k=1; k<n-1; k++) {
      a[i][j][k]=0.8*(b[i-1][j][k]+b[i+1][j][k]+b[i][j-1][k]
      + b[i][j+1][k]+b[i][j][k-1]+b[i][j][k+1]);
     }
```

a. *Evaluate the performance of computing a tiled versus non-tiled implementation in your GPU application. Explore what happens when you change the value of n. Consider the performance for at least two additional values for n.*

b. *Explore and report on other optimizations to accelerate your code on the GPU*

The code used for this part of the assignment is included in the file "*Q2a.c*". Again, the version of CUDA used is 10.2 and the version for gcc is 6.4.0. The approach followed to adapt the given code to a GPU implementation of the 6-element 3D stencil computation has been to assign one read of "b" and the corresponding constant multiplication and addition in "a" to each thread. Therefore, the total number of threads corresponds to the total number of non-zero elements of "a" ($(n-2)^3$) and the number of reads of "b" per element of "a" (6), resulting into $N_{threads} = (n-2)^3 * 6$. To validate the output of the code, we initialized all the elements of "b" to 1, computed the stencil computation for small values of n (4), and printed the output. For the code provided above, the corresponding output should be all non-boundary elements of "a" equal to 4.8 (6*0.8) and 0 for the boundary values. Figure 1 reports the code output for the case of n=4, where each 4 by 4 matrix corresponds to each value of the k-index. From there we confirm the proper behavior of the code.



```
[[aliaga.s@c2177 Question2]$ ./Q2a
Total computing time: 2.690762
RESULTS a:
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0

0.0  0.0  0.0  0.0
0.0  4.8  4.8  0.0
0.0  4.8  4.8  0.0
0.0  0.0  0.0  0.0

0.0  0.0  0.0  0.0
0.0  4.8  4.8  0.0
0.0  4.8  4.8  0.0
0.0  0.0  0.0  0.0

0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
```

Figure 1: Stencil computation for n=4

The computing time results for different values of "n" are reported in Figure 2. Here we can observe the outstanding scalability properties of GPU, with almost no increase in compute time for n=4 to n=128. Beyond those values, the number of threads is higher than the available in the GPU and some operations are serialized, resulting into an increase in computing time.

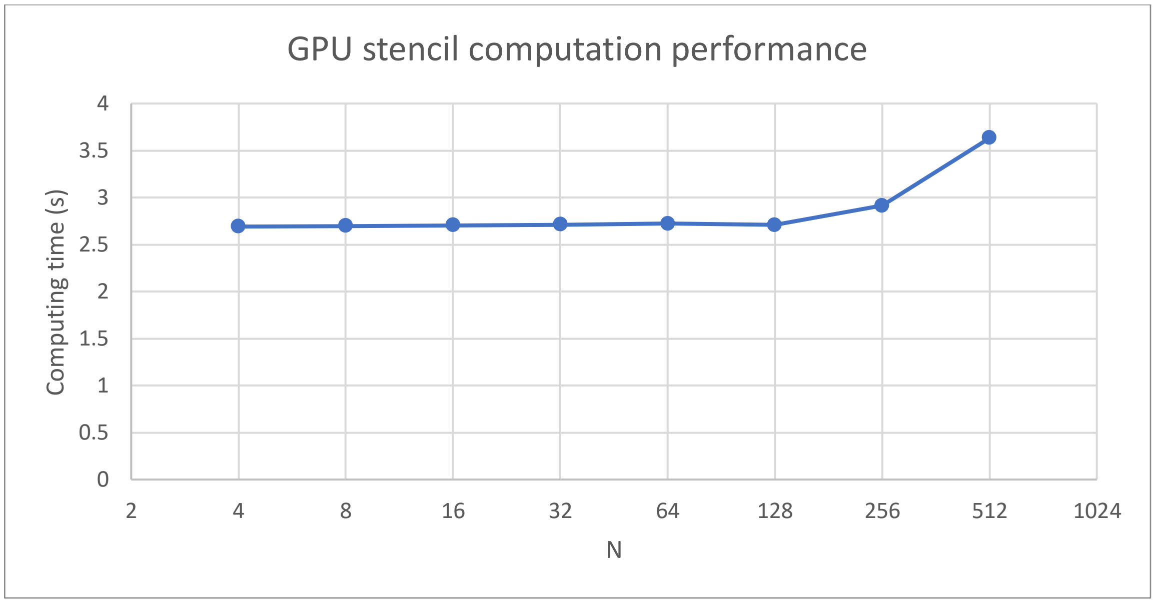


Figure 2: Computing time performance for the proposed GPU implementation of the 6-element 3D stencil computaion