GPU Teaching Kit

Accelerated Computing

Module 3.1 - CUDA Parallelism Model

Kernel-Based SPMD Parallel Programming

Objective

– To learn the basic concepts involved in a simple CUDA kernel function
  – Declaration
  – Built-in variables
  – Thread index to data index mapping

Example: Vector Addition Kernel

Device Code

```
// Compute vector sum C = A + B
// Each thread performs one pair-wise addition

__global__
void vecAddKernel(float* A, float* B, float* C, int n)
{
    int i = threadIdx.x+blockDim.x*blockIdx.x;
    if(i<n) C[i] = A[i] + B[i];
}
```

Example: Vector Addition Kernel Launch (Host Code)

Host Code

```
void vecAdd(float* h_A, float* h_B, float* h_C, int n)
{
 // d_A, d_B, d_C allocations and copies omitted
 // Run ceil(n/256.0) blocks of 256 threads each
  vecAddKernel<<<ceil(n/256.0),256>>>(d_A, d_B, d_C, n);
}
```

The ceiling function makes sure that there are enough threads to cover all elements.

More on Kernel Launch (Host Code)

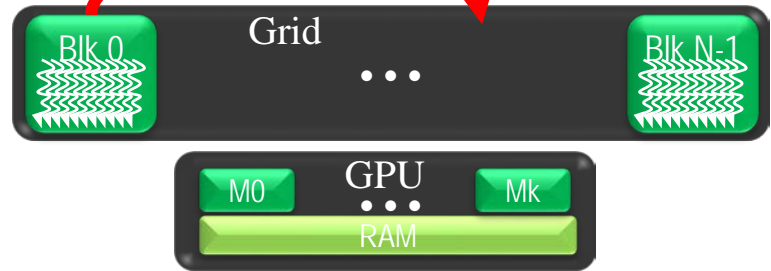Host Code

```
void vecAdd(float* h_A, float* h_B, float* h_C, int n)
{
  dim3 DimGrid((n-1)/256 + 1, 1, 1);
  dim3 DimBlock(256, 1, 1);
  vecAddKernel<<<DimGrid,DimBlock>>>(d_A, d_B, d_C, n);
}
```

This is an equivalent way to express the ceiling function.

Kernel execution in a nutshell

```
__host__
void vecAdd(…)
{
  dim3 DimGrid(ceil(n/256.0),1,1);
  dim3 DimBlock(256,1,1);
  vecAddKernel<<<DimGrid,DimBlock>>>(d_A,d_B
,d_C,n);
}
```

```
__global__
void vecAddKernel(float *A,
    float *B, float *C, int n)
{
  int i = blockIdx.x * blockDim.x
      + threadIdx.x;

  if( i<n ) C[i] = A[i]+B[i];
}
```
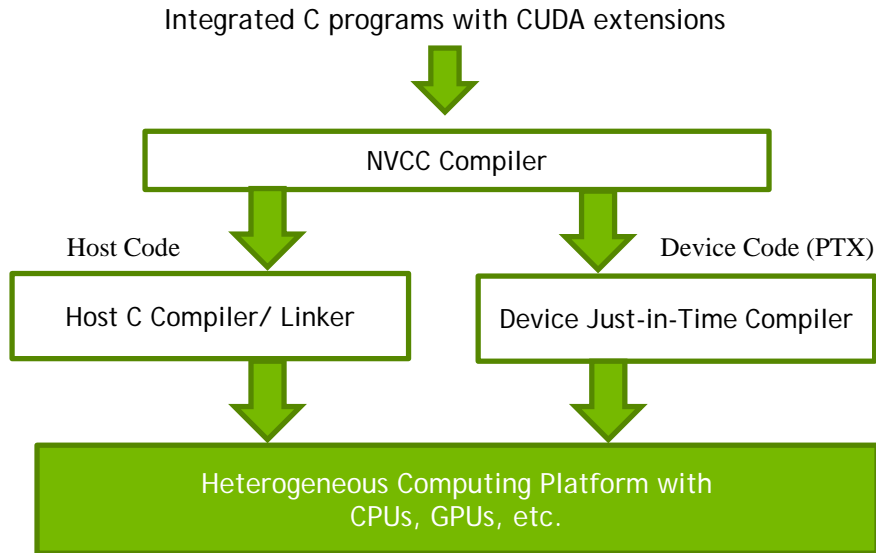
Grid

Blk 0   · · ·   Blk N-1

GPU

M0   · · ·   Mk

RAM

More on CUDA Function Declarations

	Executed on the:	Only callable from the:
`__device__ float DeviceFunc()`	device	device
`__global__ void KernelFunc()`	device	host
`__host__ float HostFunc()`	host	host

- **`__global__`** defines a kernel function
  - Each "__" consists of two underscore characters
  - A kernel function must return **`void`**
- **`__device__`** and **`__host__`** can be used together
- **`__host__`** is optional if used alone

Compiling A CUDA Program

Integrated C programs with CUDA extensions

NVCC Compiler

Host Code

Device Code (PTX)

Host C Compiler/ Linker

Device Just-in-Time Compiler

Heterogeneous Computing Platform with CPUs, GPUs, etc.

GPU Teaching Kit

Accelerated Computing