

CPSC/ECE 4780/6780

General-Purpose Computation on Graphical Processing Units (GPGPU)

Lecture 3: Introduction to CUDA

Recaps from Last Lecture

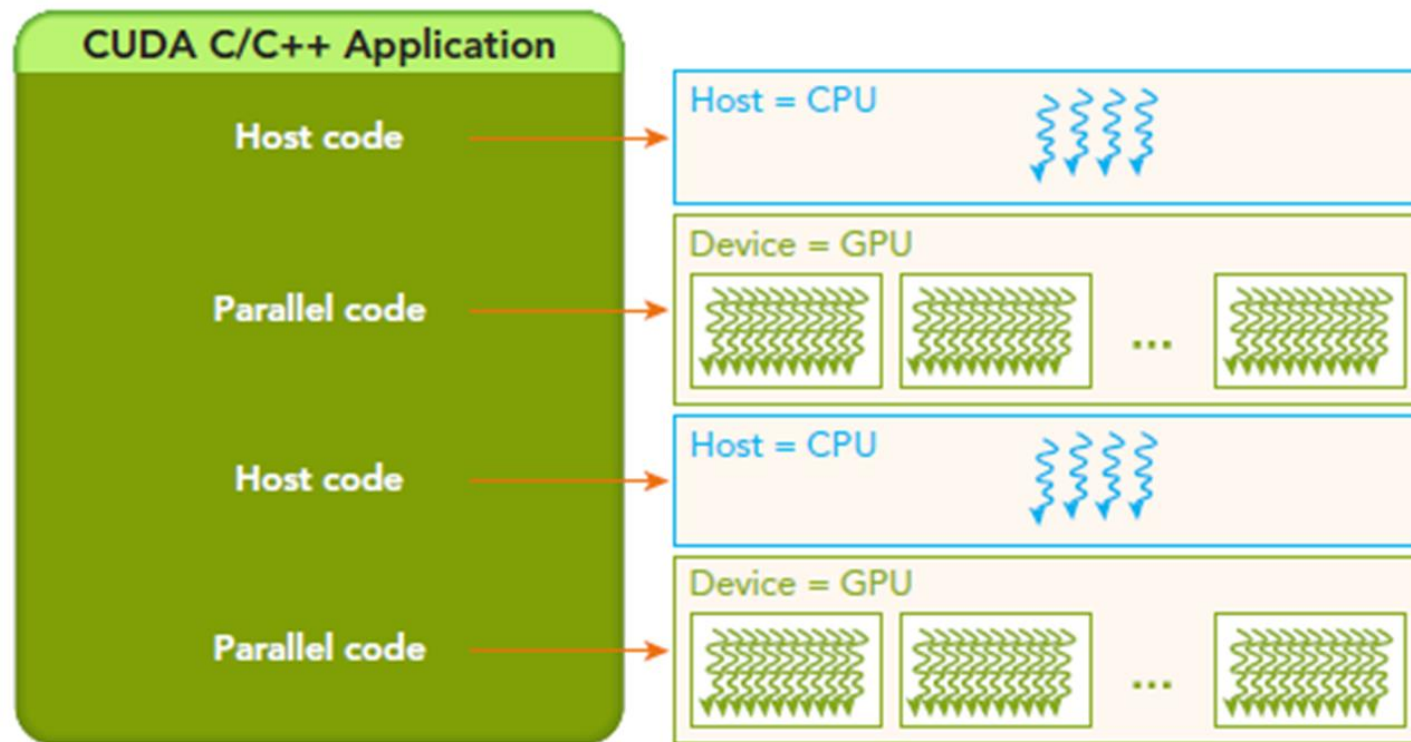
- What is GPU?
- History of GPUs
- Architecture of GPU
- CPU/GPU comparisons
- Why should we use GPUs?
- CPU+GPU acceleration
- GPGPU programming

What is CUDA?

- CUDA – “Compute Unified Device Architecture”
- General-purpose parallel computing platform and programming model
- Created by NVIDIA first in 2007
- Written mostly like C

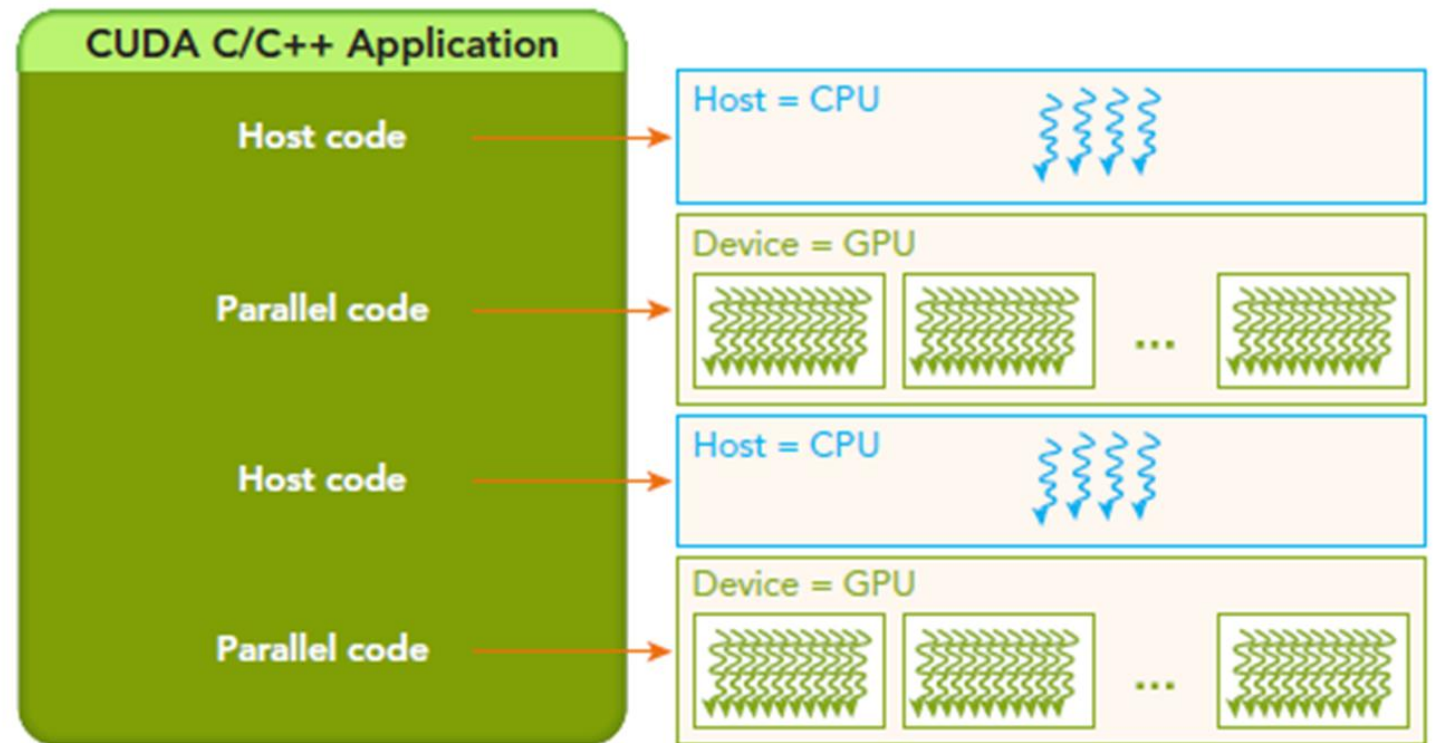
CUDA Programming Structure

- Integrated host + device application C program
 - Host – CPU and its memory
 - Serial or modestly parallel parts
 - Written in ANSI C
 - Device – GPU and its memory
 - Highly parallel parts
 - Written in CUDA C
 - “Kernel”



Processing Flow of a CUDA Program

- Copy input data from CPU memory to GPU memory
- Invoke kernels to operate on the data stored in GPU memory
- Copy data back from GPU memory to CPU memory



Memory Management and Data Transfer

- Host and device memory are separate entities
 - Host pointers point to CPU memory
 - May be passed to/from device code
 - May not be dereferenced in device code
 - Device pointers point to GPU memory
 - May be passed to/from host code
 - May not be dereferenced in host code

STANDARD C FUNCTIONS	CUDA C FUNCTIONS
<code>malloc</code>	<code>cudaMalloc</code>
<code>memcpy</code>	<code>cudaMemcpy</code>
<code>memset</code>	<code>cudaMemset</code>
<code>free</code>	<code>cudaFree</code>

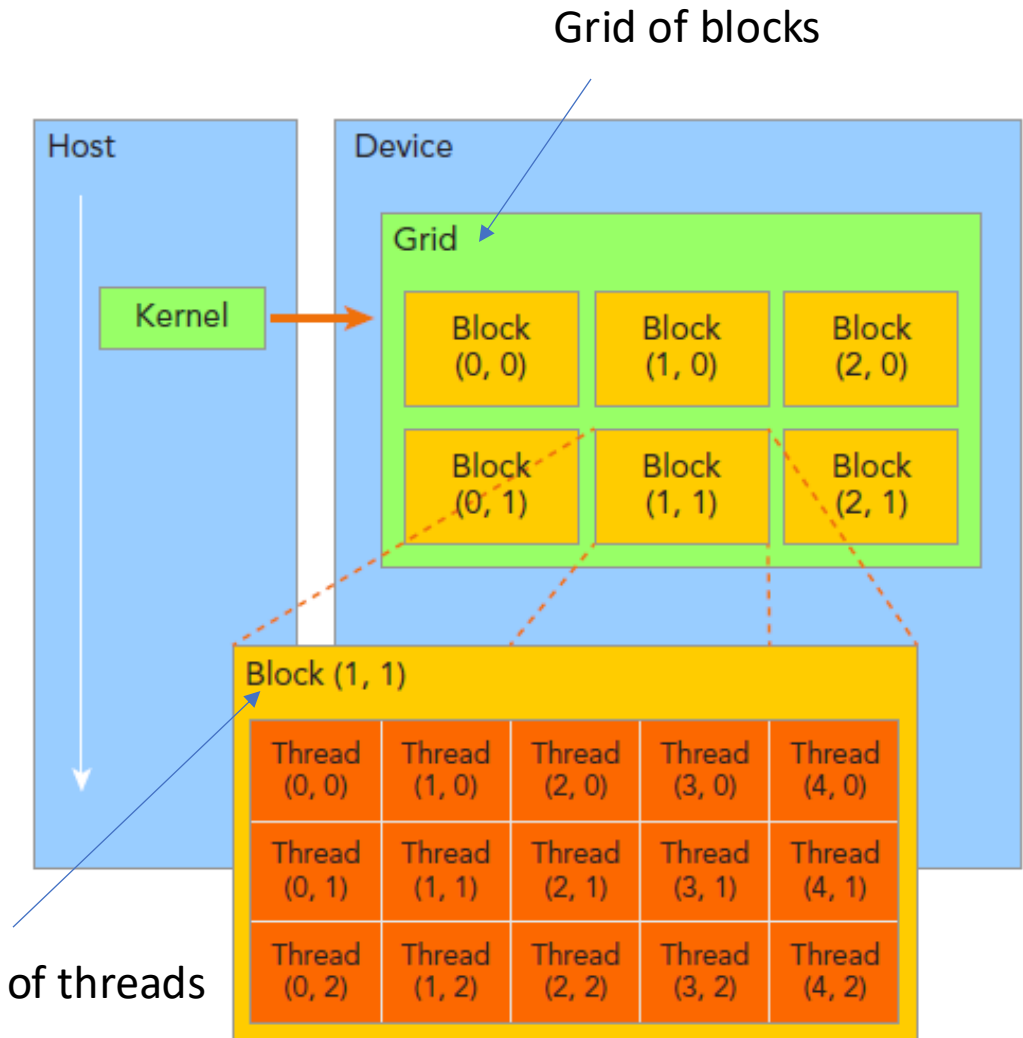
Host and device memory functions

CUDA Function Declaration

- CUDA extensions to C functional declaration
 - `__global__`: indicates a CUDA kernel function
 - executed on the device
 - Only callable from the host
 - Must have a void return type
 - `__device__`: indicates a CUDA device function
 - Executed on the device
 - Only callable from the device
 - `__host__`: indicates a CUDA host function
 - Executed on the host
 - Only callable from the host

Organizing Threads

- Two-level thread hierarchy
 - Grids of blocks
 - Blocks of threads
- All threads in a grid share the same global memory space
- A thread block is a group of threads that can cooperate with each other by:
 - Block-local synchronization
 - Block-local shared memory
- Threads coordinates:
 - blockIdx (block index within a grid)
 - threadIdx (thread index within a block)
 - Type: uint3 (.x, .y, .z)

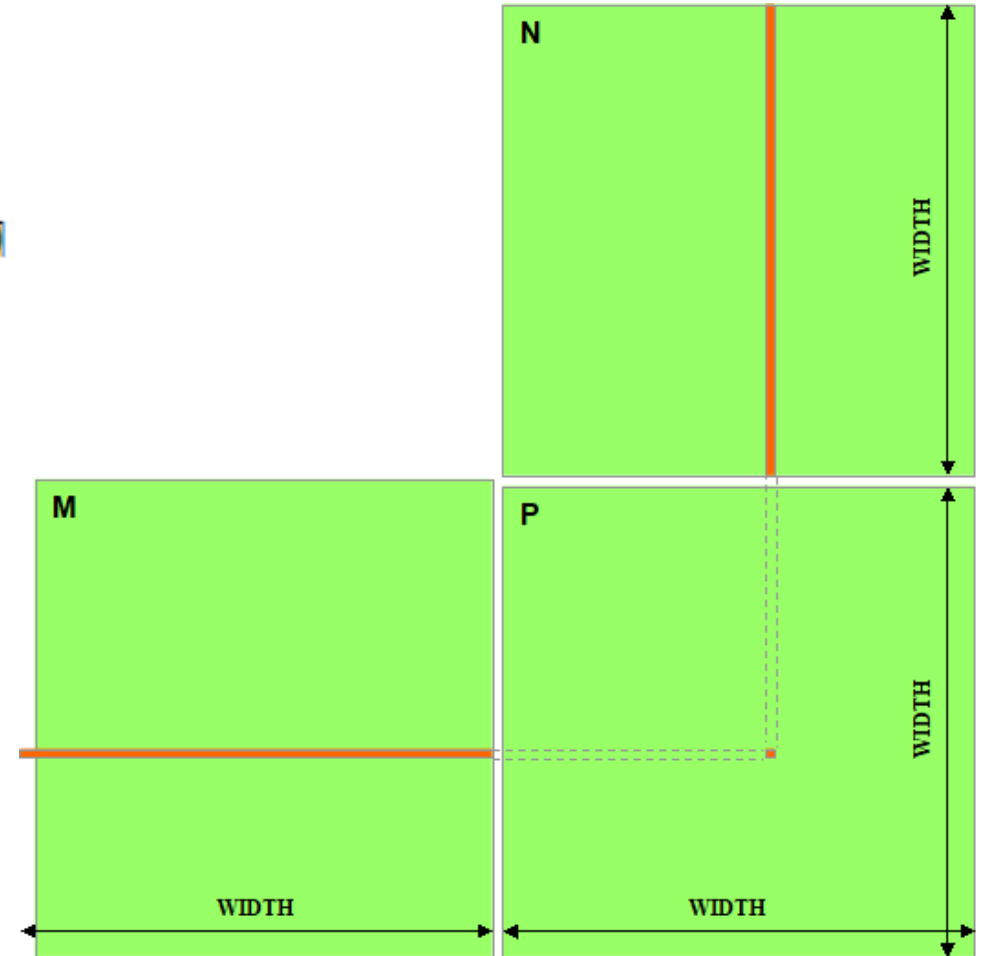


A thread hierarchy structure with a 2D grid containing 2D blocks

Matrix Multiplication on CPU

- $M * N \Rightarrow P$

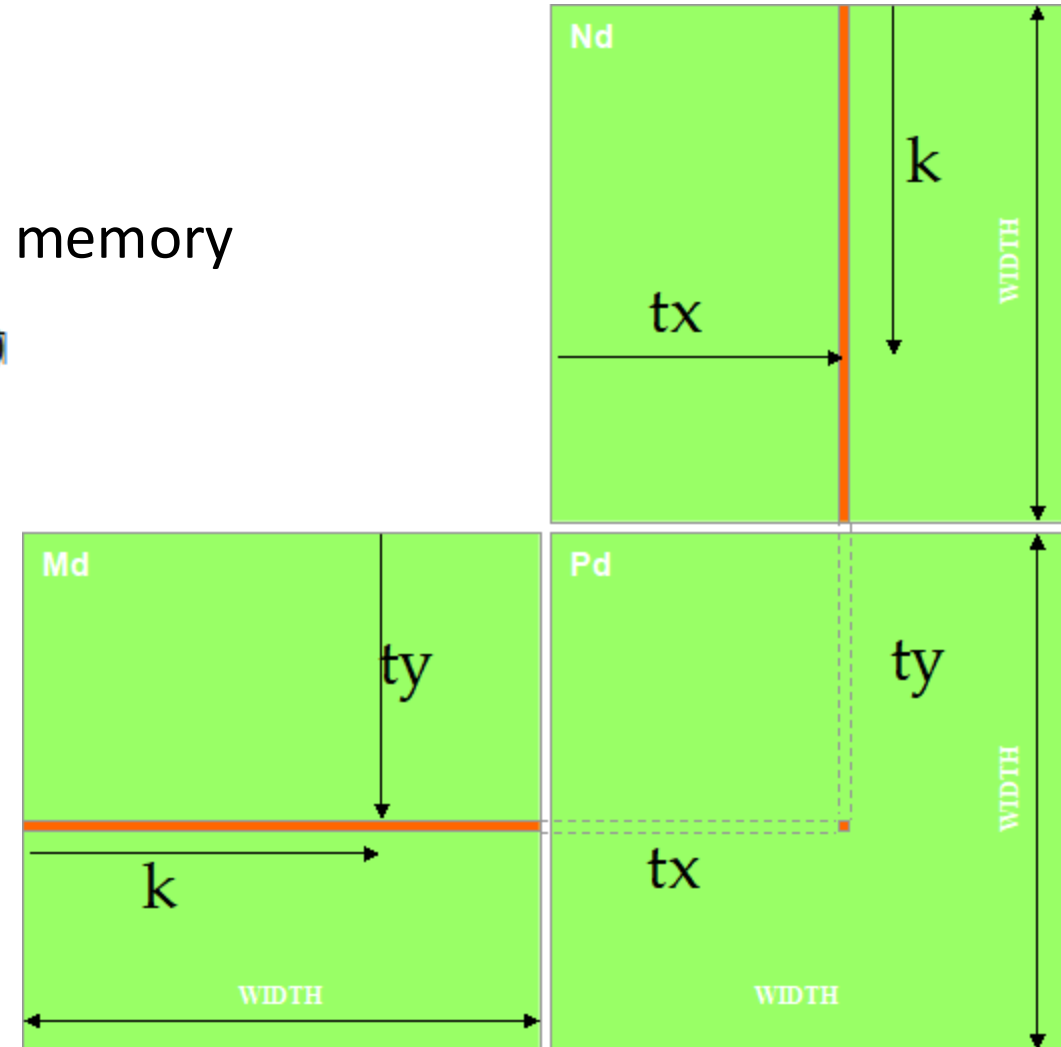
```
void MatrixMulOnHost(float* M, float* N, float* P, int Width)
{
    for (int i = 0; i < Width; ++i)
        for (int j = 0; j < Width; ++j) {
            double sum = 0;
            for (int k = 0; k < Width; ++k) {
                double a = M[i * width + k];
                double b = N[k * width + j];
                sum += a * b;
            }
            P[i * Width + j] = sum;
        }
}
```



Matrix Multiplication on GPU

- One thread calculates one element of P
- M and N are loaded width times from global memory

```
__global__ void MatrixMulKernel(float* Md, float* Nd, float* Pd, int Width)
{
    int tx = threadIdx.x;
    int ty = threadIdx.y;
    float Pvalue = 0;
    for (int k = 0; k < Width; ++k) {
        float Melement = Md[threadIdx.y*Width+k];
        float Nelement = Nd[k*Width+threadIdx.x];
        Pvalue += Melement * Nelement;
    }
    Pd[threadIdx.y*Width+threadIdx.x] = Pvalue;
}
```



CUDA Device Properties

- Get the count of CUDA devices
 - `int count;`
 - `cudaGetDeviceCount(&count);`
- Query relevant information of a device
 - `cudaDeviceProp prop;`
 - `cudaGetDeviceProperties(&prop, i);`
- Set device property and choose a proper device
 - `int dev;`
 - `cudaDeviceProp prop;`
 - `prop.major = 1;`
 - `prop.minor = 3`
 - `cudaChooseDevice(&dev, &prop);`
 - `cudaSetDevice(dev);`

Coding Examples

- Coding

- First CUDA program: Hello World
- Add two numbers
- Add two vectors
 - By blockIdx
 - By threadIdx
 - Combined
- Query device property

- Compilation: use nvcc

Option 1: Running your code on Palmetto Cluster

Palmetto is comprised of 1786 compute nodes (totaling 34916 CPU cores), and features:

- 1786 compute nodes, totaling 34916 cores
- Over 850 nodes are equipped with 2x NVIDIA Tesla GPUs (K20, K40, P100, V100 and A100)

Connecting via SSH

- `ssh username@slogin.palmetto.clemson.edu`

Or via Open OnDemand (a browser-based GUI)

- ondemand.rcd.clemson.edu

Load CUDA module:

- `module load cuda gcc`

Compile .cu code with “nvcc”, e.g.,

- `nvcc helloworld.cu`
- `nvcc -o helloworld helloworld.cu`

```
cat /etc/hardware-table
```

<https://docs.rcd.clemson.edu/palmetto/connect/ssh/>

Or Option 2: Running your code on School of Computing machines

`ssh username@titan[1..6].computing.clemson.edu`

Or through Virtual Desktop (recommended for programs with graphical output):

<https://virtual.computing.clemson.edu>

<https://computing.clemson.edu/help/virtual.html>

Open OnDemand Screenshot

This screenshot shows the configuration page for the Palmetto Desktop application in the Open OnDemand interface. The left sidebar lists 'Interactive Apps' with 'Palmetto Desktop' selected. The main panel displays configuration options for the desktop session.

Palmetto Desktop

This app will launch an interactive desktop on one or more compute nodes. You will have full access to the resources these nodes provide. This is analogous to an interactive batch job.

Partition: interact

Account: cuuser_jin6_seto_trace_pv_project

CPU cores: 1
Sequential programs should only choose a single CPU

Memory (GB): 16
Palmetto nodes usually have a CPU:Memory ratio of 1:4

GPUs: 2

GPU Model: Any

Walltime: 1 hour
Walltime is limited to 12 hours on OnDemand

Feature Constraints:
Enter a comma-delimited list of [features](#)

☐ I would like to receive an email when the session starts

Launch

* The Palmetto Desktop session data for this session can be accessed under the [data root directory](#).

This screenshot shows the confirmation page after a Palmetto Desktop session has been successfully created. A green banner at the top states 'Session was successfully created.' The left sidebar shows 'Palmetto Desktop' selected under 'Interactive Apps'.

Palmetto Desktop (487608) 1 node | 1 core | Running

Host: >_ node0075.palmetto.clemson.edu **Cancel**

Created at: 2024-08-19 11:22:16 EDT

Time Remaining: 58 minutes

Session ID: 37823e3b-8a94-4cb8-b651-14feaf272554

Compression: 0 (low) to 9 (high)

Image Quality: 0 (low) to 9 (high)

Launch Palmetto Desktop **View Only (Share-able Link)**