GPU Teaching Kit

Accelerated Computing

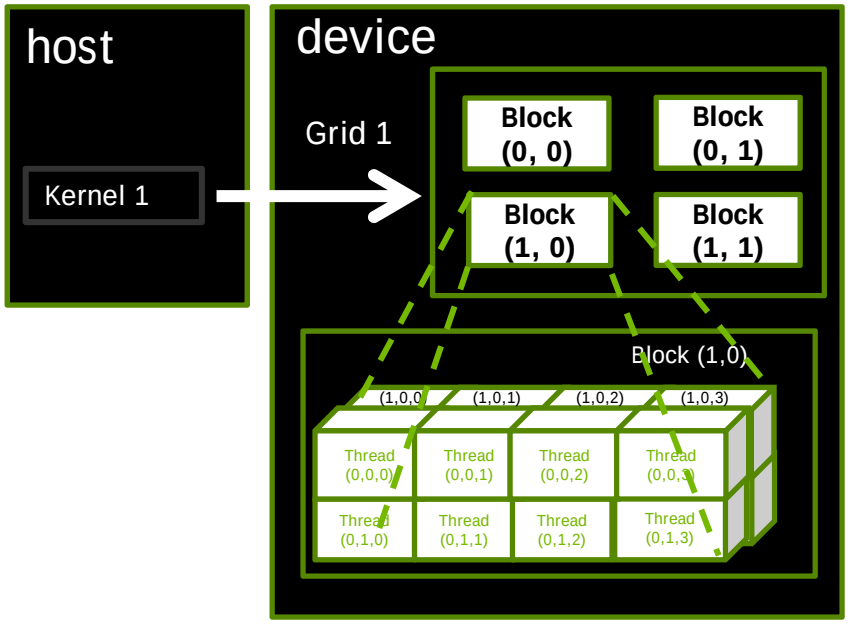
# Lecture 3.2 – CUDA Parallelism Model

## Multidimensional Kernel Configuration

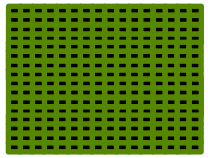
# Objective

- To understand multidimensional Grids
  - Multi-dimensional block and thread indices
  - Mapping block/thread indices to data indices

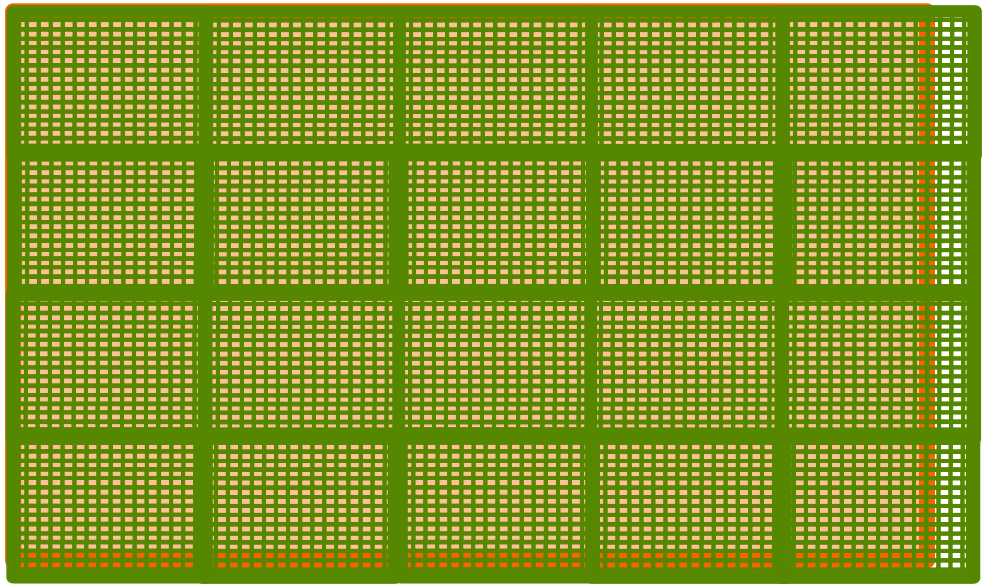
# A Multi-Dimensional Grid Example



# Processing a Picture with a 2D Grid

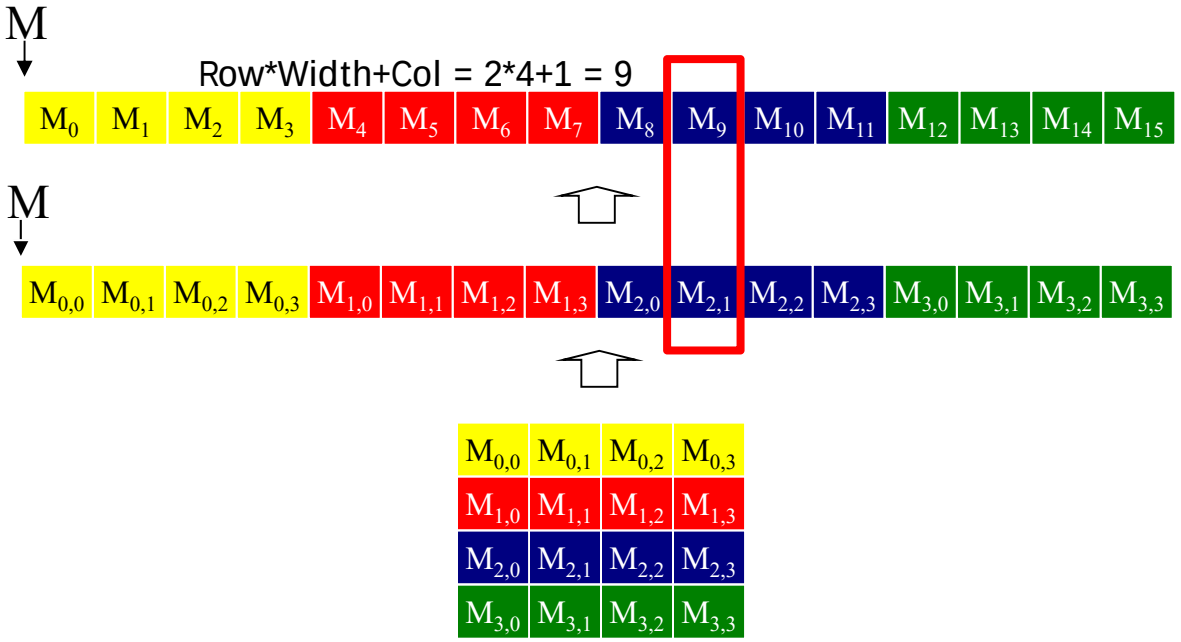


16×16 blocks



62×76 picture

# Row-Major Layout in C/C++



# Source Code of a PictureKernel

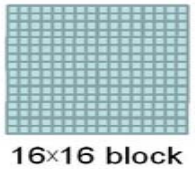
```
__global__ void PictureKernel(float* d_Pin, float* d_Pout,
                              int height, int width)
{

  // Calculate the row # of the d_Pin and d_Pout element
  int Row = blockIdx.y*blockDim.y + threadIdx.y;

  // Calculate the column # of the d_Pin and d_Pout element
  int Col = blockIdx.x*blockDim.x + threadIdx.x;

  // each thread computes one element of d_Pout if in range
  if ((Row < height) && (Col < width)) {
    d_Pout[Row*width+Col] = 2.0*d_Pin[Row*width+Col];
  }
}
```

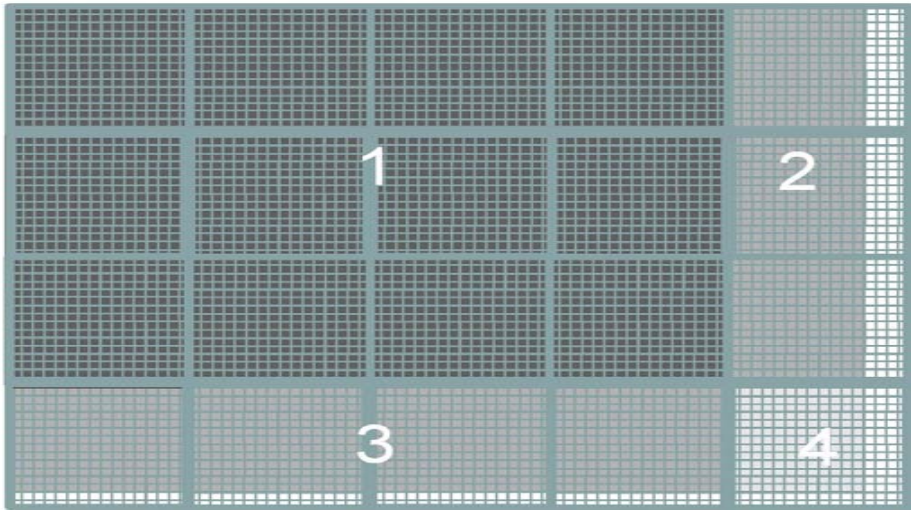
Scale every pixel value by 2.0

# Host Code for Launching PictureKernel

```
// assume that the picture is m × n,
// m pixels in y dimension and n pixels in x dimension
// input d_Pin has been allocated on and copied to device
// output d_Pout has been allocated on device
...
dim3 DimGrid((n-1)/16 + 1, (m-1)/16+1, 1);
dim3 DimBlock(16, 16, 1);
PictureKernel<<<DimGrid,DimBlock>>>(d_Pin, d_Pout, m, n);
...
```

# Covering a 62×76 Picture with 16×16 Blocks




Not all threads in a Block will follow the same control flow path.

GPU Teaching Kit

Accelerated Computing