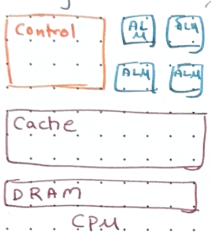
CUDA Introduction

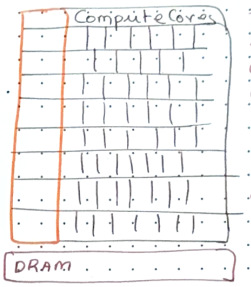
CPU designed to minimize latency



→ CPU
→ majority of silicon is dedicated to
• Advanced Control logic
• Large Cache

→ GPU
→ Cores are Called CUDA Cores

→ To Access Comptue Cores they are accesed by Kernels which are Basic Building Blocks to access the massive Compute Cores (CUDA Cores)



These massive Cores can Compute Basic operation in parallel and faster

GPU

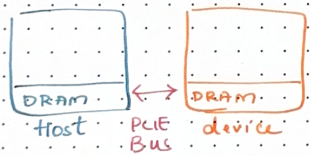
→ majority of silicon area dedicated to
→ massive # of Cores.
→ Don't need to a advanced/sophesticated Control logic as sophisticated as CPU.

Kernels
Function that Runs On GPU.

Threads
Kernels execute as a set parallel threads.

CUDA
⇒ Heterogenous Host + device
⇒ C with extensions
Allow Heterogenous Programming on Nvidia GPU's

⇒ In CUDA Host is in Control and Context passes to GPU only when required.
⇒ The transfer B/w Host to device & vice versa are Costly operations.

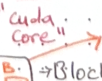


Concept of threads:
→ Kernels execute as a set of parallel threads.
→ Cuda is designed to execute 1000's or even more threads.
→ Threads executions are in SIMD / SIMT → single instruction multiple thread
→ Threads are Similar to data parallel tasks:
→ Each thread one a subset of data.
→ Threads execute indepently.

* Threads do not execute at the same Rate
→ This gives Rise to various performance Bottlenecks and Can Be optimized Further.

Organization & Hierarchy of threads:
* Thread * Blocks * Grid

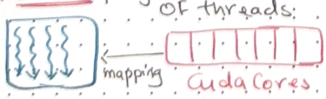
* Thread lowest level in hierarchy.
Execution of Kernel on a single piece of data.
Each thread gets mapped to One Core in GPU.



GridDim = 3×2 = 6 Blocks

→ Blocks are Grouped into Grids
→ Each Kernel launch Creats a Single Grid.
* Grids have dimensions 1D or 2D or 3D of Blocks

* Blocks | They have 1D or 2D or 3D structure of threads:



* when the Kernel is launched. Block gets mapped to Correspondg set of Cuda Cores.