GPU Teaching Kit

Accelerated Computing

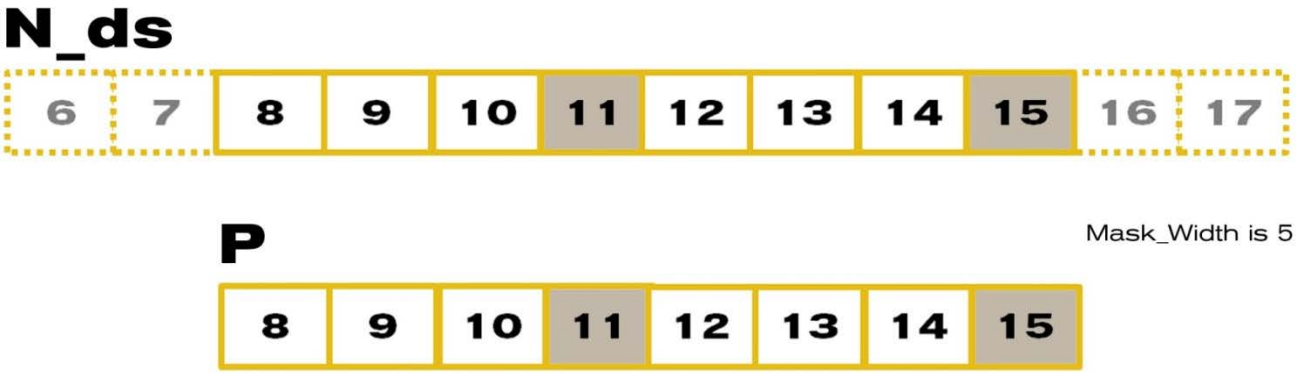
Module 8.4 – Parallel Computation Patterns (Stencil)

Analyzing Data Reuse in Tiled Convolution

Objective

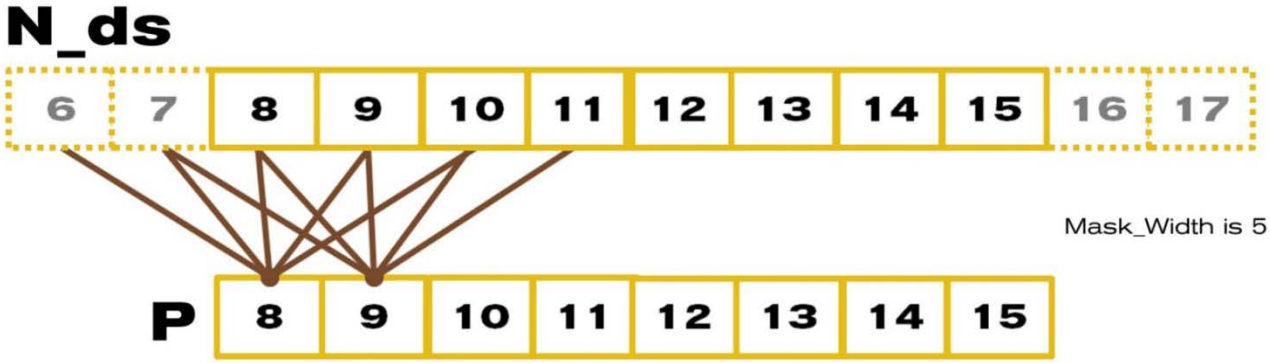
- To learn to analyze the cost and benefit of tiled parallel convolution algorithms
  - More complex reuse pattern than matrix multiplication
  - Less uniform access patterns

An 8-element Convolution Tile



For Mask_Width=5, we load 8+5-1=12 elements (12 memory loads)

Each output P element uses 5 N elements



P[8] uses N[6], N[7], N[8], N[9], N[10]
P[9] uses N[7], N[8], N[9], N[10], N[11]
P[10] use N[8], N[9], N[10], N[11], N[12]
…
P[14] uses N[12], N[13], N[14], N[15], N[16]
P[15] uses N[13], N[14], N[15], N[16], N[17]

A simple way to calculate tiling benefit

- (8+5-1)=12 elements loaded
- 8*5 global memory accesses replaced by shared memory accesses
- This gives a bandwidth reduction of 40/12=3.3

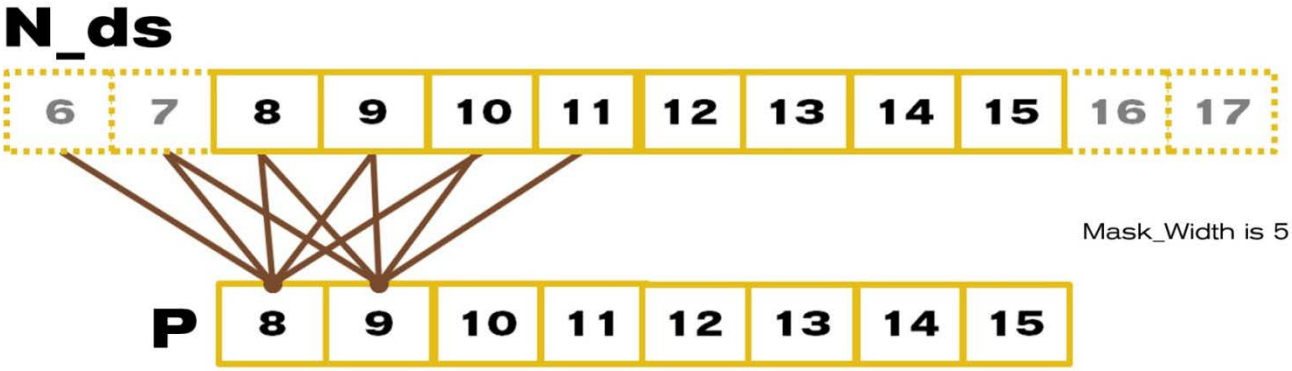
In General, for 1D TILED CONVOLUTION

- `O_TILE_WIDTH+MASK_WIDTH -1` elements loaded for each input tile
- `O_TILE_WIDTH*MASK_WIDTH` global memory accesses replaced by shared memory accesses
- This gives a reduction factor of

```
(O_TILE_WIDTH*MASK_WIDTH)/(O_TILE_WIDTH+MASK_WIDTH-1)
```

This ignores ghost elements in edge tiles.

Another Way to Look at Reuse



N[6] is used by P[8] (1X)
N[7] is used by P[8], P[9] (2X)
N[8] is used by P[8], P[9], P[10] (3X)
N[9] is used by P[8], P[9], P[10], P[11] (4X)
N10 is used by P[8], P[9], P[10], P[11], P[12] (5X)
… (5X)
N[14] is used by P[12], P[13], P[14], P[15] (4X)
N[15] is used by P[13], P[14], P[15] (3X)

Another Way to Look at Reuse

The total number of global memory accesses (to the (8+5-1)=12 N elements) replaced by shared memory accesses is:

$$1 + 2 + 3 + 4 + 5 * (8-5+1) + 4 + 3 + 2 + 1$$
$$= 10 + 20 + 10$$
$$= 40$$

So the reduction is:

$$40/12 = 3.3$$

In General, for 1D

– The total number of global memory accesses to the input tile can be calculated as

$$1 + 2 + \ldots + \text{MASK\_WIDTH}-1 + \text{MASK\_WIDTH} * (\text{O\_TILE\_WIDTH} - \text{MASK\_WIDTH} + 1) + \text{MASK\_WIDTH}-1 + \ldots + 2 + 1$$

$$= \text{MASK\_WIDTH} * (\text{MASK\_WIDTH}-1) + \text{MASK\_WIDTH} * (\text{O\_TILE\_WIDTH} - \text{MASK\_WIDTH} + 1)$$

$$= \text{MASK\_WIDTH} * \text{O\_TILE\_WIDTH}$$

For a total of O_TILE_WIDTH + MASK_WIDTH -1 input tile elements

Examples of Bandwidth Reduction for 1D

The reduction ratio is:

MASK_WIDTH * (O_TILE_WIDTH)/(O_TILE_WIDTH+MASK_WIDTH-1)

O_TILE_WIDTH	16	32	64	128	256
MASK_WIDTH= 5	4.0	4.4	4.7	4.9	4.9
MASK_WIDTH = 9	6.0	7.2	8.0	8.5	8.7

For 2D Convolution Tiles

- $(O\_TILE\_WIDTH+MASK\_WIDTH-1)^2$ input elements need to be loaded into shared memory
- The calculation of each output element needs to access $MASK\_WIDTH^2$ input elements
- $O\_TILE\_WIDTH^2 * MASK\_WIDTH^2$ global memory accesses are converted into shared memory accesses
- The reduction ratio is

$$O\_TILE\_WIDTH^2 * MASK\_WIDTH^2 / (O\_TILE\_WIDTH+MASK\_WIDTH-1)^2$$

Bandwidth Reduction for 2D

The reduction ratio is:

$O\_TILE\_WIDTH^2 * MASK\_WIDTH^2 / (O\_TILE\_WIDTH+MASK\_WIDTH-1)^2$

O_TILE_WIDTH	8	16	32	64
MASK_WIDTH = 5	11.1	16	19.7	22.1
MASK_WIDTH = 9	20.3	36	51.8	64

Tile size has significant effect on of the memory bandwidth reduction ratio.

This often argues for larger shared memory size.

GPU Teaching Kit

Accelerated Computing