UNIVERSITÀ DEGLI STUDI FIRENZE

Parallel Computing

Prof. Marco Bertini

Data parallelism: GPU computing

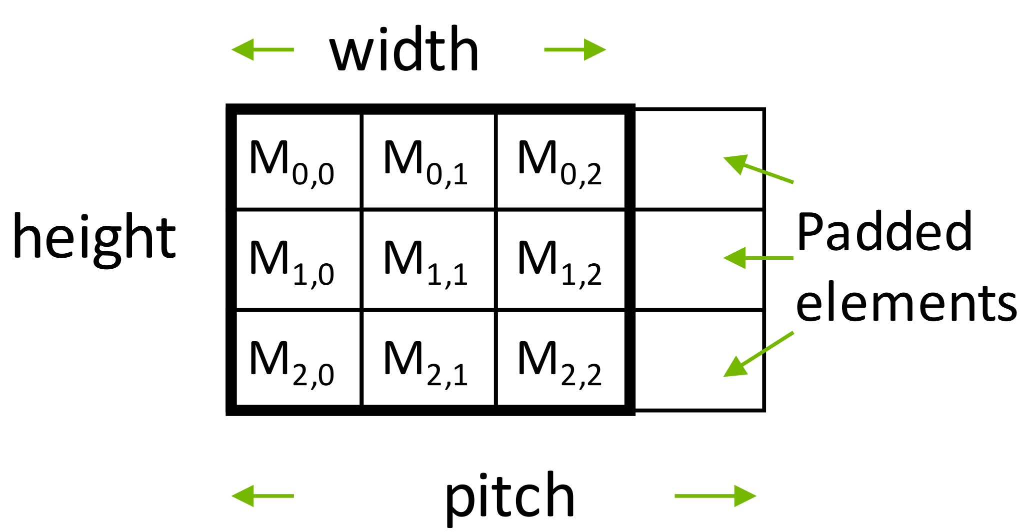
UNIVERSITÀ DEGLI STUDI FIRENZE

2D convolution: tile boundaries

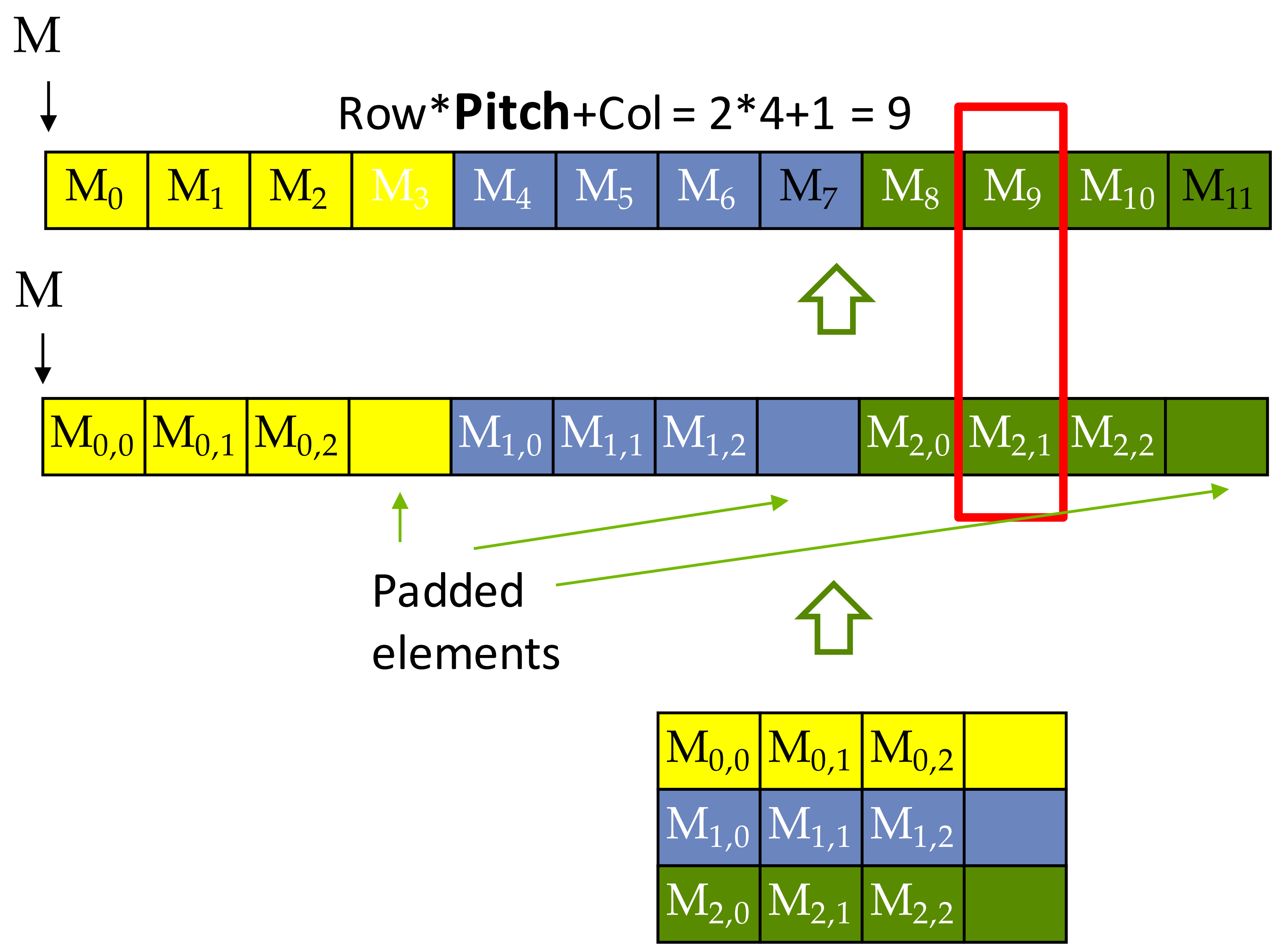
2D Image Matrix with Automated Padding

- It is sometimes desirable to pad each row of a 2D matrix to multiples of DRAM bursts
  - So each row starts at the DRAM burst boundary
  - Effectively adding columns
  - This is usually done automatically by matrix allocation function
  - Pitch can be different for different hardware
- Example: a 3X3 matrix padded into a 3X4 matrix

Height is 3
Width is 3
Channels is 1 (e.g. gray level image)
Pitch is 4



Row-Major Layout with Pitch



Sample image struct

```
// Image Matrix Structure declaration
//
typedef struct {
   int width;
   int height;
   int pitch;
   int channels;
   float* data;
} Image_t;
```

Setting Block Size

```
#define O_TILE_WIDTH 12

#define BLOCK_WIDTH (O_TILE_WIDTH + 4)



dim3 dimBlock(BLOCK_WIDTH,BLOCK_WIDTH);

dim3 dimGrid((Image_Width-1)/O_TILE_WIDTH+1,
(Image_Height-1)/O_TILE_WIDTH+1, 1)
```

- In general, BLOCK_WIDTH should be
- `O_TILE_WIDTH + (MASK_WIDTH-1)`

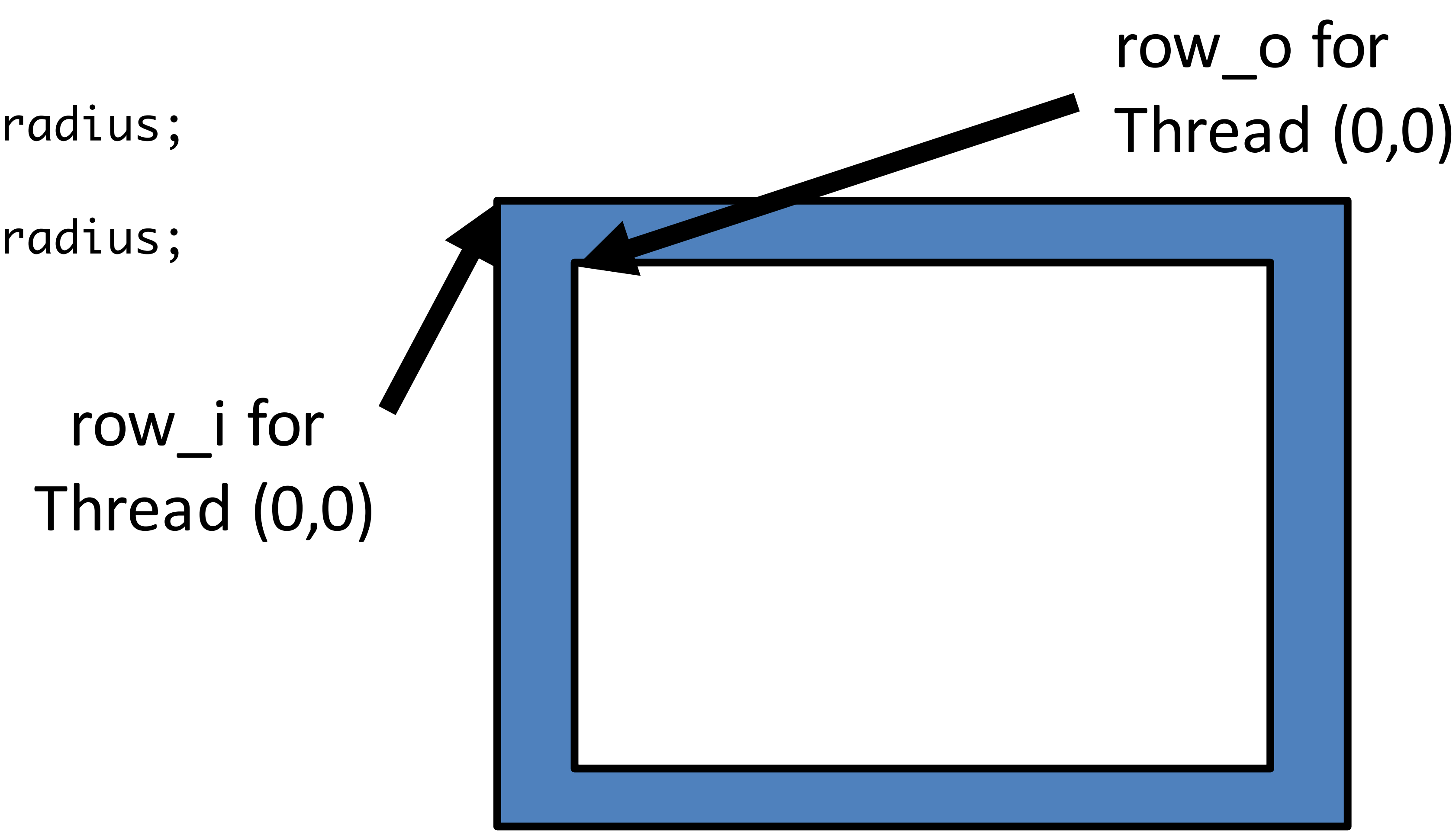
Using constant memory and caching for Mask

- Mask is used by all threads but not modified in the convolution kernel
  - All threads in a warp access the same locations at each point in time
- CUDA devices provide constant memory whose contents are aggressively cached
  - Cached values are broadcast to all threads in a warp
  - Effectively magnifies memory bandwidth without consuming shared memory
- Use of `const __restrict__` qualifiers for the mask parameter informs the compiler that it is eligible for constant caching, for example:

```
__global__ void convolution_2D_kernel(float *P, float
*N, int height, int width, int channels,
const float __restrict__ *M);
```

Shifting from output coordinates to input coordinate

```
int tx = threadIdx.x;

int ty = threadIdx.y;

int row_o = blockIdx.y*O_TILE_WIDTH + ty;

int col_o = blockIdx.x*O_TILE_WIDTH + tx;


int row_i = row_o - mask_radius;

int col_i = col_o - mask_radius;
```



Taking Care of Boundaries (1 channel example)

```
if((row_i >= 0) && (row_i < height) &&
     (col_i >= 0)  && (col_i < width)) {
    Ns[ty][tx] = data[row_i * width + col_i];
} else{
    Ns[ty][tx] = 0.0f;
}
```

- Use of width here is OK if pitch is set to width (no padding)

Calculating output

Some threads do not participate in calculating output

```
float output = 0.0f;

if(ty < O_TILE_WIDTH && tx < O_TILE_WIDTH){

     for(i = 0; i < MASK_WIDTH; i++) {

       for(j = 0; j < MASK_WIDTH; j++) {

         output += M[i][j] * Ns[i+ty][j+tx];

     }

}
```

Writing output

- Some threads do not write output (1 channel example)

```
if(row_o < height && col_o < width)
        data[row_o*width + col_o] =
            output;
```

Credits

- These slides report material from:
  - NVIDIA GPU Teaching Kit

Books

- Programming Massively Parallel Processors: A Hands-on Approach, D. B. Kirk and W-M. W. Hwu, Morgan Kaufman - Chapt. 8