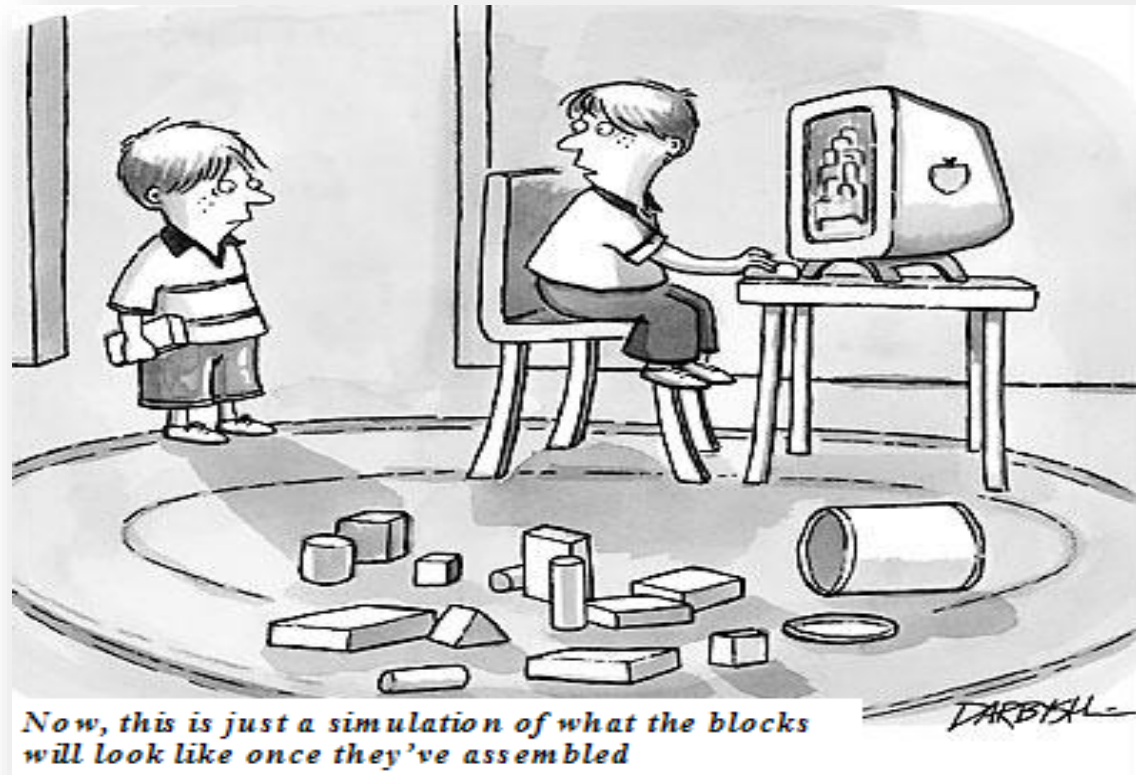


ECE569

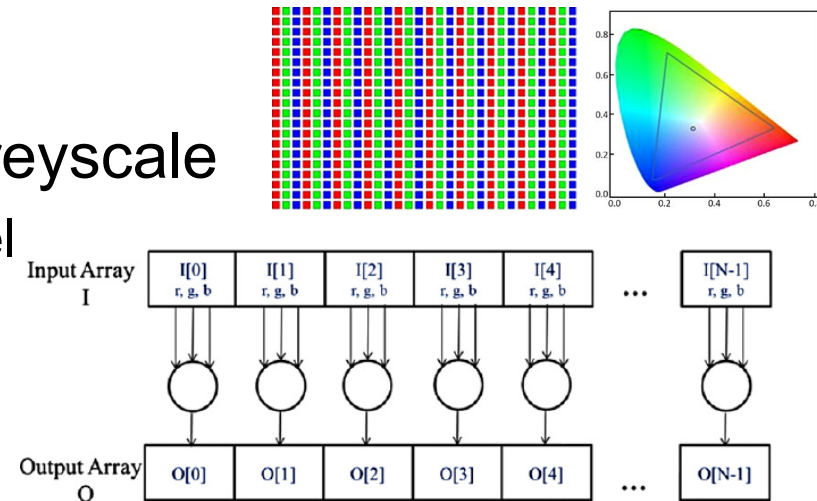
Module 5



- Data vs Task Parallelism

Data Level Parallelism

- Image and video processing
 - Converting a color pixel to a greyscale
 - requires only the data of that pixel
 - $L=r*0.21+g*0.72+b*0.07$
 - Blurring an image
 - averages each pixel's color with the colors of nearby pixels, requiring only the data of that small neighborhood of pixels.
- Data parallelism principle:
 - Break computations down into many smaller computations
 - organize the computation around the data,
 - execute the resulting independent computations in parallel to complete the overall job faster, much faster.



Data Parallelism vs Task Parallelism

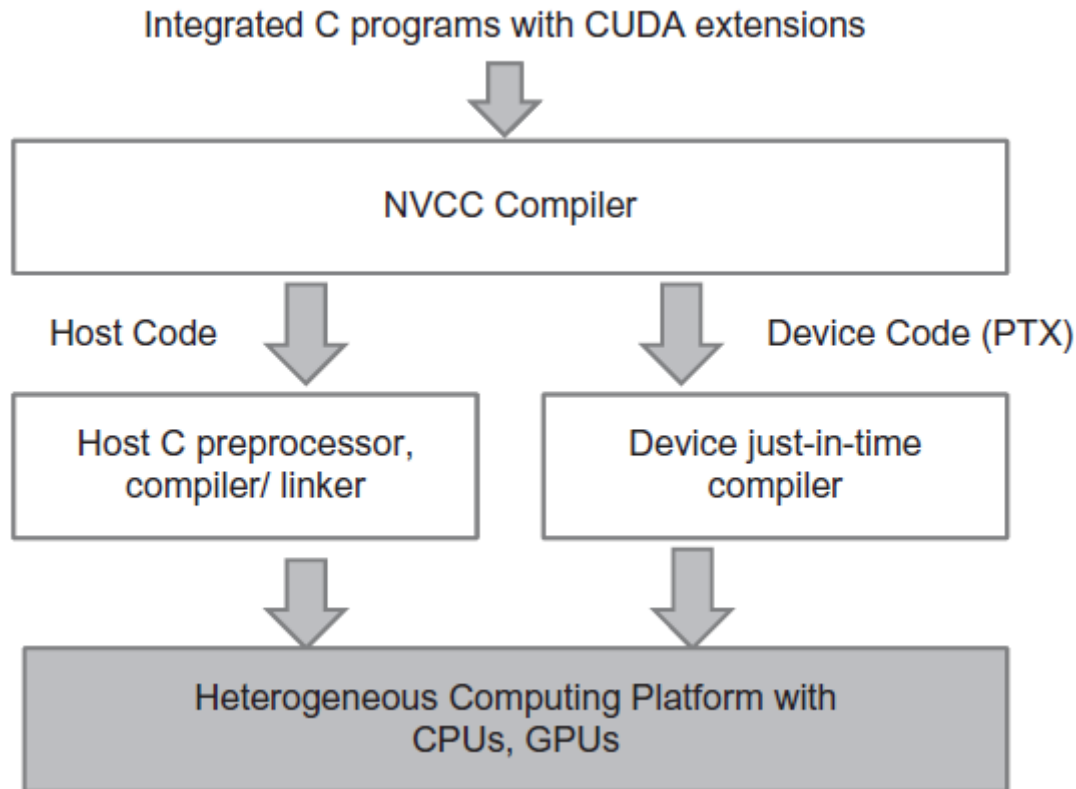
- Task parallelism is typically exposed through decomposing the job into parallel tasks.
- molecular dynamics simulator, tasks include
 - vibrational forces,
 - rotational forces,
 - neighbor identification for non-bonding forces,
 - non-bonding forces,
 - velocity and position
- Data parallelism offers fine grained and much more scalable parallelism!

CUDA Programming Model

- The GPU is viewed as a compute device that:
 - Is a co-processor to the CPU or host
 - Has its own DRAM (global memory in CUDA parlance)
 - Runs many threads in parallel
- Data-parallel portions of an application run on the device as kernels which are executed in parallel by many threads

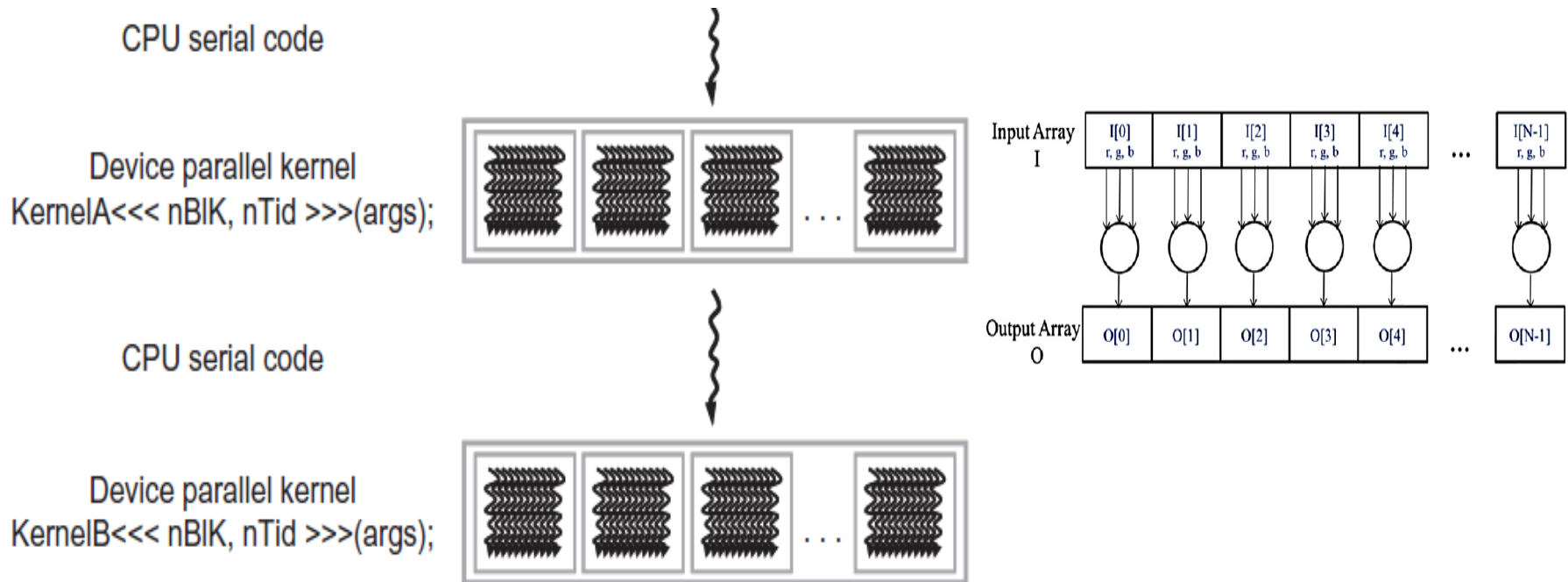
CUDA C Programming Model

- Each CUDA source file can have a mixture of both host and device code.
- NVCC (NVIDIA C Compiler)



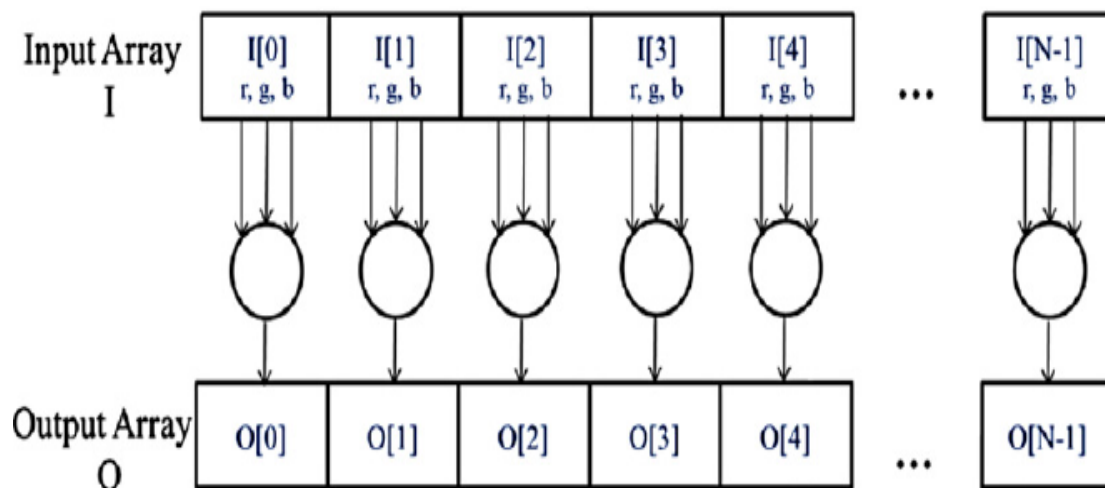
CUDA C Program Execution

- execution starts with host code (CPU serial code)



A Thread in CUDA

- simplified view of how a processor executes a sequential program in modern computers.



$$L = r * 0.21 + g * 0.72 + b * 0.07$$

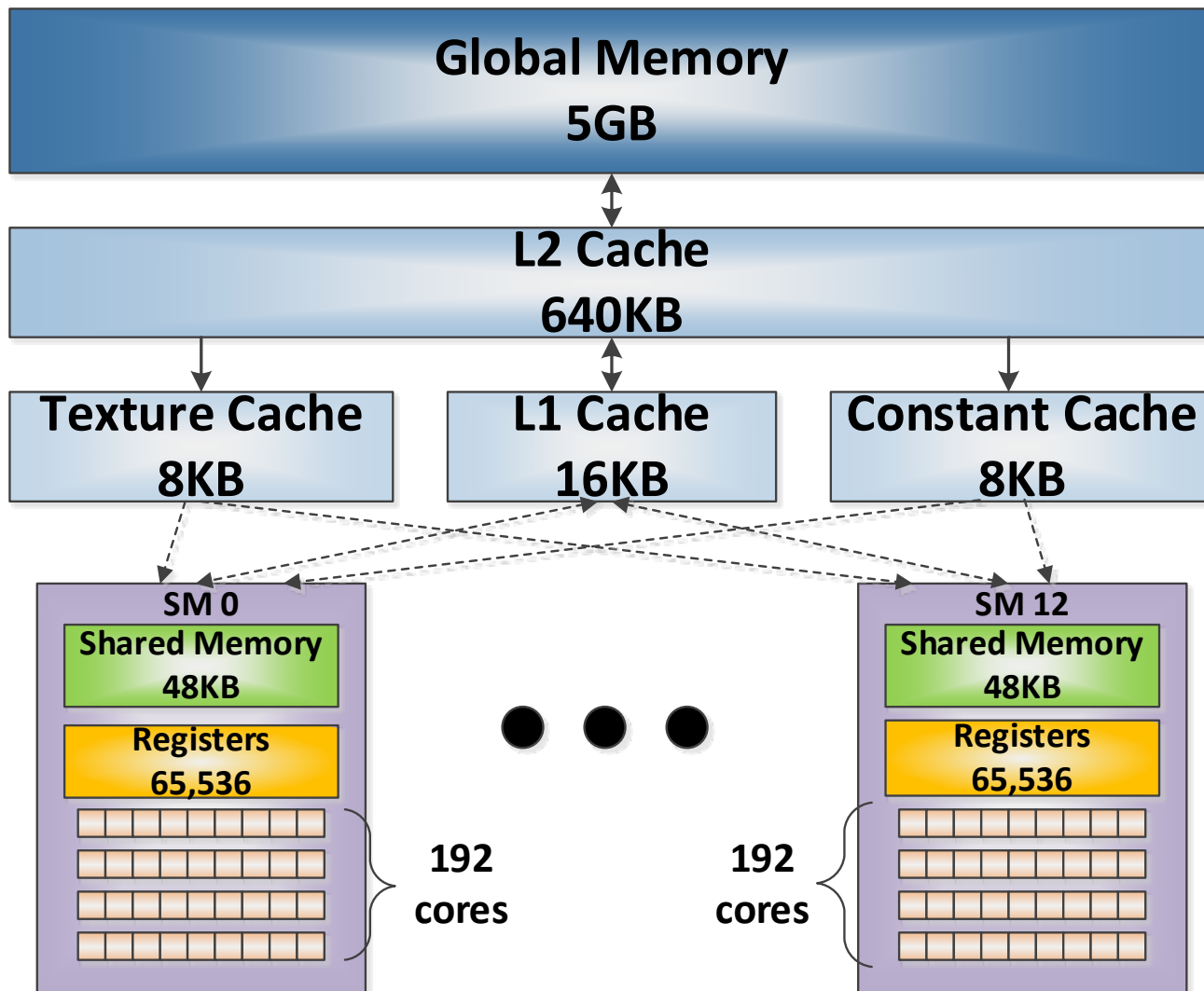
↓

$$\begin{aligned} Temp1 &= r * 0.21 \\ Temp2 &= g * 0.72 \\ Temp1 &= Temp1 + Temp2 \\ Temp2 &= b * 0.07 \\ Temp1 &= Temp1 + Temp2 \end{aligned}$$

Differences between GPU and CPU threads

- **GPU is a SIMD device: “streams” of data**
 - Each “GPU thread” executes one general instruction on the stream of data that the GPU is assigned to process
 - The NVIDIA calls this model SIMT (single instruction multiple thread)
 - GPU threads are extremely lightweight
 - Very little creation overhead
 - GPU needs 1000s of threads for full efficiency
 - A collection of Streaming Multiprocessors (SMs)
 - Each SM has a set of Scalar Processors (SPs)
 - Multi-core CPU needs only a few heavy ones

Tesla K20 GPU



Up to
2048
Threads/
SM

Pascal, Volta

- **K40: 30,720**
- **P100: 114,688**
- **V100: 163,840**

Feature	Tesla K40	P100	V100
Onboard Memory	12 GB	16 GB	16GB
# of SMX Units	15	56	80
# of CUDA Cores	2880	3584	5120
WARPS per SM	64	64	64
Threads per SM	2048	2048	2048
GPU Base Clock	745 MHz	1480 MHz	1530MHz
Wattage	235W	300W	300W
Transistors	7.1 billion	15.3	21.1

Compute Capability vs. CUDA Version





- <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capabilities>
- **Compute Capability of a Device” refers to hardware**
 - used by applications at runtime to determine which hardware features and/or instructions are available on the present GPU
 - Defined by a major revision number and a minor revision number

The “CUDA Version” indicates what version of the software you are using to run on the hardware

Compute Capability vs. CUDA Version



CUDA-Enabled NVIDIA GPUs

NVIDIA Ampere Architecture (compute capabilities 8.x)				Tesla A Series
NVIDIA Turing Architecture (compute capabilities 7.x)		GeForce 2000 Series	Quadro RTX Series	Tesla T Series
NVIDIA Volta Architecture (compute capabilities 7.x)	DRIVE/JETSON AGX Xavier		Quadro GV Series	Tesla V Series
NVIDIA Pascal Architecture (compute capabilities 6.x)	Tegra X2	GeForce 1000 Series	Quadro P Series	Tesla P Series
	 Embedded	 Consumer Desktop/Laptop	 Professional Workstation	 Data Center

Compute Capability

	Compute Capability												
Technical Specifications	3.5	3.7	5.0	5.2	5.3	6.0	6.1	6.2	7.0	7.2	7.5	8.0	8.6
Maximum number of resident grids per device (Concurrent Kernel Execution)	32				16	128	32	16	128	16	128		
Maximum dimensionality of grid of thread blocks	3												
Maximum x-dimension of a grid of thread blocks	2 ³¹ -1												
Maximum y- or z-dimension of a grid of thread blocks	65535												
Maximum dimensionality of a thread block	3												
Maximum x- or y-dimension of a block	1024												
Maximum z-dimension of a block	64												
Maximum number of threads per block	1024												
Warp size	32												
Maximum number of resident blocks per SM	16		32								16	32	16
Maximum number of resident warps per SM	64										32	64	48
Maximum number of resident threads per SM	2048										1024	2048	1536
Number of 32-bit registers per SM	64 K	128 K	64 K										
Maximum number of 32-bit registers per thread block	64 K				32 K	64 K		32 K	64 K				
Maximum number of 32-bit registers per thread	255												
Maximum amount of shared memory per SM	48 KB	112 KB	64 KB	96 KB	64 KB		96 KB	64 KB	96 KB		64 KB	164 KB	100 KB
Maximum amount of shared memory per thread block ³¹	48 KB								96 KB	48 KB	64 KB	163 KB	99 KB
Number of shared memory banks	32												

Next

- **Memory management and data movement using “Vector addition” as example**