EE217: GPU Architecture and Parallel Programming
Lab #1

Akber Raza
araza008@ucr.edu

January 28, 2019

GPGPU-Sim Results

Naive

```
gpu_sim_cycle = 7825
gpu_sim_insn = 2342720
gpu_ipc = 299.3891
gpu_tot_sim_cycle = 7825
gpu_tot_sim_insn = 2342720
gpu_tot_ipc = 299.3891
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 470
gpu_stall_icnt2sh = 8996
gpu_total_sim_rate = 780906

Warp Occupancy Distribution:
Stall:7129      W0_Idle:37256     W0_Scoreboard:42481
W1:12096        W2:6144  W3:0     W4:6144  W5:0     W6:0     W7:0     W8:6144
W9:0     W10:0    W11:0    W12:0    W13:0    W14:0    W15:0    W16:6144
W17:0    W18:0    W19:0    W20:0    W21:0    W22:0    W23:0    W24:0
W25:0    W26:0    W27:0    W28:0    W29:0    W30:0    W31:0    W32:70144
```

Optimized

```
gpu_sim_cycle = 5781
gpu_sim_insn = 2047936
gpu_ipc = 354.2529
gpu_tot_sim_cycle = 5781
gpu_tot_sim_insn = 2047936
gpu_tot_ipc = 354.2529
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 574
gpu_stall_icnt2sh = 8984
gpu_total_sim_rate = 1023968

Warp Occupancy Distribution:
Stall:5502      W0_Idle:39493     W0_Scoreboard:36307
W1:448   W2:256   W3:0     W4:256   W5:0     W6:0     W7:0     W8:256
W9:0     W10:0    W11:0    W12:0    W13:0    W14:0    W15:0    W16:256
W17:0    W18:0    W19:0    W20:0    W21:0    W22:0    W23:0    W24:0
W25:0    W26:0    W27:0    W28:0    W29:0    W30:0    W31:0    W32:66816
```

Questions

For the naive reduction kernel, how many steps execute without divergence? How many steps execute with divergence?

Only the first step is non-divergent while all subsequent 14 steps are divergent.

For the optimized reduction kernel, how many steps execute without divergence? How many steps execute with divergence?

No divergence in first 10 steps; final 5 steps are divergent.

Which kernel performed better? (for both real GPUs and GPGPU-Sim)

Optimized

How does the warp occupancy distribution compare between the two Reduction implementations?

Naive reduction implementation resulted in a considerable number of cycles used when only few number of threads are active. For example, 12,096 cycles are used when only one thread is active. However, optimized implementation caused significantly fewer number of cycle to be used when less than all threads were active.

Why do GPGPUs suffer from warp divergence?

Once a thread block is allocated to SM, threads are divided into groups of 32 which are called warps. If each thread takes a branch different from other threads in the same warp, performance is degraded by a factor of 32 because other threads remain inactive during the process.