In the name of God

Multicore Programming Course

Lab 6 Report

Amir M Pirhosseinloo 9531014

Mahdi Safari 9531050

Serial version execution time: 1150.637 seconds

	Block size	Grid size	Block size	Grid size	Block size	Grid size
	8*8	512*512*1	16*16	256*256*1	32*32	128*128*1
Elapsed time	4722.5 ms		3401.0271 ms		3463.711426 ms	
Speed up	243.65		338.32		332.19	
Occupancy	68.65		73.28		85.71	

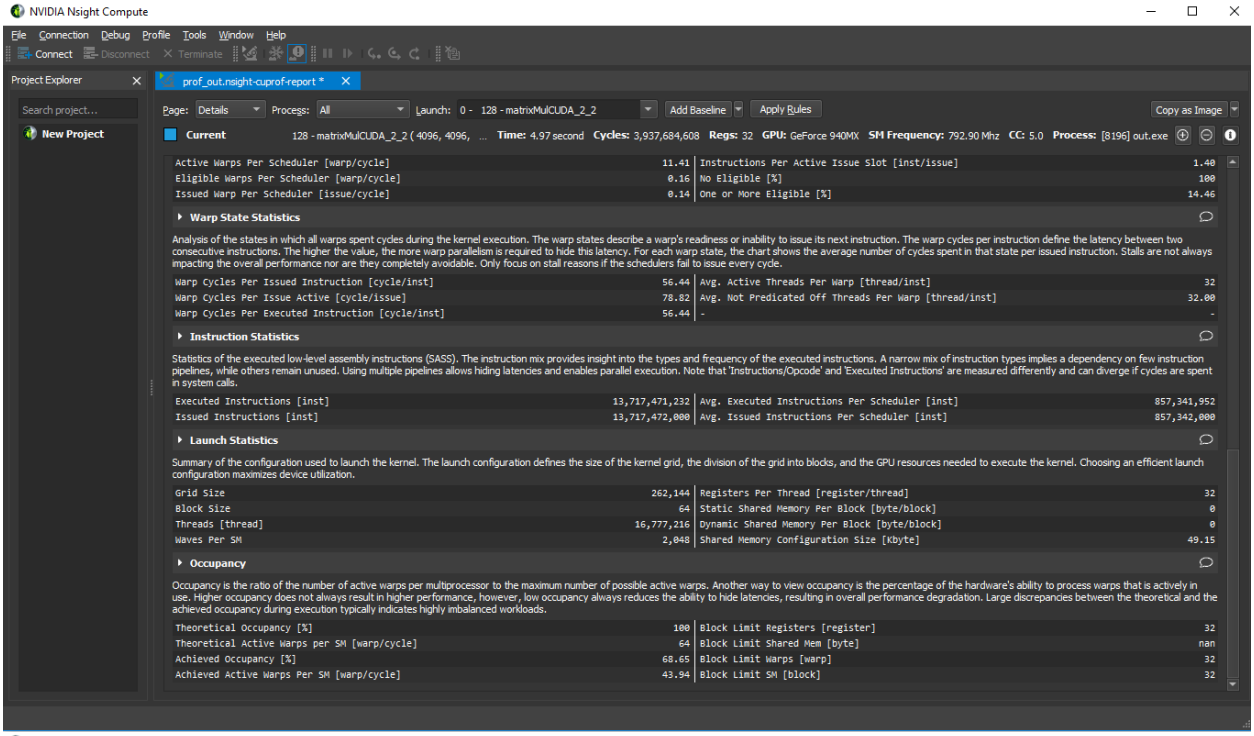
	Block size	Grid size	Block size	Grid size	Block size	Grid size
	32*32	128*128*1	32*32	256*256*1	32*32	1*1*1
solution 1 matrixMulCUDA_2_1	-	-	-	-	took very long time	
solution 2 matrixMulCUDA_2_2	-	-	27776.244 ms		-	-
solution 3 matrixMulCUDA_2_3	89040.695 ms		-	-	-	-

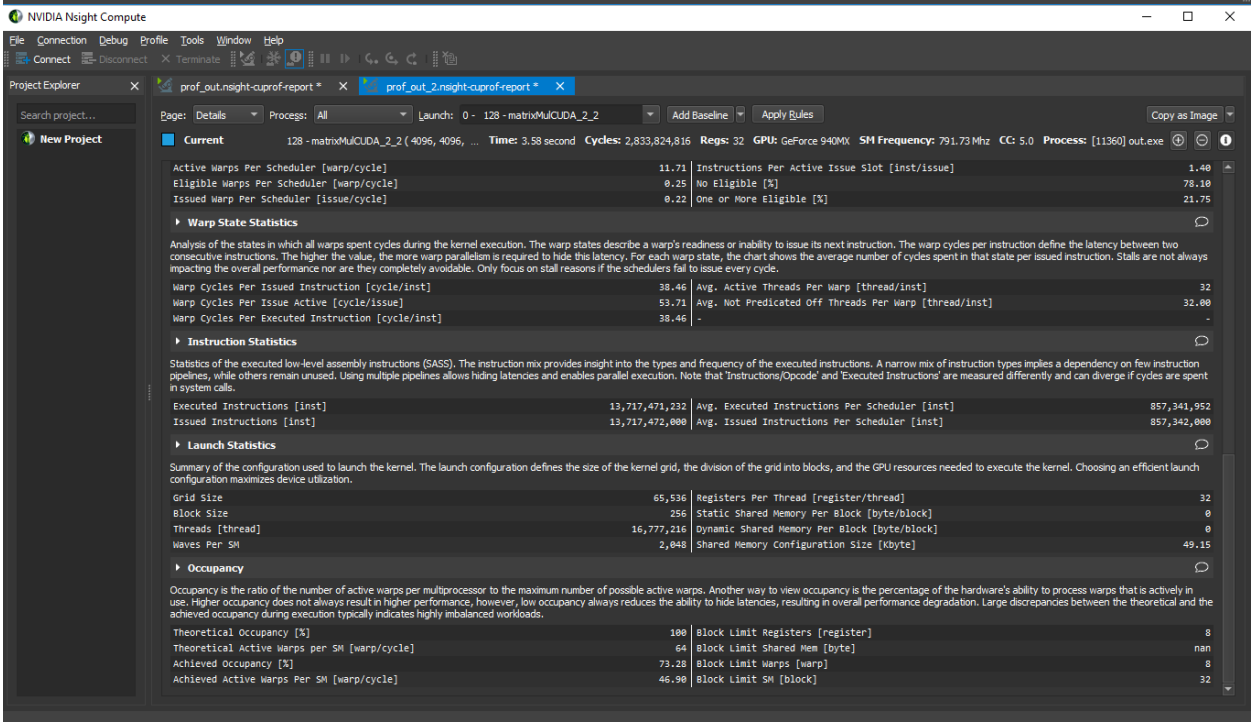
TILE_WIDTH in solution 3 = 2,

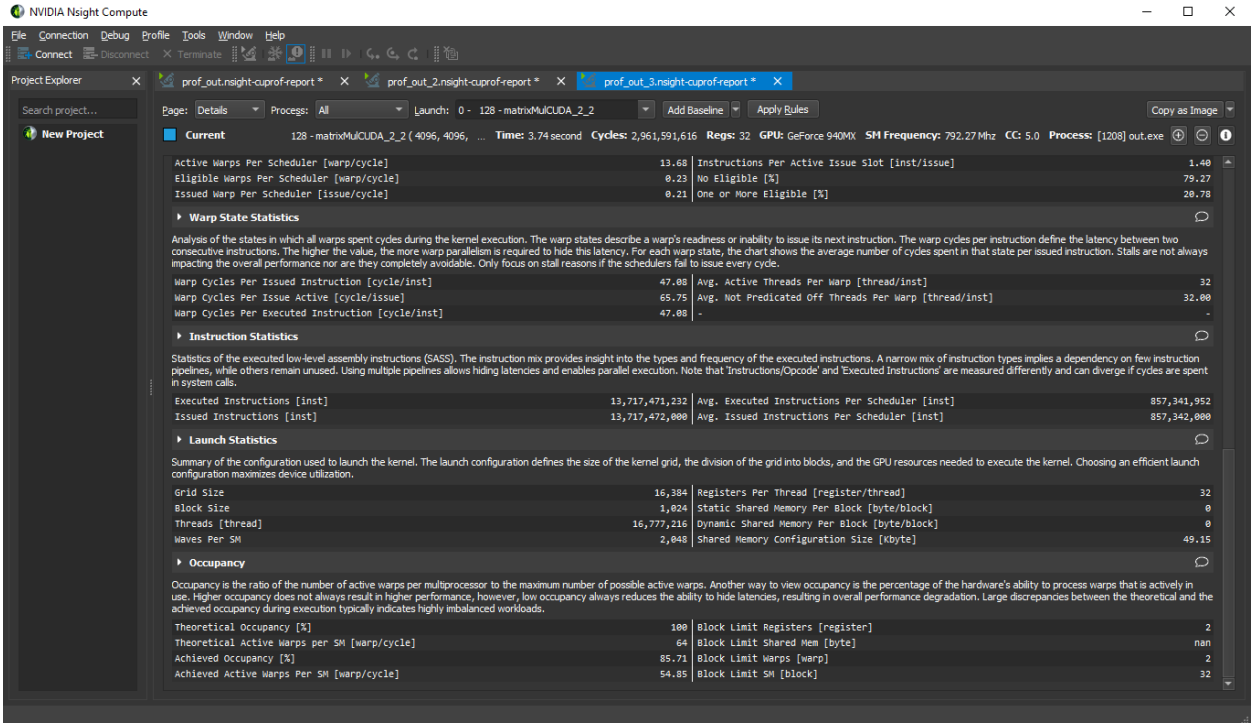
TILE_WIDTH in solution 1 = 256,

Input Arrays Dimensions: 8192 * 8192

```
__global__ void
matrixMulCUDA_3(float *C, float *A, float *B, int n){
        int start_row = blockDim.y * blockIdx.y * TILE_WIDTH + threadIdx.y * TILE_WIDTH;
        int end_row = start_row + TILE_WIDTH;
        int start_col = blockDim.x * blockIdx.x * TILE_WIDTH + threadIdx.x * TILE_WIDTH;
        int end_col = start_col + TILE_WIDTH;
        if(n < end_col) end_col = n;
        if(n < end_row) end_row = n;
        for (int row = start_row; row < end_row; row++) {
                for (int col = start_col; col < end_col; col++) {
                float C_val = 0;
                        for (int k = 0; k < n; ++k) {
                                float A_elem = A[row * n + k];
                                float B_elem = B[k * n + col];
                                C_val += A_elem * B_elem;
                        }
                        C[row*n + col] = C_val;
                }
        }
}
```


NVIDIA Nsight Compute
File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate
Project Explorer
prof_out.nsight-cuprof-report *
Search project...
New Project
Page: Details Process: All Launch: 0 - 128 - matrixMulCUDA_2_2
Add Baseline Apply Rules
Copy as Image
Current 128 - matrixMulCUDA_2_2 (4096, 4096, ... Time: 4.97 second Cycles: 3,937,684,608 Regs: 32 GPU: GeForce 940MX SM Frequency: 792.90 Mhz CC: 5.0 Process: [8196] out.exe
Active Warps Per Scheduler [warp/cycle] 11.41
Eligible Warps Per Scheduler [warp/cycle] 0.16
Issued Warp Per Scheduler [issue/cycle] 0.14
Instructions Per Active Issue Slot [inst/issue] 1.40
No Eligible [%] 100
One or More Eligible [%] 14.46
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.
Warp Cycles Per Issued Instruction [cycle/inst] 56.44
Warp Cycles Per Issue Active [cycle/issue] 78.82
Warp Cycles Per Executed Instruction [cycle/inst] 56.44
Avg. Active Threads Per Warp [thread/inst] 32
Avg. Not Predicated Off Threads Per Warp [thread/inst] 32.00
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 13,717,471,232
Issued Instructions [inst] 13,717,472,000
Avg. Executed Instructions Per Scheduler [inst] 857,341,952
Avg. Issued Instructions Per Scheduler [inst] 857,342,000
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 262,144
Block Size 64
Threads [thread] 16,777,216
Waves Per SM 2,048
Registers Per Thread [register/thread] 32
Static Shared Memory Per Block [byte/block] 0
Dynamic Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [Kbyte] 49.15
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp/cycle] 64
Achieved Occupancy [%] 68.65
Achieved Active Warps Per SM [warp/cycle] 43.94
Block Limit Registers [register] 32
Block Limit Shared Mem [byte] nan
Block Limit Warps [warp] 32
Block Limit SM [block] 32


NVIDIA Nsight Compute
File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate
Project Explorer
prof_out.nsight-cuprof-report *
prof_out_2.nsight-cuprof-report *
Search project...
New Project
Page: Details Process: All Launch: 0 - 128 - matrixMulCUDA_2_2
Add Baseline Apply Rules
Copy as Image
Current 128 - matrixMulCUDA_2_2 (4096, 4096, ... Time: 3.58 second Cycles: 2,833,824,816 Regs: 32 GPU: GeForce 940MX SM Frequency: 791.73 Mhz CC: 5.0 Process: [11360] out.exe
Active Warps Per Scheduler [warp/cycle] 11.71
Eligible Warps Per Scheduler [warp/cycle] 0.25
Issued Warp Per Scheduler [issue/cycle] 0.22
Instructions Per Active Issue Slot [inst/issue] 1.40
No Eligible [%] 78.10
One or More Eligible [%] 21.75
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.
Warp Cycles Per Issued Instruction [cycle/inst] 38.46
Warp Cycles Per Issue Active [cycle/issue] 53.71
Warp Cycles Per Executed Instruction [cycle/inst] 38.46
Avg. Active Threads Per Warp [thread/inst] 32
Avg. Not Predicated Off Threads Per Warp [thread/inst] 32.00
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 13,717,471,232
Issued Instructions [inst] 13,717,472,000
Avg. Executed Instructions Per Scheduler [inst] 857,341,952
Avg. Issued Instructions Per Scheduler [inst] 857,342,000
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 65,536
Block Size 256
Threads [thread] 16,777,216
Waves Per SM 2,048
Registers Per Thread [register/thread] 32
Static Shared Memory Per Block [byte/block] 0
Dynamic Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [Kbyte] 49.15
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp/cycle] 64
Achieved Occupancy [%] 73.28
Achieved Active Warps Per SM [warp/cycle] 46.90
Block Limit Registers [register] 8
Block Limit Shared Mem [byte] nan
Block Limit Warps [warp] 8
Block Limit SM [block] 32


NVIDIA Nsight Compute
File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate
Project Explorer
Search project...
New Project
prof_out.nsight-cuprof-report *
prof_out_2.nsight-cuprof-report *
prof_out_3.nsight-cuprof-report *
Page: Details Process: All Launch: 0 - 128 - matrixMulCUDA_2_2 Add Baseline Apply Rules Copy as Image
Current 128 - matrixMulCUDA_2_2 (4096, 4096, ... Time: 3.74 second Cycles: 2,961,591,616 Regs: 32 GPU: GeForce 940MX SM Frequency: 792.27 Mhz CC: 5.0 Process: [1208] out.exe
Active Warps Per Scheduler [warp/cycle] 13.68
Eligible Warps Per Scheduler [warp/cycle] 0.23
Issued Warp Per Scheduler [issue/cycle] 0.21
Instructions Per Active Issue Slot [inst/issue] 1.40
No Eligible [%] 79.27
One or More Eligible [%] 20.78
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.
Warp Cycles Per Issued Instruction [cycle/inst] 47.08
Warp Cycles Per Issue Active [cycle/issue] 65.75
Warp Cycles Per Executed Instruction [cycle/inst] 47.08
Avg. Active Threads Per Warp [thread/inst] 32
Avg. Not Predicated Off Threads Per Warp [thread/inst] 32.00
-
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 13,717,471,232
Issued Instructions [inst] 13,717,472,000
Avg. Executed Instructions Per Scheduler [inst] 857,341,952
Avg. Issued Instructions Per Scheduler [inst] 857,342,000
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 16,384
Block Size 1,024
Threads [thread] 16,777,216
Waves Per SM 2,048
Registers Per Thread [register/thread] 32
Static Shared Memory Per Block [byte/block] 0
Dynamic Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [Kbyte] 49.15
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp/cycle] 64
Achieved Occupancy [%] 85.71
Achieved Active Warps Per SM [warp/cycle] 54.85
Block Limit Registers [register] 2
Block Limit Shared Mem [byte] nan
Block Limit Warps [warp] 2
Block Limit SM [block] 32