Lecture 1.5
Introduction to Parallel Programming

Outline

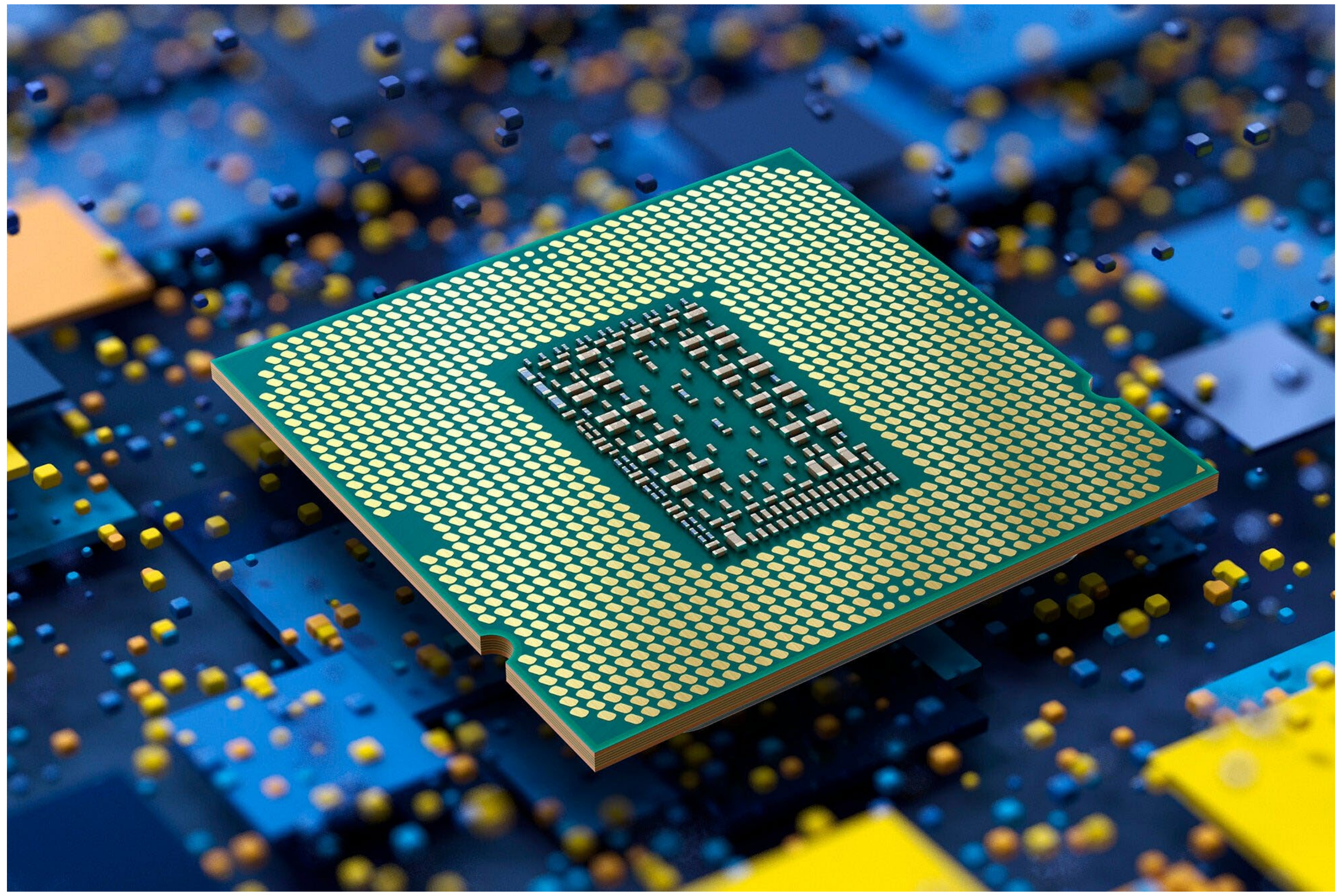
- Introduction to Central processing unit (CPU)
- Introduction to Graphics processing unit (GPU)
- Introduction to Parallel Programming
- Introduction to CUDA
- Introduction to CUDA libraries

Objectives

- Define central processing unit (CPU)
- Explain how CPU works
- Define graphics processing unit (GPU)
- Explain differences of CPU and GPU
- Define parallel programming
- Explain CUDA platform and CUDA libraries

What is CPU?

A “**central processing unit**“ (also called processor) is the brain of the computer system that initiates and executes the instructions of a computer program.

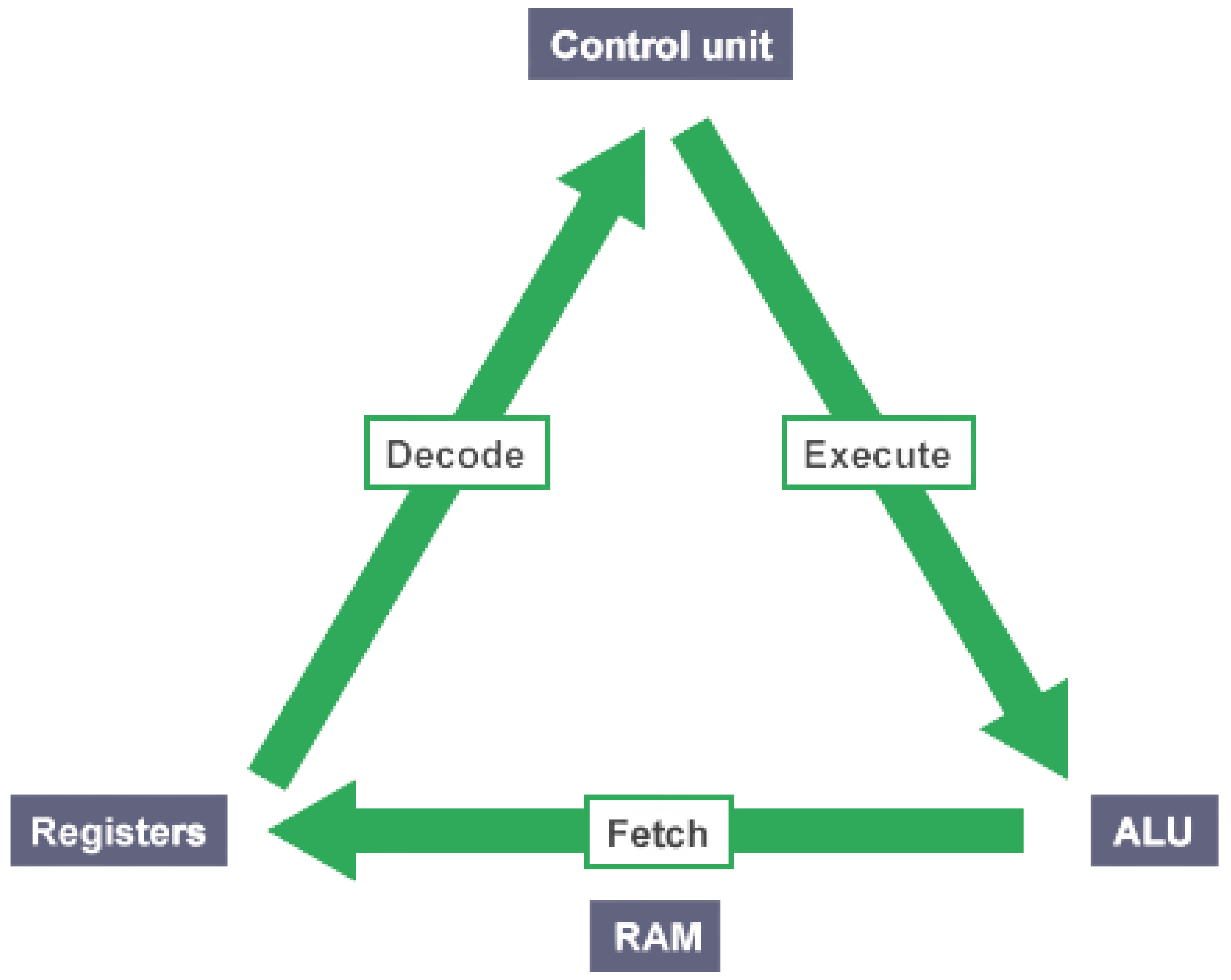


Program execution in CPU

In serial processing, the processor executes the program instructions sequentially, one after the other. After completing that, it executes the next instruction in a sequential manner.

CPU follows the steps below to process a sequence of program instructions that stored in main memory(RAM).

1- Fetch the instruction
2- Decode the instruction
3- Execute the instruction



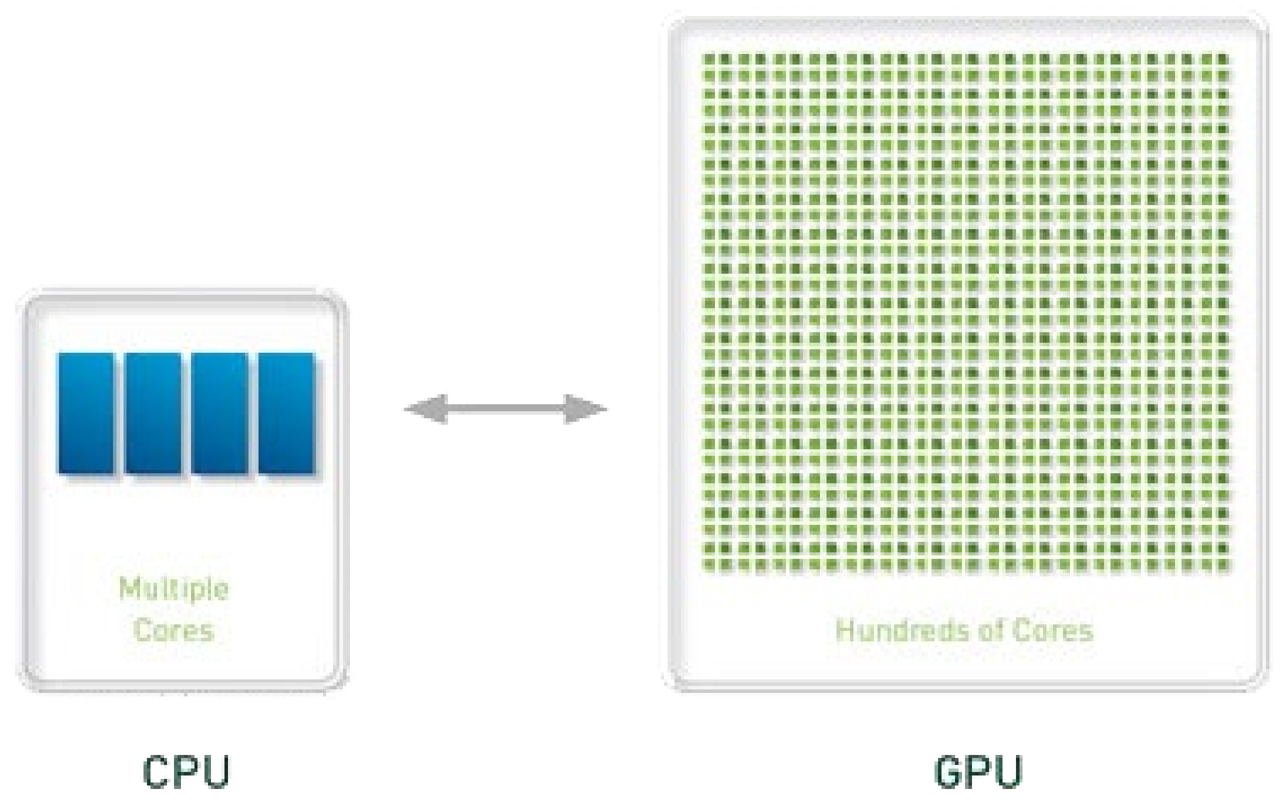
What is GPU?

GPU stands for "**Graphic Processing Unit**" that is a piece of hardware designed to accelerate graphics rendering tasks.

- GPU computing is the use of the GPU as a co-processor to accelerate CPUs for general-purpose scientific and engineering computing.
- It can be used to execute the instructions parallelly on multiple GPU cores, thus accelerating the processing.

Differences CPU and GPU

CPU	GPU
Several cores	Many cores
Designed to minimize latency	Designed to maximize throughput
Good for serial processing	Good for parallel processing
Multiple type of data	Single type of data
Can do a handful of operations at once	Can do thousands of operations at once



What is parallel programming?

Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously.

Parallel programming breaks down the problem into a series of smaller steps, delivers instructions, and processors execute the solutions at the same time.

What is CUDA?

CUDA is parallel architecture of modern GPUs with hundreds of cores. CUDA parallel programming model can be used to program these GPUs.

Benefits of CUDA

- Efficiently process thousands of elements for a particular task in parallel
- Task can communicate and collaborate efficiently

CUDA Kernel

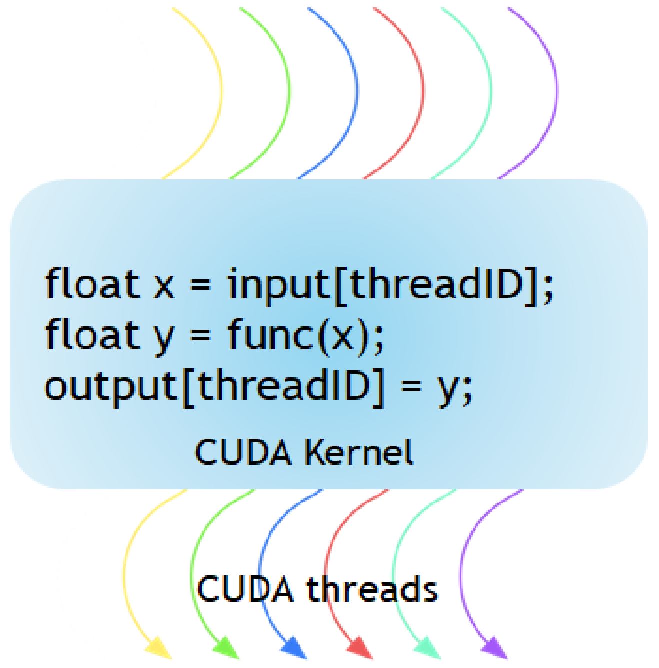
Kernels: CUDA kernels are C functions that get executed in parallel on GPU.

Threads: Kernels execute as a set parallel threads.

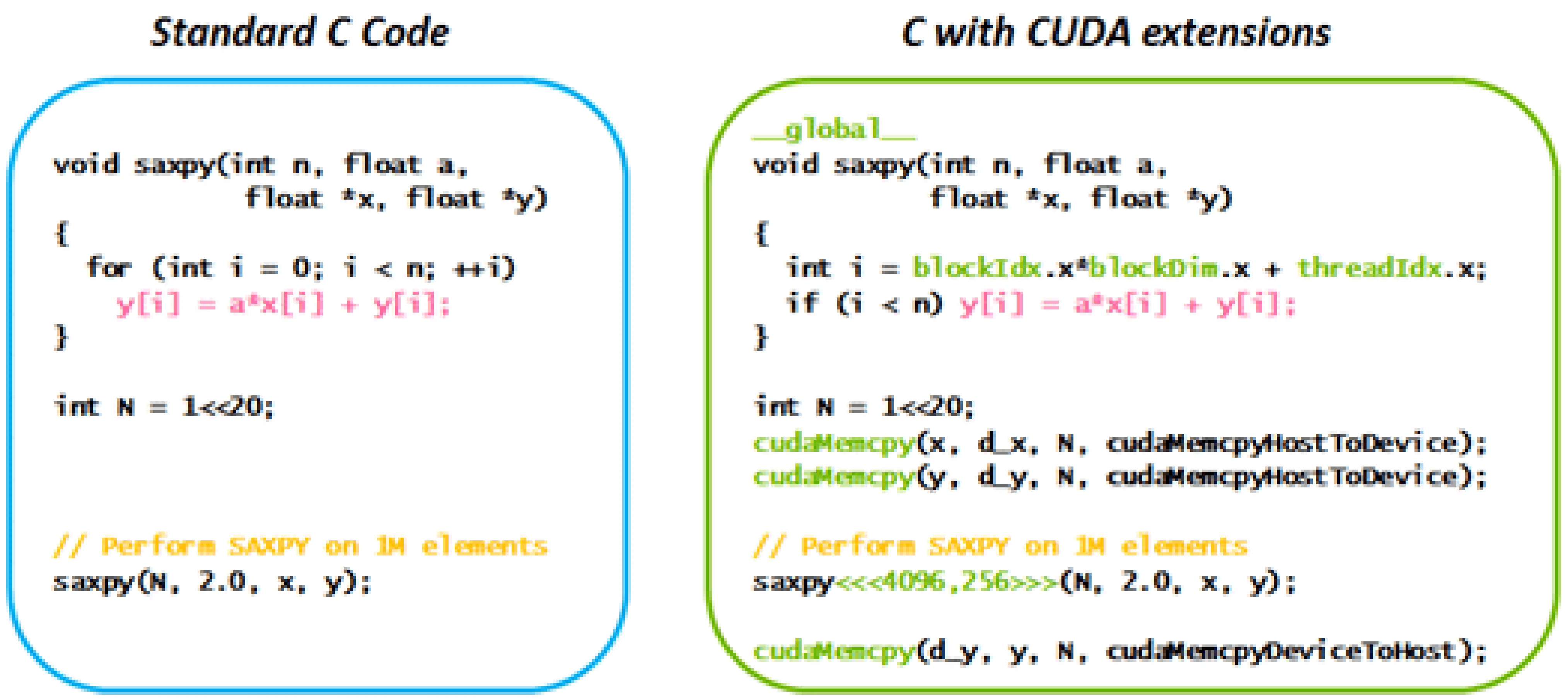
Blocks: A group of threads that can execute tasks in parallel.

Grids: CUDA blocks are grouped into grids with each block having same number of threads.

A CUDA kernel is executed by an array of threads.
- All threads run the same code
- Each thread has an ID that it uses to compute memory addresses and make control decisions


```
float x = input[threadID];
float y = func(x);
output[threadID] = y;
```

A simple example of code is shown below.
It’s written first in plain “C” and then in “C with CUDA extensions.”



Standard C Code

```
void saxpy(int n, float a,
            float *x, float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

int N = 1<<20;



// Perform SAXPY on 1M elements
saxpy(N, 2.0, x, y);
```

C with CUDA extensions

```
__global__
void saxpy(int n, float a,
            float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

int N = 1<<20;
cudaMemcpy(x, d_x, N, cudaMemcpyHostToDevice);
cudaMemcpy(y, d_y, N, cudaMemcpyHostToDevice);

// Perform SAXPY on 1M elements
saxpy<<<4096,256>>>(N, 2.0, x, y);

cudaMemcpy(d_y, y, N, cudaMemcpyDeviceToHost);
```

Keywords in CUDA programming model: host and device

The CUDA programming model is a heterogeneous parallel programming model in which both the CPU and GPU are used. CUDA is C language with set of extensions that allows for the host and device to work together for NVIDIA GPUs.

In CUDA,

Host: The host refers the CPU and its memory.

Device: The device refers GPU and its memory.

Code run on the host can manage memory on both the host and device, and also launches kernels which are functions executed on the device. These kernels are executed by many GPU threads in parallel.

Memory management on CUDA

Memory management refers allocating memory space and transferring data between host and device.

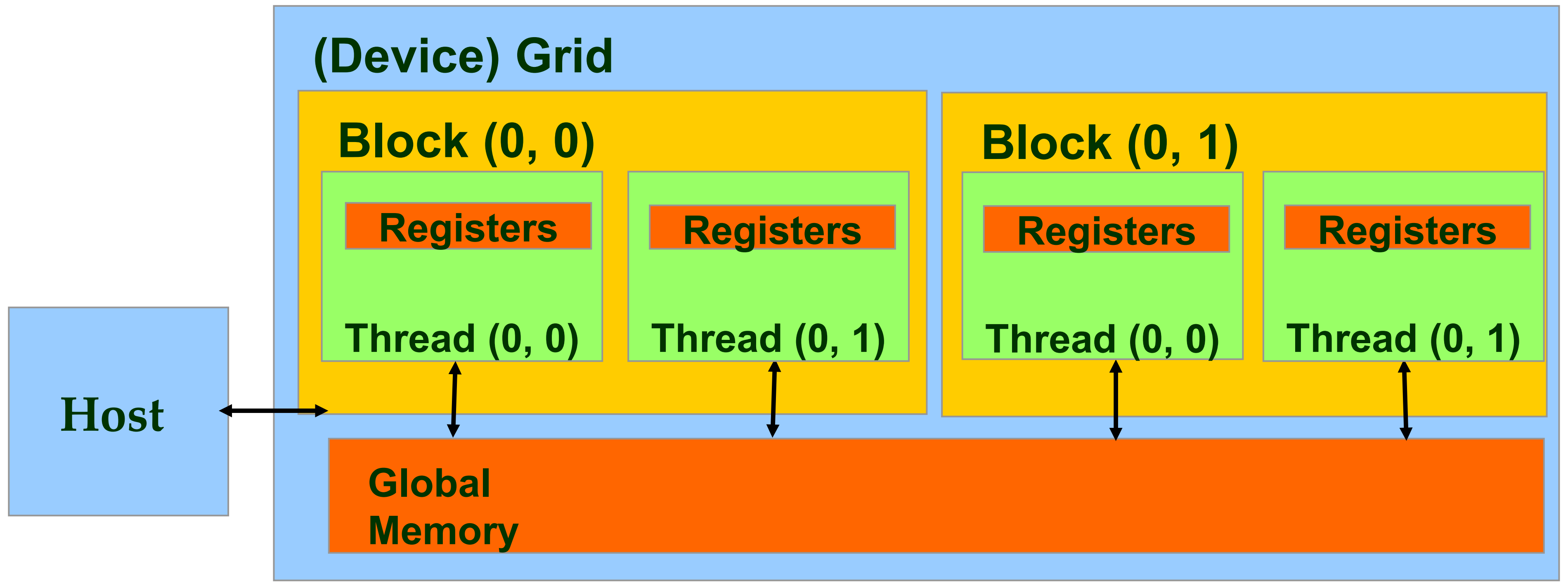
Memory management tasks are done on the host

1. Allocate memory space on the host
2. Transfer the data to the device using built-in APU
3. Retrieve the data back to the host
4. Free the allocated memory

Partial Overview of CUDA Memories

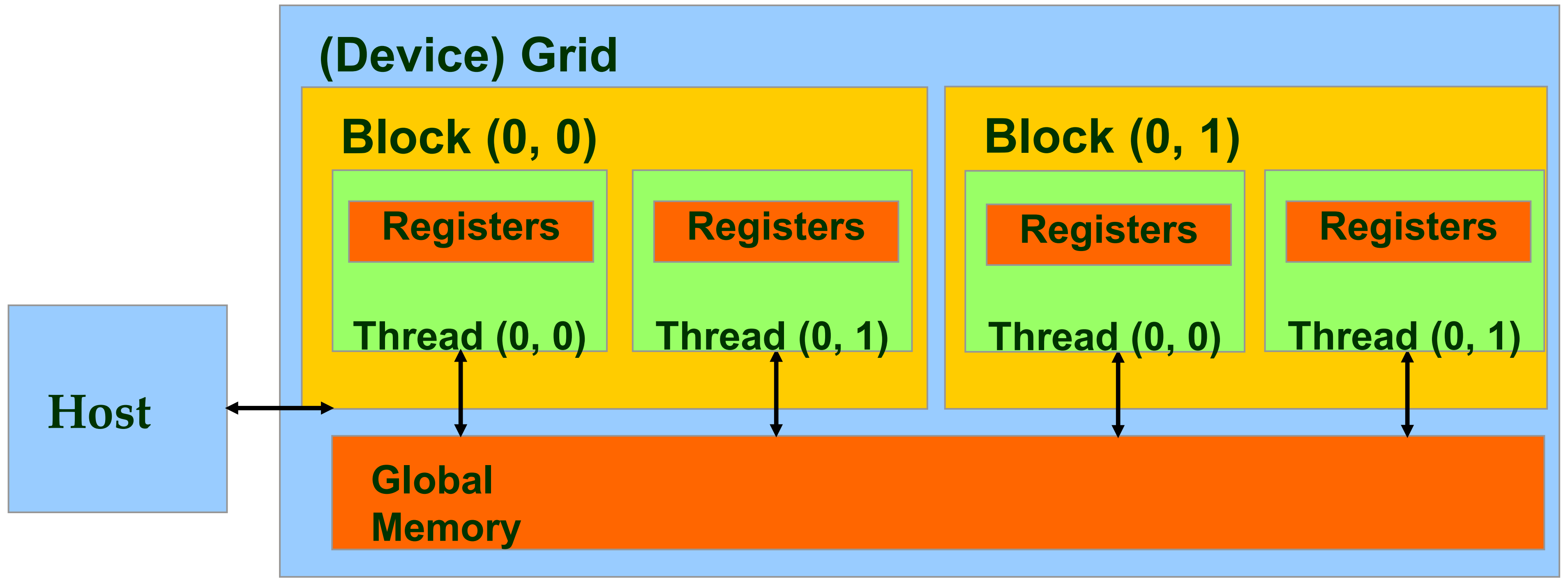
- Device code can:
  - R/W per-thread registers
  - R/W all-shared global memory

- Host code can
  - Transfer data to/from per grid global memory



CUDA Device Memory Management API functions

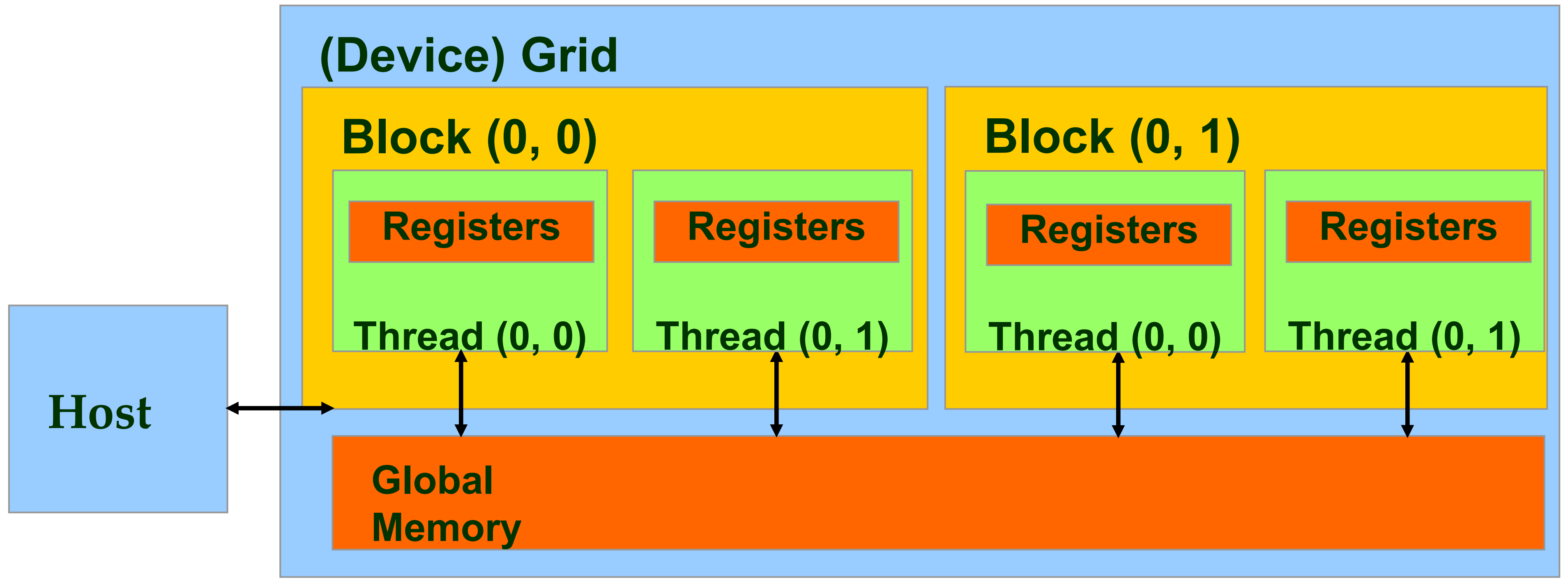
- cudaMalloc()
  - Allocates an object in the device global memory
  - Two parameters
    - **Address of a pointe**r to the allocated object
    - **Size of** allocated object in terms of bytes
- cudaFree()
  - Frees object from device global memory
  - One parameter
    - **Pointer** to freed object



Host-Device Data Transfer API functions

- cudaMemcpy()
  - memory data transfer
  - Requires four parameters
    - Pointer to destination
    - Pointer to source
    - Number of bytes copied
    - Type/Direction of transfer

  - Transfer to device is synchronous with respect to the host

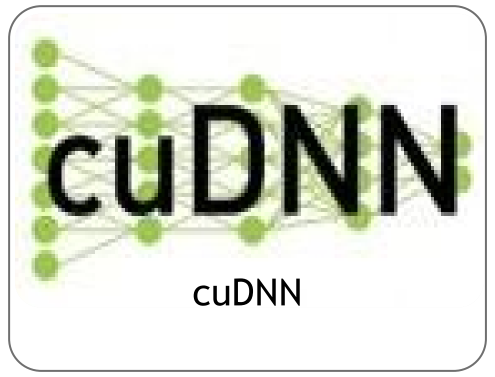


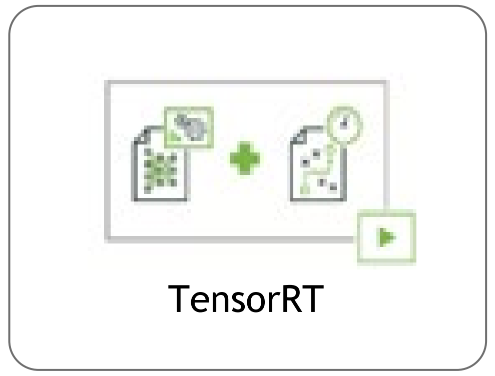
CUDA accelerated libraries

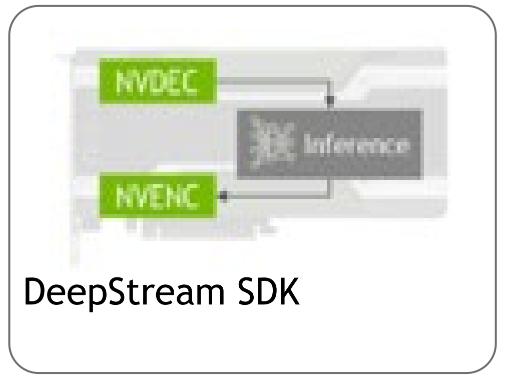
- CUDA Toolkit (nvcc compiler)
- cuBLAS, cuFFT, cuDNN
- CuPy (numpy for CUDA)

CUDA accelerated libraries

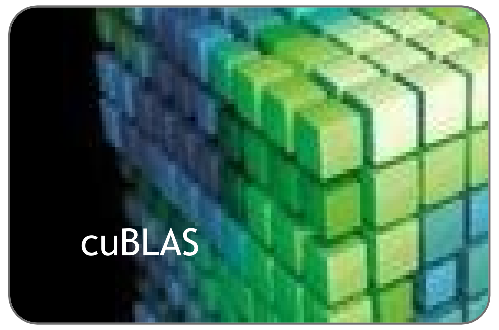
DEEP LEARNING

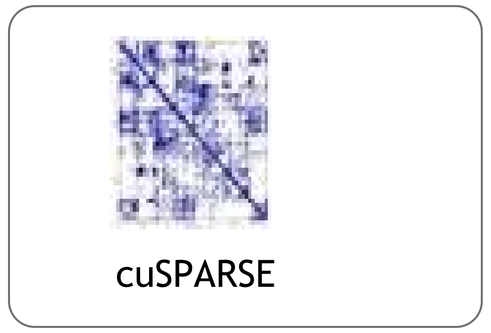




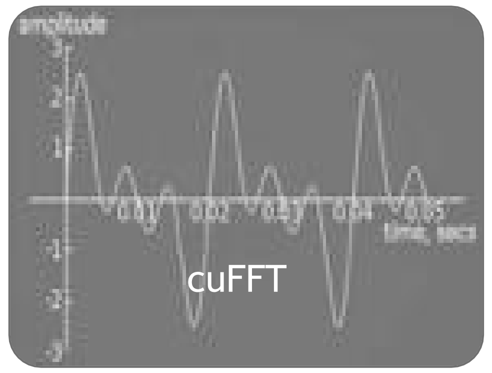


LINEAR ALGEBRA

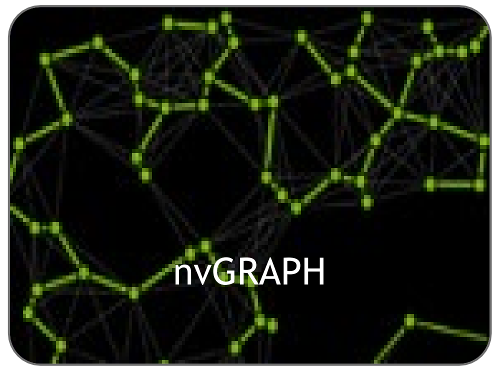


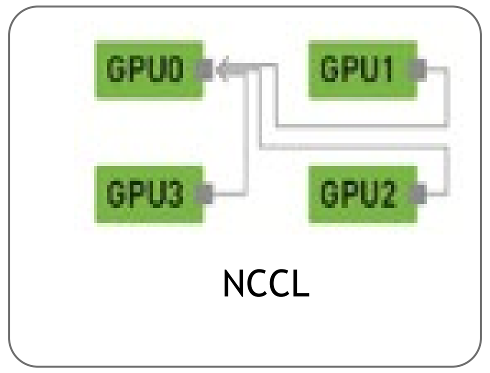


SIGNAL, IMAGE, VIDEO

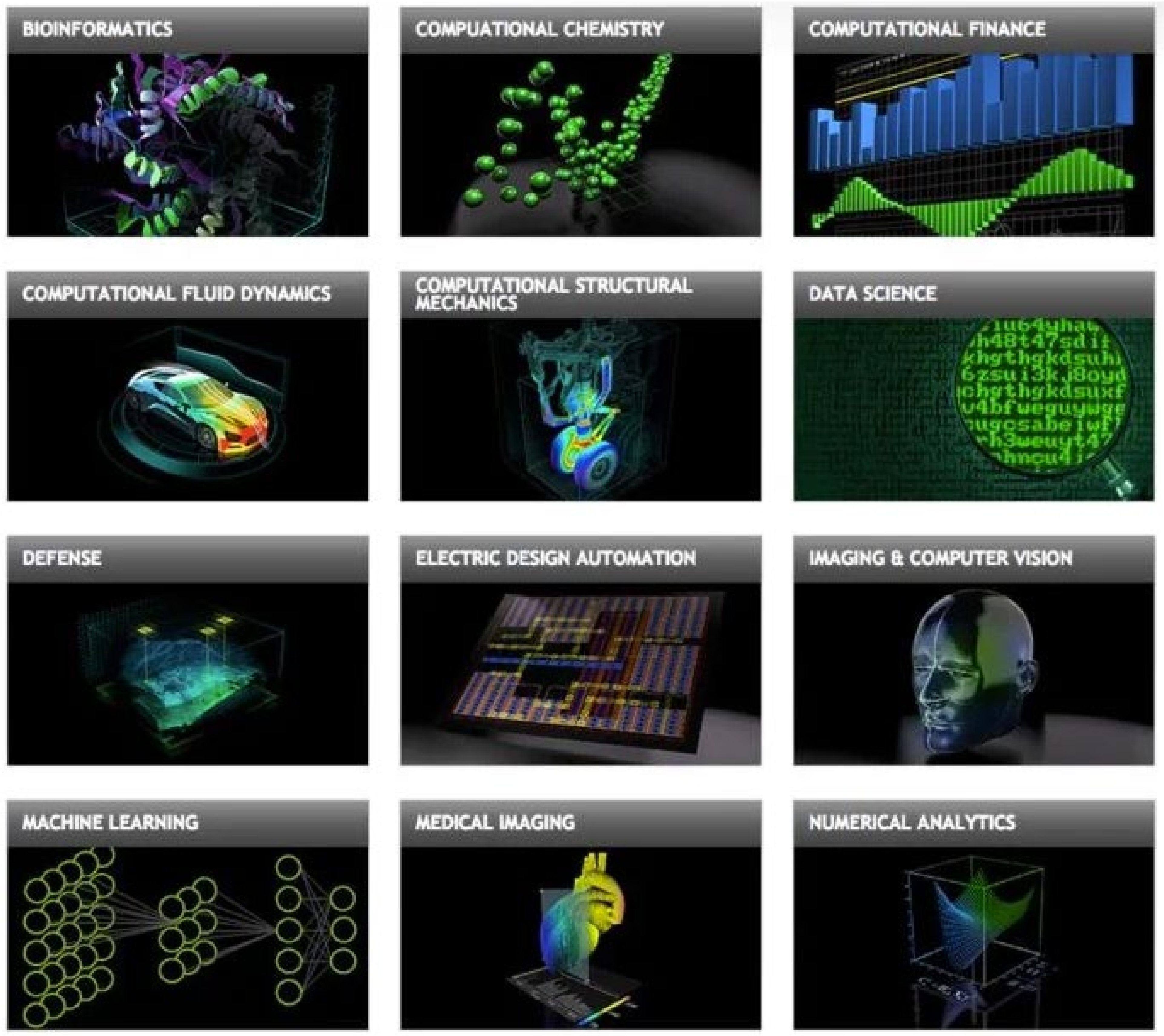


PARALLEL ALGORITHMS





CUDA application domains


BIOINFORMATICS
COMPUATIONAL CHEMISTRY
COMPUTATIONAL FINANCE
COMPUTATIONAL FLUID DYNAMICS
COMPUTATIONAL STRUCTURAL MECHANICS
DATA SCIENCE
DEFENSE
ELECTRIC DESIGN AUTOMATION
IMAGING & COMPUTER VISION
MACHINE LEARNING
MEDICAL IMAGING
NUMERICAL ANALYTICS