Hairu-Wen hwen020@ucr.edu

Assume we run reduction with an input size of 1,000,000 and thread block size of 512 threads.

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

In the first step, there is no warp divergence (all active threads in a block is divisible by 32). For the remaining steps, all warps are divergent. Within each block we have 512 threads and our reduction takes 10 steps operating on 1024.

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

The first step has no warp divergence. For the first 976 blocks that operate on 1024 elements each, divergence only occurs when they have less than 32 threads active; therefore the first 5 steps are non-divergent (512, 256, 128, 64, 32 active threads).

In subsequent steps, the number of active threads are not divisible by 32, and therefore are divergent.

3、Which kernel performed better? Use profiling statistics to support your claim.

Individual kernel evaluation(on gpgpu-sim):

	NaiveReduction	OptimizedReduction	
gpu_sim_cycle	124718	93773	↑：longer execution time； performance↓
gpu_sim_insn	71024154	63023501	↑：complexity / workload ↑
gpu_ipc	569.4780	672.0858	↑：efficiency↑

Overall application evaluation(on gpgpu-sim):

gpu_tot_sim_cycle	124718	93773	↑：longer execution time； performance↓
gpu_tot_sim_insn	71024154	63023501	↑：complexity / workload ↑
gpu_tot_ipc	569.4780	672.0858	↑：efficiency↑

A higher gpu_sim_cycle indicates longer execution time. A better-performing kernel will typically have a higher IPC and lower execution time.

naiveReduction execute time: gpgpu_simulation_time = 0 days, 0 hrs, 1 min, 39 sec (99 sec) (0.098253 s on Bender)
optimizedReduction execute time: gpgpu_simulation_time = 0 days, 0 hrs, 2 min, 9 sec (129 sec) (0.128574 s on Bender)

Obviously OptimizedReduction is better.

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

NaiveReduction：

```
Warp Occupancy Distribution:
Stall:75038 W0_Idle:120291  W0_Scoreboard:336097    W1:13678
    W2:7816 W3:0    W4:7816 W5:0    W6:0    W7:0    W8:7816 W9:0
    W10:0   W11:0   W12:0   W13:0   W14:0   W15:0   W16:7816    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   W32:2024274
```

OptimizedReduction：

```
Warp Occupancy Distribution:
Stall:73320 W0_Idle:249186  W0_Scoreboard:330330    W1:14655
    W2:8793 W3:0    W4:8793 W5:0    W6:0    W7:0    W8:8793 W9:0
    W10:0   W11:0   W12:0   W13:0   W14:0   W15:0   W16:8793    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   W32:2054561
```

OptimizedReduction has a higher number of fully active warps.(W32)

5、Why do GPGPUs suffer from warp divergence?

GPGPUs often suffer from warp divergence due to conditional branching within warps. When threads within a warp take different code paths (e.g., if-else conditions), warp divergence occurs. This leads to inefficiency as different threads in the same warp need to execute different instructions, causing stalls.