Roll no. 241110406

Assignment 4

Problem 1:

Timings:

CPU : 1.2790 ms

GPU:
Host to device Memcpy time (ms): 0.324064
Default Kernel time (ms): 0.70944
Device to host Memcpy time (ms): 0.92304

GPU (shared memory tiling):
Host to device Memcpy time (ms): 0.309696
    Tile size: 1:
    Memory tiled Kernel time (ms): 1.33254
    Tile size: 2:
    Memory tiled Kernel time (ms): 0.197344
    Tile size: 4:
    Memory tiled Kernel time (ms): 0.071744
    Tile size: 8:
    Memory tiled Kernel time (ms): 0.075008
Device to host Memcpy time (ms): 0.287488

GPU (pinned memory):
Host to device Memcpy time (ms): 0.2112
    Tile size: 1:
    Pinned Memory Kernel time (ms): 1.30733
    Tile size: 2:
    Pinned Memory Kernel time (ms): 0.17952
    Tile size: 3:
    Pinned Memory Kernel time (ms): 0.054752
    Tile size: 4:
    Pinned Memory Kernel time (ms): 0.061184
Device to host Memcpy time (ms): 0.21008

GPU (UVM support):
UVM Kernel time (ms): 1.19219

Problem 2:

The native CUDA kernel works well as long as the size of array is upto 2048. Beyond that the check_result function aborts the program saying that my output and reference output are not similar. Reference taken from GPU Gems book for writing this code (see here Chapter 39. Parallel Prefix Sum (Scan) with CUDA | NVIDIA Developer).

Time taken by Thrust version : 0.00496 ms

Time taken by host to device memcpy : 0.017184 ms
Time taken by cuda Kernel : 1.75597 ms
Time taken by device to host memcpy : 0.01328 ms

Problem 3:

Completely changed the implementation from loop to a backtracking algorithm which (correctly) predicts if the current iteration of the loop is useful or not and skips the iteration if it is not useful. Default thread stack size was 1KB which had to be increased to 4KB to correctly complete the recursive call till the end.

C code time : 553 seconds

Default Cuda kernel : 2.850501 seconds

Optimized Cuda kernel : 2.505030 seconds

UVM supported Kernel : 2.830323 seconds

Problem 4:

Default kernel implementation was straight-forward. For optimized kernel Shared memory tiling was used which caused ~15x speedup in 2D kernel, but very minor speedup in 3D kernel, which somehow was already 3x faster than 2D kernel. This most likely is caused due to internal optimizations performed by the cuda compiler.

Results:

2d kernel time on CPU: 0.11301 msec

2D kernel time on GPU:
host to device memcpy time (ms) : 0.022528
Kernel2D time (ms): 0.176064
device to host memcpy time (ms) : 0.014816

Optimized 2D kernel time on GPU
host to device memcpy time (ms) : 0.022528
Optimized Kernel2D time (ms): 0.012768
device to host memcpy time (ms) : 0.009024

3d kernel time on CPU: 22.2859 msec

3D kernel time on GPU
host to device memcpy time (ms) : 0.185856
Kernel3D time (ms): 0.067296
device to host memcpy time (ms) : 0.229696

Optimized 3D kernel time on GPU
host to device memcpy time (ms) : 0.185856
Optimized Kernel3D time (ms): 0.06208
device to host memcpy time (ms) : 0.191776