CS217 : LAB ASSIGNMENT 3

Name: Akash Suresh Bilgi | SID: 862395080 | Email: abilg003@ucr.edu

1. On Bender, compare the execution time of a 256 x 256 square matrix multiplication compared to a 1024 x 64 and 64 x 1024 rectangular matrix multiply. All input matrices have 65k entries. What do you observe? Which is faster? Can you explain the observed behavior? Tip: You may want to comment out the verify() function in main.cu when timing this question.

   A. 256x265 is faster with kernel time of 0.000088 secs , where as 1024*64 takes 0.000211 secs. Since there are more mem accesses and context switching occurring in rectangular (1024x64) than in square matrix.

2. Conceptual Question: For a 64 square *tiled* matrix multiplication, how many times is each element of the input matrices loaded from global memory? Assume 16x16 tiles.

   A. 64/16 = 4 times

3. Conceptual Question: For a 64 square *non-tiled* matrix multiplication, how many times is each element of the input matrices loaded from global memory?

   A. 1 time for each => 64 times

4. GPGPU-Sim related question: In this part, we will compare the execution of a 128x128 square tiled matrix multiplication across different tile sizes. Run ./sgemm-tiled 128 in GPGPU-Sim with TILE_SIZE of 8, 16 (default), and 32. Fill the following table:

Tile size	8	16	32	Note
gpu_tot_sim_cycle	40101	26817	56031	Total cycles
gpu_tot_ipc	417.5569	448.4415	399.4315	Instruction per cycle
gpgpu_n_load_insn	524288	262144	131072	Total loads to global memory
gpgpu_n_store_insn	16384	16384	16384	Total stores to global memory
gpgpu_n_shmem_insn	4718592	4456448	4325376	Total accesses to shared memory

5. Which tile size resulted in the least number of accesses to global memory? Which tile size resulted in the most number of accesses to global memory? What is the reasoning behind this observation?

A. 32 had the lowest access and 8 had the most. Reasoning: As the tile grows the number of accesses to the global memory decreases.

6. Which tile size performed the fastest, which tile size performed the slowest? Why do you think that is?

A. The performance distribution looks like a bell curve (ref: q4).
8 tiles has worse performance.
16 tiles has best performance,
32 has the worst performance.

This happened due to the limited number of threads and shared mem, and 16 tile size is the sweet spot where the more threads and shared memory can be balanced, and execute more instructions per cycle.