To: Christina Taylor
From: Anwar Khaddaj
Date: April 24, 2023
Subject: CAAM 420/520 – Homework 5
I collaborated with Arshia Singhal, Jonathan Cangelosi and Prani Nalluri on this homework.

Problem 1 Submitted as code.

Problem 2 Kernel Characteristics:

Setup	Block	Wrapped	Line
Ele/thread	1	N	n_rows
Blocks/SM	2	64	64
GMT per block	1	1	1
GMT entire grid	N_rows*N_cols	(n_rows/N)*N_cols	N_cols

(a)
- Block Setup: Every thread will process 1 element only as we have 32×32 threads (32 warps) and 32×32 elements per memory block.
- Wrapped Setup: Every thread will process N elements as we have N×32 elements per memory block and we have a thread block consisting of 1 warp (32 threads) that will access the first row of the block and then the same warp will access the rest of the rows. Since we have N rows, every thread will process N elements.
- Line Setup: Every thread will process n_rows elements as we now have n_rows×32 elements per memory block so the reasoning follows similar to the wrapped setup.

(b)
- Block Setup: We have 32×64 threads per SM, and each thread block is 32×32, so we can have at most 2 thread blocks running at the same time.
- Wrapped Setup: Each thread block will be 1×32 so we can run 64 blocks at the same time.
- Line Setup: Similar to the wrapped line setup, we can run 64 blocks at the same time.

(c) For all the setups, the number of global memory transactions for a single thread block for just the array D is 1 as we only need 32 elements to read from D for each block.

(d) Since the number of global memory transactions is 1 per block for each setup, the number of global memory transactions per entire grid will be the number of blocks in the grid depending on the setup. So the answer will be N_rows×N_cols for block setup, (n_rows/N)×N_cols for wrapped setup, and N_cols for line setup.

Problem 3 64KB=64000 bytes. 1 float=4 bytes.

(a) Each chunk from A, D and B has 32×32, 1×32, and 32×32 floats respectively, so we have a total of 32×32+2×32+32×32=2080 floats=8320 bytes for each block. Thus, the most blocks that could run on a single SM is: 64000 bytes / 8320 bytes per block $=7.69 \approx 7$ blocks.

(b) Each chunk from A, D and B has N×32, 1×32, and N×32 floats respectively.

- Most blocks: Let N=1. Thus, the total number of floats per block: 1×32+1×32+1×32=96 floats=384 bytes, so the blocks that could run on a single SM is: 64000 bytes / 384 bytes per block = 166.66 ≈ 166 blocks.
- Fewest blocks: Since we don't know what n_rows is, choosing N=n_rows might be risky as we might end up getting 0 blocks. Thus, we need to ensure the fewest blocks is at least 1. (This calculation is computed in part c and we get n_rows = 249) Thus, we need to choose N=min{n_rows, 249}. Therefore, the total number of floats per block: N×32×2+1×32 floats = (N×32×2+1×32)×4 bytes per block. Thus, the number of blocks that could run on a single SM:

$$\left\lfloor \frac{64000}{(N \times 32 \times 2 + 1 \times 32) \times 4} \right\rfloor = \max\{1, \left\lfloor \frac{64000}{(\text{n\_rows} \times 32 \times 2 + 1 \times 32) \times 4} \right\rfloor\}$$

(c) Each chunk from A, D and B has n_rows×32, 1×32, and n_rows×32 floats respectively, so we have a total of 2×n_rows×32+1×32 floats = (2×n_rows×32+1×32)×4 bytes. Thus, for a single block's shared memory to fit on an SM, we solve the following equation:

$$\frac{64000}{(2 \times \text{n\_rows} \times 32 + 1 \times 32) \times 4} = 1$$

$$64000 = (2 \times \text{n\_rows} \times 32 + 1 \times 32) \times 4$$

$$\text{n\_rows} = \left\lfloor \frac{64000 - 32 \times 4}{2 \times 32 \times 4} \right\rfloor = \lfloor 249.5 \rfloor \approx 249.$$

Problem 4 For parts a-c, the kernel will invoke the blocks as per the grid size (domain) for each setup, and since for each memory block, we have 1 thread block, so the number of thread blocks=the number of memory blocks.

(a) The kernel will use N_rows x N_cols = 40×10 = 400 thread blocks. (each thread block is 32×32 and we have 1280×320 domain)

(b) The kernel will use (n_rows/N) × N_cols thread blocks. The most thread blocks are (for N=1) n_rows × N_cols = 1280×10 = 12800 thread blocks, and the fewest thread blocks are (for N=n_rows=1280) 1 × Ncols = 10 thread blocks.

(c) The kernel will use 1×N_cols = 1×10 = 10 thread blocks.

(d) The setup that will most likely allow two of its kernels to run at the same time is the line setup and the wrapped setup (for large N i.e N=n_rows). The reason is that the memory blocks for these setups are larger in size but fewer in number than the memory blocks for the block setup, so knowing that we cannot split the memory blocks across SMs, we need fewer SMs for wrapped and line setup allowing us to run two kernels at the same time given SM usage.

(e) The setup that will see the biggest improvement in concurrency and run-time is the block setup and the wrapped setup (for small N) as we have now more potential to run blocks and thus kernels at the same time given that we have more SMs now. As for wrapped and line setups, they might not even need that many SMs, so expanding the number might not dramatically improve the performance.

(f) The factors affecting the choice of N knowing that we plan on having kernel concurrency are:

- Global memory transactions: Knowing that shared memory is within an SM and is shared between kernels, this will affect N as we need to choose N that results in less global memory transactions, so that a kernel doesn't take up the full shared memory.
- Cost of assigning thread blocks to SMs: If the cost is too high, we need N to be large so that we assign fewer thread blocks to SMs thus achieving kernel concurrency.
- SM usage for the kernel (blocks/SM): We need a larger N then to make sure that fewer SMs are used (similar to what we had in part d); thus ensuring better kernel concurrency.