GPU Teaching Kit

Accelerated Computing

Module 8.2 – Parallel Computation Patterns (Stencil)

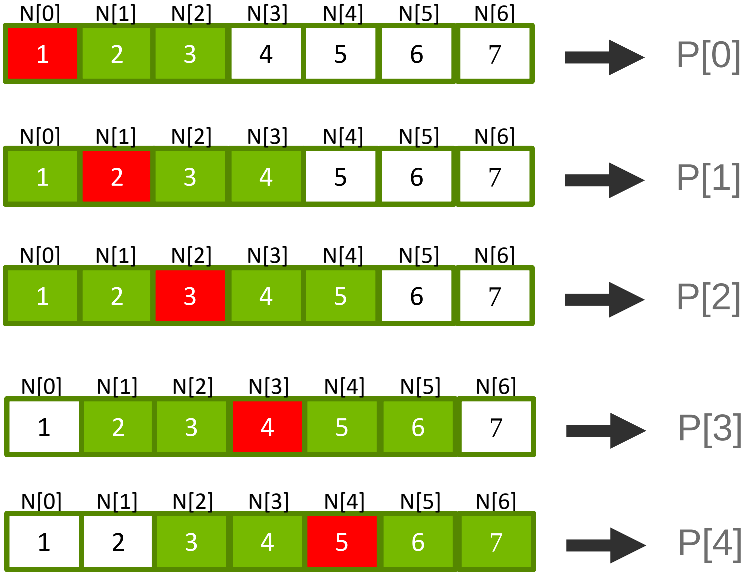
Tiled Convolution

Objective

- To learn about tiled convolution algorithms
  - Some intricate aspects of tiling algorithms
  - Output tiles versus input tiles

Tiling Opportunity Convolution

- Calculation of adjacent output elements involve shared input elements
  - E.g., N[2] is used in calculation of P[0], P[1], P[2]. P[3 and P[5] assuming a 1D convolution Mask_Width of width 5
- We can load all the input elements required by all threads in a block into the shared memory to reduce global memory accesses



N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	
1	2	3	4	5	6	7	→ P[0]

N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	
1	2	3	4	5	6	7	→ P[1]

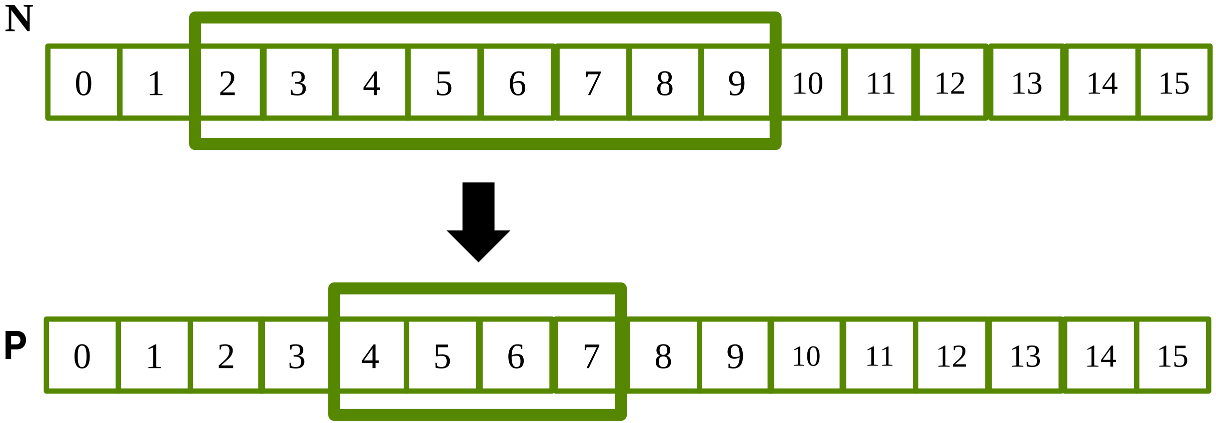
N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	
1	2	3	4	5	6	7	→ P[2]

N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	
1	2	3	4	5	6	7	→ P[3]

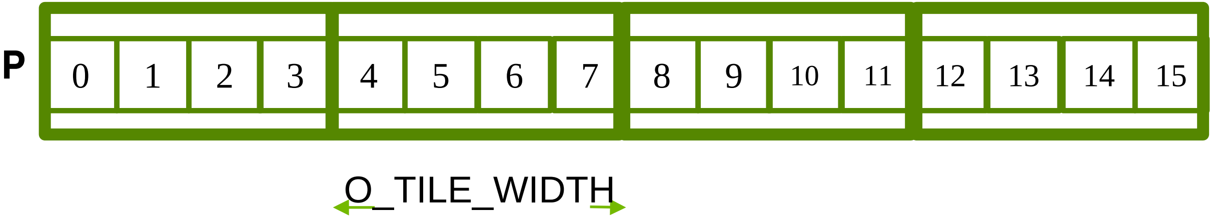
N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	
1	2	3	4	5	6	7	→ P[4]

Input Data Needs

- Assume that we want to have each block to calculate T output elements
  - T + Mask_Width -1 input elements are needed to calculate T output elements
  - T + Mask_Width -1 is usually not a multiple of T, except for small T values
  - T is usually significantly larger than Mask_Width



Definition – output tile



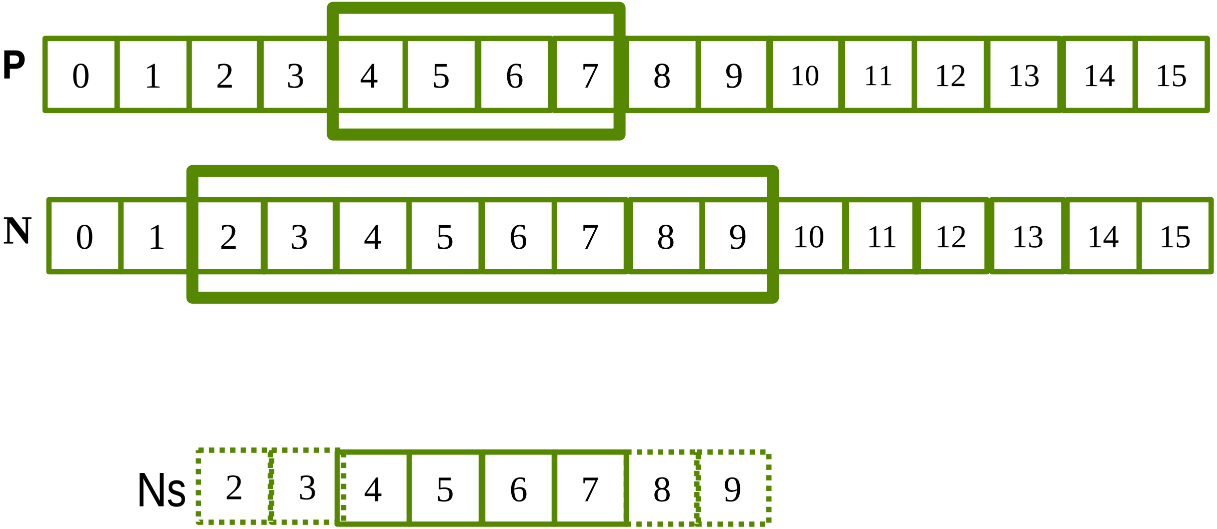
Each thread block calculates an output tile

Each output tile width is O_TILE_WIDTH

For each thread,

O_TILE_WIDTH is 4 in this example

Definition - Input Tiles



P 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

N 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Ns 2 3 4 5 6 7 8 9

Each input tile has all values needed to calculate the corresponding output tile.

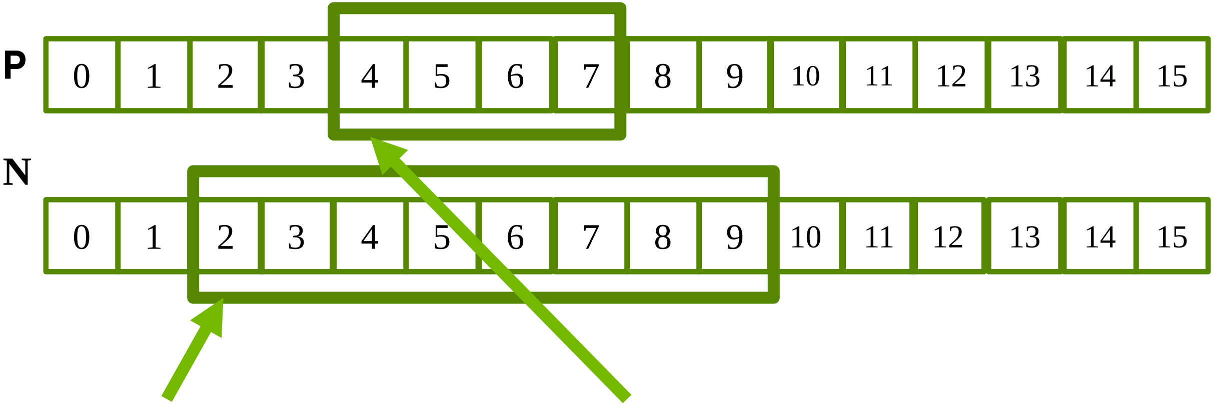
Two Design Options

- Design 1: The size of each thread block matches the size of an output tile
  - All threads participate in calculating output elements
  - blockDim.x would be 4 in our example
  - Some threads need to load more than one input element into the shared memory

- Design 2: The size of each thread block matches the size of an input tile
  - Some threads will not participate in calculating output elements
  - blockDim.x would be 8 in our example
  - Each thread loads one input element into the shared memory

- We will present Design 2 and leave Design 1 as an exercise.

Thread to Input and Output Data Mapping



Thread 0 reads this

Thread 0 writes this

For each thread,
Index_i = index_o – n

were n is Mask_Width /2
n is 2 in this example

All Threads Participate in Loading Input Tiles

```
float output = 0.0f;

if((index_i >= 0) && (index_i < Width)) {
  Ns[tx] = N[index_i];
}
else{
  Ns[tx] = 0.0f;
}
```

Some threads do not participate in calculating output

```
if (threadIdx.x < O_TILE_WIDTH){
   output = 0.0f;
   for(j = 0; j < Mask_Width; j++) {
       output += M[j] * Ns[j+threadIdx.x];
   }
   P[index_o] = output;
}
```

– index_o = blockIdx.x*O_TILE_WIDTH + threadIdx.x

– Only Threads 0 through O_TILE_WIDTH-1 participate in calculation of output.

Setting Block Size

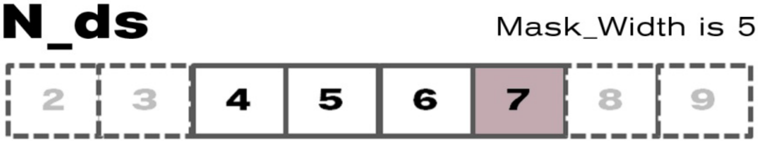
```
#define O_TILE_WIDTH 1020
#define BLOCK_WIDTH (O_TILE_WIDTH + 4)

dim3 dimBlock(BLOCK_WIDTH,1, 1);

dim3 dimGrid((Width-1)/O_TILE_WIDTH+1, 1, 1)
```

The Mask_Width is 5 in this example

In general, block width should be

```
   output tile width + (mask width-1)
```

Shared Memory Data Reuse



N_ds

Mask_Width is 5

2	3	4	5	6	7	8	9

Element 2 is used by thread 4 (1X)

Element 3 is used by threads 4, 5 (2X)

Element 4 is used by threads 4, 5, 6 (3X)

Element 5 is used by threads 4, 5, 6, 7 (4X)

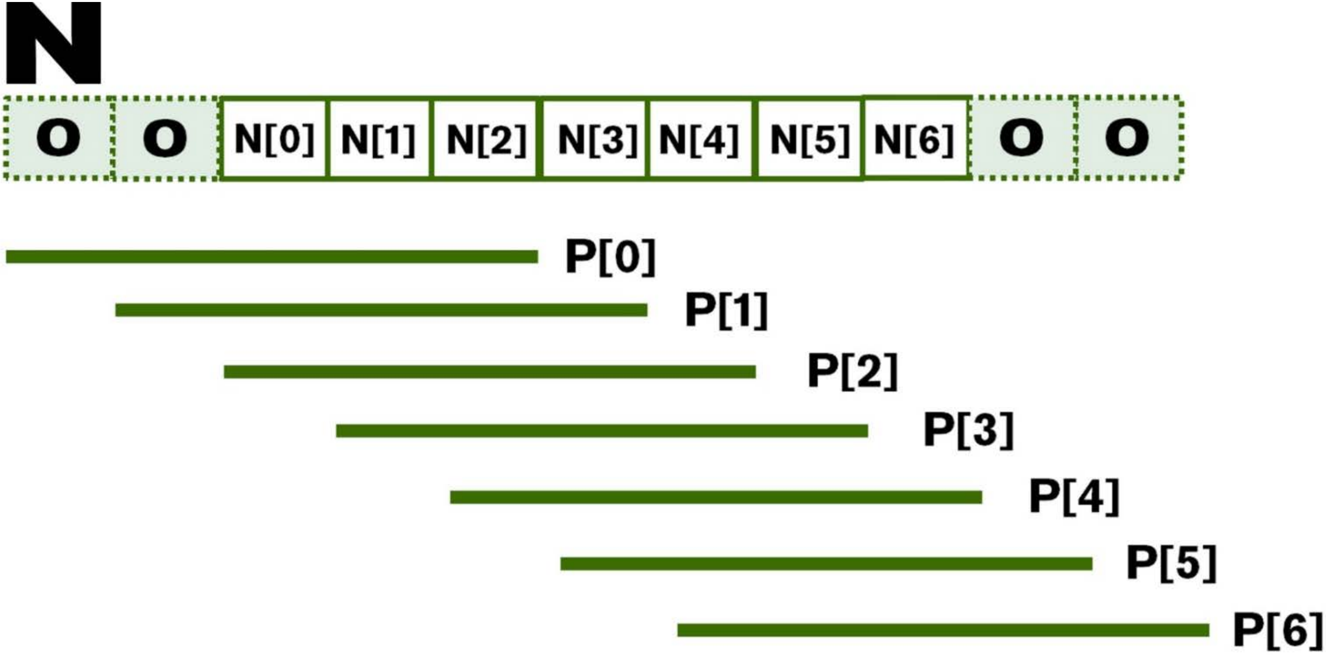
Element 6 is used by threads 4, 5, 6, 7 (4X)

Element 7 is used by threads 5, 6, 7 (3X)

Element 8 is used by threads 6, 7 (2X)

Element 9 is used by thread 7 (1X)

Ghost Cells



NVIDIA

GPU Teaching Kit

Accelerated Computing

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN