**"An Even Easier Introduction to CUDA"**

**✅ Core Takeaways from the Course**

**🔹 1. What CUDA Is**

* CUDA is NVIDIA’s toolkit to **run programs on the GPU**.
* It allows you to offload parallel tasks from CPU to GPU for massive speed gains.

**🔹 2. How GPU Programming Differs from CPU**

* GPU runs **thousands of lightweight threads in parallel**.
* You write a *kernel* function that gets executed by each GPU thread.
* CPU: One or few tasks at a time. GPU: Many tasks **simultaneously**.

**🔹 3. Kernel Functions**

* Special functions marked with \_\_global\_\_.
* Called from the host (CPU), executed on the device (GPU).
* Syntax: kernel<<<numBlocks, numThreads>>>(args);

**🔹 4. Thread and Block Concepts**

* threadIdx.x → ID of a thread inside a block.
* blockIdx.x → ID of a block inside the grid.
* blockDim.x → Total threads in a block.
* This allows each thread to figure out its **global index** using:
* int index = threadIdx.x + blockIdx.x \* blockDim.x;

**🔹 5. Unified Memory**

* Used cudaMallocManaged() to allocate memory accessible by both CPU and GPU.
* No need to do separate cudaMemcpy() for transferring data.

**🔹 6. Grid-Stride Loop**

* Used to scale your program for **any size** input, even if you don’t have enough threads.
* This pattern is widely used in real CUDA programs.

**🔹 7. Performance Comparison**

* You observed the performance difference between:
  + CPU version
  + Single-threaded GPU
  + Multi-thread (single block)
  + Multi-block, multi-thread fully parallelized GPU
* Final version should run **10–30× faster** than CPU.

**🔹 8. Correctness Validation**

* Printed max error (should be 0.0 or very tiny), showing GPU result matches CPU result.
* Shows GPU isn't just fast—it’s accurate when coded right.

**🧠 Concepts You Now Understand (Or Should Review If Not)**

| **Concept** | **Use It For** |
| --- | --- |
| **Kernels** | Writing code to run on GPU |
| **Threads & Blocks** | Spreading work across GPU cores |
| **Memory allocation** | Sharing data between CPU and GPU |
| **Performance tuning** | Launching optimal thread/block sizes |
| **Grid-stride loop** | Handling arrays larger than available threads |