Name: Aparna Mohan, Netid: amoha121, SID: 862396080

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

All threads in a block participate in the first step. Hence, it executes without divergence. However, all other 9 steps execute with divergence. So, the total number of steps for BLOCK_SIZE=512 is log2(512*2) = 10 steps.

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

In the optimized version, the first five steps execute without divergence as the number of threads is smaller than the warp. Rest of the 5 steps executes with divergence as the threads within a warp take different paths.

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

From the profiling statistics obtained after the run:

```
kernel_name = _Z18optimizedReductionPfS_j
kernel_launch_uid = 1
gpu_sim_cycle = 88656
gpu_sim_insn = 62023582
gpu_ipc =     699.5983
gpu_tot_sim_cycle = 88656
gpu_tot_sim_insn = 62023582
gpu_tot_ipc =     699.5983
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 4291
gpu_stall_icnt2sh    = 42601
gpu_total_sim_rate=224723
kernel_name = _Z14naiveReductionPfS_j
kernel_launch_uid = 1
gpu_sim_cycle = 124829
gpu_sim_insn = 70523930
gpu_ipc =     564.9643
gpu_tot_sim_cycle = 124829
gpu_tot_sim_insn = 70523930
gpu_tot_ipc =     564.9643
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 1905
gpu_stall_icnt2sh    = 10996
gpu_total_sim_rate=199784
```

Hence optimized reduction outperforms naive reduction. Optimized reduction has taken lesser gpu cycles (88656) compared to naive reduction (124829). Optimized reduction also has lesser gpu_sim_insn, meaning it executed fewer instructions. Instructions per cycle is greater in the case of optimized reduction hence it has achieved a higher instruction throughput. GPU stall dramfull value is also lower in optimized reduction indicating that it experienced fewer stalls.

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

Warp Occupancy Distribution Naïve Reduction

```
Warp Occupancy Distribution:
Stall:143415    W0_Idle:65095   W0_Scoreboard:294890    W1:369306       W2:187584       W3:0    W4:187584       W5:0    W6:0    W7:0    W8:1875
84      W9:0    W10:0   W11:0   W12:0   W13:0   W14:0   W15:0   W16:187584      W17:0   W18:0   W19:0   W20:0   W21:0   W22:0   W23:0   W24:0 W
25:0    W26:0   W27:0   W28:0   W29:0   W30:0   W31:0   W32:2079014
```

Warp Occupancy Distribution Optimized Reduction

```
Warp Occupancy Distribution:
Stall:88670     W0_Idle:112299  W0_Scoreboard:394589    W1:13678        W2:7816 W3:0    W4:7816 W5:0    W6:0    W7:0    W8:7816 W9:0    W10:0 W
11:0    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 W
28:0    W29:0   W30:0   W31:0   W32:1993024
```

The warp stall for naive reduction is more when compared to optimized reduction. Thus, a comparison between the warp stall is made to get the occupancy distribution.

5. **Why do GPGPUs suffer from warp divergence?**

The fundamental design of GPU involves SIMD (Single instruction, Multiple Data) architecture. Due to this GPGPUs suffer from warp divergence. It occurs when threads within a warp follow different execution paths. It occurs when they do not access adjacent memory locations, varying input data, many conditional branches, function calls, loop iterations etc. The amount of data processing is greater than the data scheduling and transmission. It will lower the performance of GPGPU suffers from wrap divergence.