CS516: Parallelization of Programs

Assignment-2

Due date: Feb 15, 2024 11:59 pm

Instructions:

1. Prepare your submission in a zip file and name it as <ROLL NO>.zip.
2. Your submission should include a README file containing the instructions to execute your programs. README carries 5 Marks.
3. Programs that do not compile/run will not get any points.

Question-1: **[35 Points]**

Consider a Matrix M of size m*m having integer elements. Write a program under the following versions the following scenarios that takes input M and gives output matrix N (size m*m) by applying the below stencil operation.

N[i][j] = M[i-1][j] + M[i+1][j] + M[i][j-1] + M[i][j-1]

Assume that M[i][j] = 0 if i<0 or i>=m or j<0 or j>=m

Version-1: Implement a sequential version of the program using C/C++ language.
Version-2: Implement a parallel version of the program using CUDA. Use global memory and launch only one thread block to compute, where each thread is responsible for computing one element of the N[i][j].
Version-3: Implement a parallel version of the program using CUDA. Use global memory memory, launch as many threads and thread blocks as needed, and use the concept of tiling discussed in class.
Version-4: Repeat Version-3 using shared memory for storing the input matrix.

Submission Instructions:

1. Prepare your solutions in a folder Q1.
2. The folder file should contains 4 files:
    a. File named as "***seq.c***", which is the implementation for Version-1
    b. File named as "***parallel_1tb.cu***", which is the implementation for Version-2
    c. File named as "***parallel_multiple_tb.cu***", which is the implementation for Version-3
    d. File named as "***parallel_shared_memory.cu***", which is the implementation for Version-4
    e. A ***README*** file containing the instructions to execute both programs.
    f. A report named "***report.pdf***", which is described below.
3. Your report should contain the following
    a. Report the GPU hardware parameters that you are using. You can use NVIDIA Documentation to obtain them.
        i. GPU device name (such as NVIDIA Tesla T4, NVIDIA K80, etc).

ii. Number of streaming multiprocessors present in the GPU
iii. Shared memory available in each SM
iv. Maximum size of the thread block

b. Report the maximum value for m that your Version-2 supports
c. Prepare a table as shown in Table 1 that compares the execution time of your Version-1, Version-3 and Version-4 implementations. To measure the execution time, you can use the *gettimeofday* function; an example of usage is listed here.

Matrix Size (m*m)	**Thread Block Size**	**Number of Thread Blocks**	**Version-1 Execution Time**	**Version-3 Execution Time**	**Version-4 Execution Time**
128*128	1024	16	To be filled	To be filled	To be filled
256*256	1024	64	To be filled	To be filled	To be filled
512*512	1024	256	To be filled	To be filled	To be filled
1024*1024	1024	1024	To be filled	To be filled	To be filled
2048*2048	1024	4096	To be filled	To be filled	To be filled
4096*4096	1024	16384	To be filled	To be filled	To be filled
8192*8192	1024	65536	To be filled	To be filled	To be filled

Question-2: [65 Points]

Consider an unsorted array A of integers having size N, and another sorted array of integers B of size M. Array B defines the partitions of A by specifying index values of A ranging from 0 to N-1. Each index element (i) in the array B denotes the new partition of A starting with the index i. Assume that the array A and B are stored in two files A.txt and B.txt.

Example:
Contents of A.txt
45 8 3 1 23 56 78 9 23 24 56 22 12 59 89 14

Contents of B.txt
0 4 13

The three partitions of A are:
{45, 8, 3}, {1, 23, 56, 78, 9, 23, 24, 56, 22, 12}, and {59, 89,14}

1. **Version-1:** Write a sequential implementation of your code
2. **Version-2:** Write a CUDA program using global memory to sort each partition of A that is defined by the array B, the sorted array should be saved in C.txt.
3. **Version-3:** Repeat Version-2 by using shared memory.

Report the results in the below format and explain the performance differences.

N	M	Thread Block Size	No. of thread blocks	Version-1 Execution time (V1)	Version-2 Execution time (V2)	Version-2 Speed up (V2/V1)	Version-3 Execution time	Version-3 Speed up (V3/V1)
1048576	128	1024	1024	To be filled	To be filled	To be filled	To be filled	To be filled
1048576	128	1024	512	To be filled	To be filled	To be filled	To be filled	To be filled
1048576	128	1024	256	To be filled	To be filled	To be filled	To be filled	To be filled
1048576	128	1024	128	To be filled	To be filled	To be filled	To be filled	To be filled
1048576	128	1024	64	To be filled	To be filled	To be filled	To be filled	To be filled
1048576	128	1024	32	To be filled	To be filled	To be filled	To be filled	To be filled