

GPU architecture basics

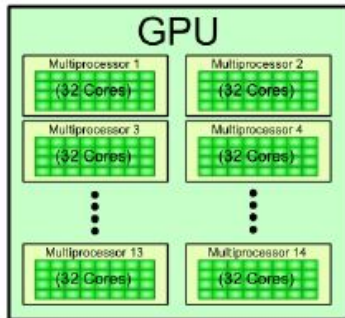
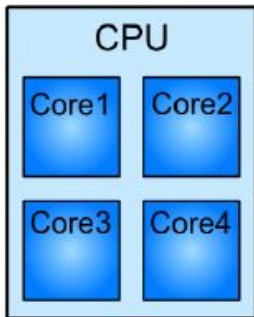
Triton abstracts away many low-level details of how GPUs work so that you don't have to think about them

This lesson is just a rough primer; do not feel like you need to understand the specifics of each diagram. It'll make more sense when we start coding

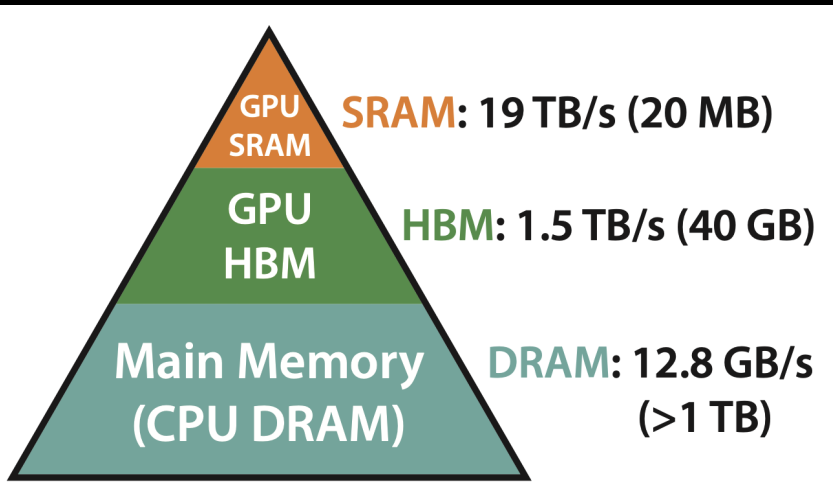


CPU vs GPU

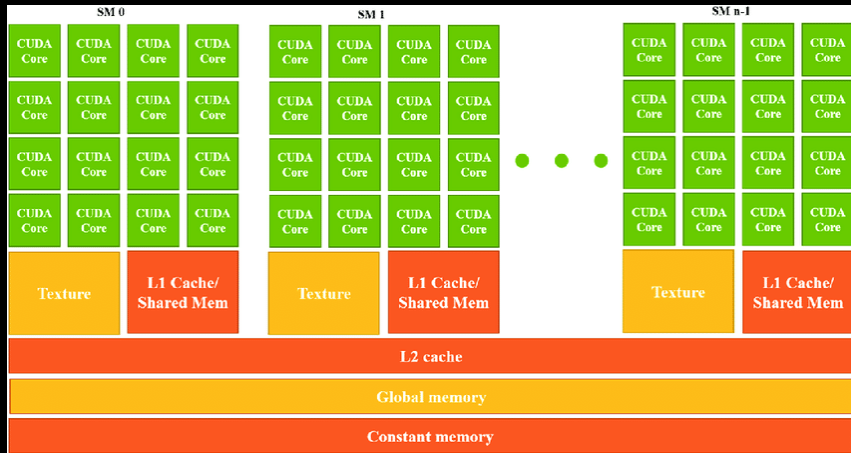
CPU/GPU Architecture Comparison



memory hierarchy



one streaming multi-processor (SM) per pool of SRAM



programs

a program is a specific instance of our kernel code run in parallel alongside many others. It is differentiated from other programs by the chunk of data it is assigned to work on

- ▶ each program is defined by a program ID (PID) which is a tuple full of integers
 - ▶ we use this PID alongside indexing logic to figure out which chunk of a tensor the program works on
- ▶ at least one program is called per SM, depending on how many can fit
 - ▶ $\text{number of PIDs per SM} = \text{amount of SRAM} // \text{SRAM required per PID}$
- ▶ if your code requires more data per single program than fits within the SM's pool of SRAM, it'll error
- ▶ If different PIDs in the same SM load the same data, then they'll share rather than loading & storing duplicates (we will take advantage of this)

cores & warps

A core is the smallest unit of compute; it performs the actual floating point operations

- ▶ unlike CPU cores, GPU cores cannot do any arbitrary operation; they're built primarily for FLOPs
- ▶ one GPU core performs one FLOP at a time

A warp is the smallest grouping of GPU cores

- ▶ 32 cores per warp on Nvidia; 64 on AMD
- ▶ all cores in a warp must perform the exact same operation
- ▶ there are multiple warps per PID; you don't have to worry about how many