Programming Assignment IV

1. Conv2dV1.cu (Without Tiling Optimization)

a) Input Parameters:

- float* input: Pointer to the padded image data
- float* filter: Pointer to the filter data
- float* output: Pointer to store the convolution result
- H, W: Padded image height and width
- N, K: Total number of images (N) and Total number of filters (K)

b) Thread Indexing:

- $int\ tid = blockIdx.x * blockDim.x + threadIdx.x;\ //\ global\ thread\ ID$

- $int\ n = tid\ /\ (K * (H - 2) * (W - 2));\ //\ Determine\ the\ image\ number$
- $int\ k = (tid\ \%\ (K * (H - 2) * (W - 2)))\ /\ ((H - 2) * (W - 2));\ //\ filter$
- $int\ h = ((tid\ \%\ ((H - 2) * (W - 2)))\ /\ (W - 2)) + 1;\ //\ image\ row$
- $int\ w = (tid\ \%\ (W - 2)) + 1;\ //\ image\ column$

c) Convolution:

- Threads iterate over the filter elements, multiply them with corresponding input image elements, and accumulate the result.

This version of the kernel implementation uses global memory to store the input and filter data. The use of __global__ dictates that the GPU is using the global memory for its computation. The next version (V2) will try to optimize the memory usage by introducing the concept of tiling where we will be using the shared memory to load a tile of the input image and iterate through the entire image in tiles to compute our convolution.

2. Conv2dV2.cu (With Tiling Optimization)

a) Input Parameters:

- float* input: Pointer to the padded image data
- float* filter: Pointer to the filter data
- float* output: Pointer to store the convolution result
- H, W: Input image height and width

- N, K: Total number of images (N) and Total number of filters (K)

b) Shared Memory Parameters:

- $\_\_shared\_\_ float\ tile[TILE\_SIZE\ +\ 2][TILE\_SIZE\ +\ 2];$
- The TILE_SIZE should be less than or equal to BLOCK_SIZE for efficient usage.
- $\#define\ TILE\_SIZE\ 16\ //\ macro\ for\ tile\ size$

c) Convolution:

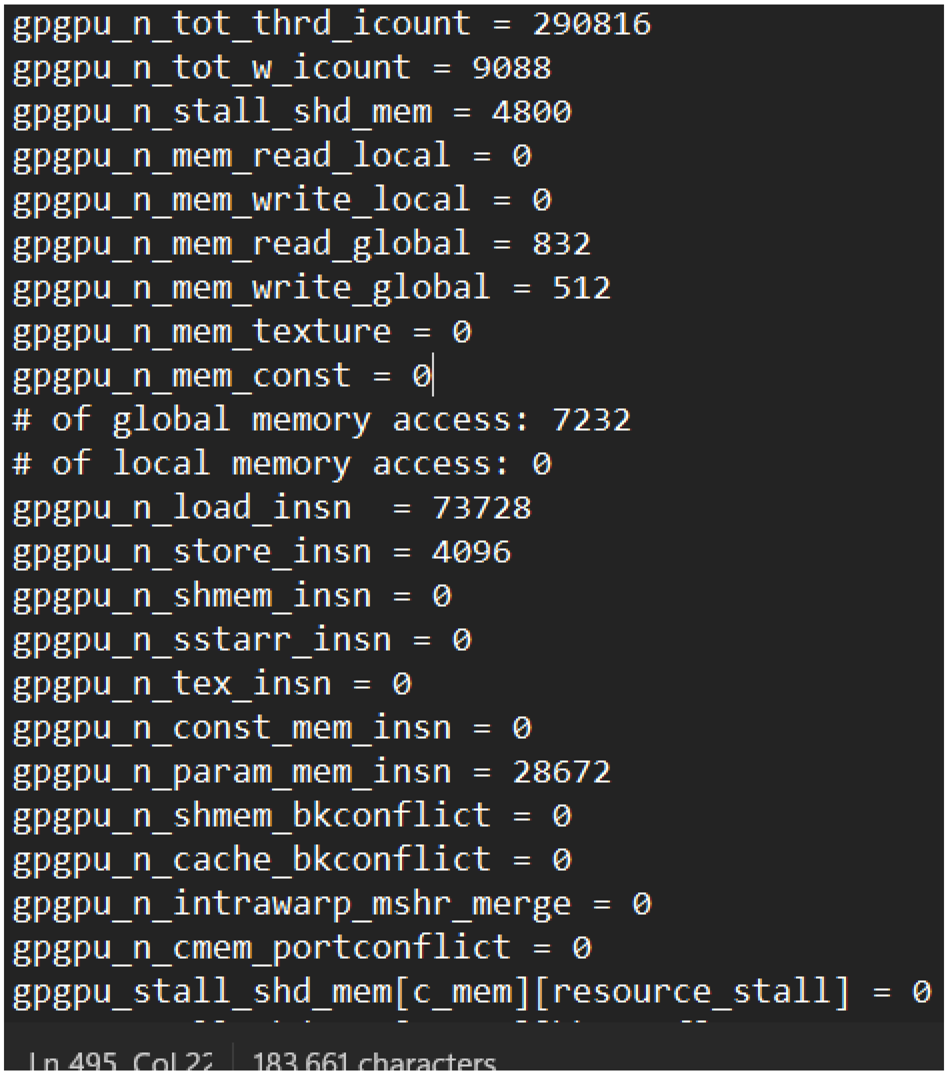
- We will loop over the tiles to generate a small part of the output. We will keep on looping until we exhaust all our N images to generate our output. The output size will be (N*K*inputheight) x inputWidth.

This version of the kernel implementation better utilizes our memory usage and optimizes the kernel convolution of N images with K filters. I tried to use the padded image to perform my convolution but the index calculations were becoming messy to handle so I switched to the original input image and took care of the boundary conditions.

3. GPGPU-SIM

- We will see how our program functions in the GPGPU SIM. For this we will take the smallest input files of all the cases (conv_input0.txt, conv_filter0.txt).

- **conv2dV1.cu (gpgpu_n_mem_read_global)**

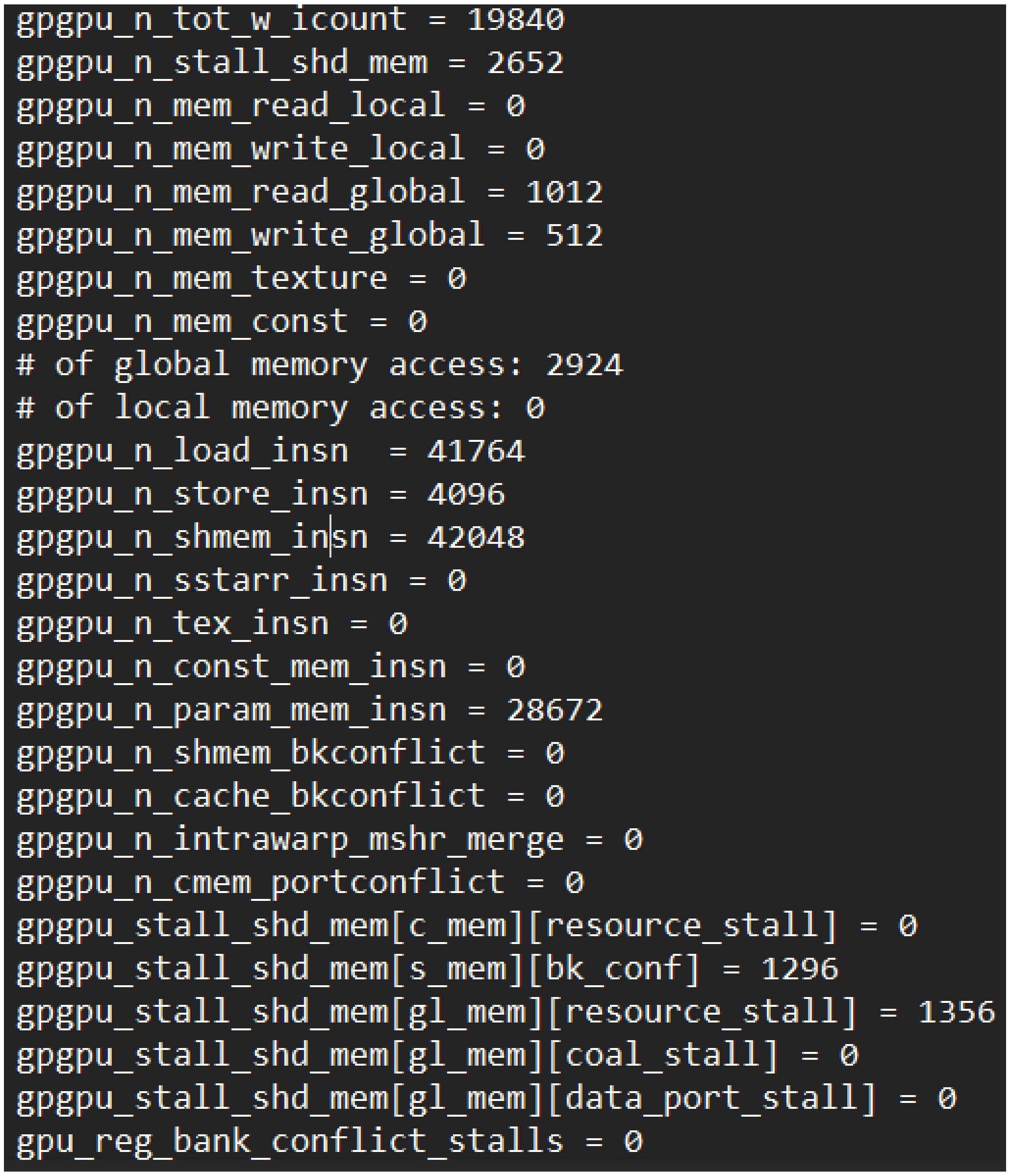


```
gpgpu_n_tot_thrd_icount = 290816
gpgpu_n_tot_w_icount = 9088
gpgpu_n_stall_shd_mem = 4800
gpgpu_n_mem_read_local = 0
gpgpu_n_mem_write_local = 0
gpgpu_n_mem_read_global = 832
gpgpu_n_mem_write_global = 512
gpgpu_n_mem_texture = 0
gpgpu_n_mem_const = 0
# of global memory access: 7232
# of local memory access: 0
gpgpu_n_load_insn  = 73728
gpgpu_n_store_insn = 4096
gpgpu_n_shmem_insn = 0
gpgpu_n_sstarr_insn = 0
gpgpu_n_tex_insn = 0
gpgpu_n_const_mem_insn = 0
gpgpu_n_param_mem_insn = 28672
gpgpu_n_shmem_bkconflict = 0
gpgpu_n_cache_bkconflict = 0
gpgpu_n_intrawarp_mshr_merge = 0
gpgpu_n_cmem_portconflict = 0
gpgpu_stall_shd_mem[c_mem][resource_stall] = 0
```

Ln 495, Col 22 | 183,661 characters

Since this version of the kernel doesn't use the shared memory, we can see that the local read and writes are 0. We can see that our global memory is being accessed which is to be expected. We can also see that there are no shared memory instructions indicating that shared memory has not been accessed.

- **conv2dV2.cu (gpgpu_n_mem_read_global)**



```
gpgpu_n_tot_w_icount = 19840
gpgpu_n_stall_shd_mem = 2652
gpgpu_n_mem_read_local = 0
gpgpu_n_mem_write_local = 0
gpgpu_n_mem_read_global = 1012
gpgpu_n_mem_write_global = 512
gpgpu_n_mem_texture = 0
gpgpu_n_mem_const = 0
# of global memory access: 2924
# of local memory access: 0
gpgpu_n_load_insn  = 41764
gpgpu_n_store_insn = 4096
gpgpu_n_shmem_insn = 42048
gpgpu_n_sstarr_insn = 0
gpgpu_n_tex_insn = 0
gpgpu_n_const_mem_insn = 0
gpgpu_n_param_mem_insn = 28672
gpgpu_n_shmem_bkconflict = 0
gpgpu_n_cache_bkconflict = 0
gpgpu_n_intrawarp_mshr_merge = 0
gpgpu_n_cmem_portconflict = 0
gpgpu_stall_shd_mem[c_mem][resource_stall] = 0
gpgpu_stall_shd_mem[s_mem][bk_conf] = 1296
gpgpu_stall_shd_mem[gl_mem][resource_stall] = 1356
gpgpu_stall_shd_mem[gl_mem][coal_stall] = 0
gpgpu_stall_shd_mem[gl_mem][data_port_stall] = 0
gpu_reg_bank_conflict_stalls = 0
```

In this version of the kernel we can see that the shared memory is being utilized. The variable gpgpu_n_shmem_insn indicates that there are shared memory instructions being used in the kernel as opposed to the V1 of the program. We can also see that there are no bank conflicts indicated by the gpgpu_n_shmem_bkconflict. While this may not be the most optimized code for our convolution problem, it's slightly better than our V1 version of the code.