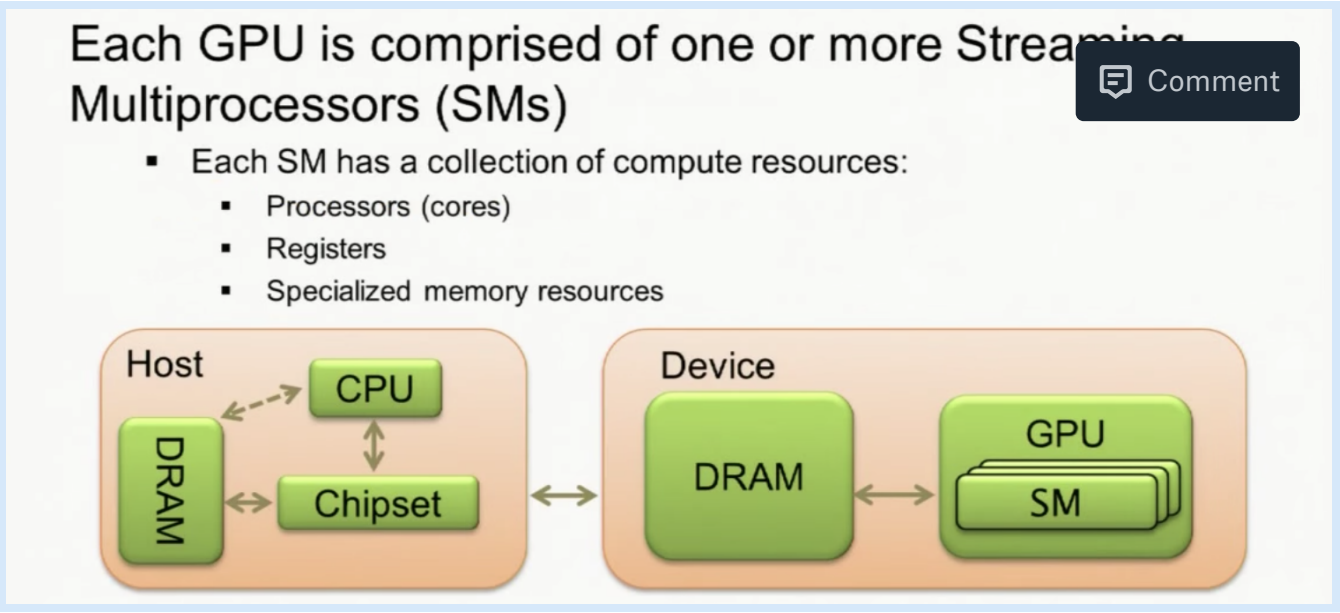
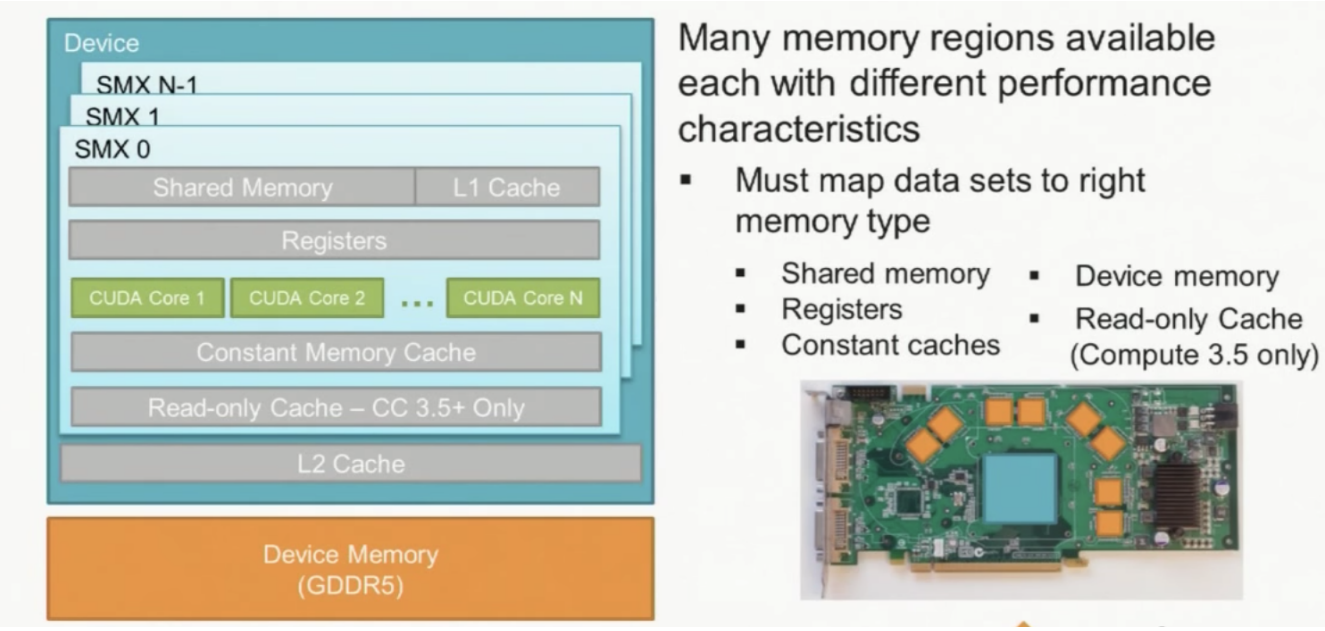
gpu memory model

- `nvcc` is the NVIDIA compiler that compiles device code for the GPU.
- Task parallelism is the simultaneous execution on multiple cores of many different functions across the same or different datasets. Data parallelism (aka SIMD) is the simultaneous execution on multiple cores of the same function across the elements of a dataset.
- think of a GPU as a collection of streaming multiprocessors (SMs)



- Each GPU has a few 10s of SMs each with cores numbering in the few 100s
- remember, every GPU task has a grid with some dimensions. at each slot in the grid is a block, that has a group of threads (can be multidimensional)
- it is guaranteed that each block will be scheduled on the same SM. Multiple blocks can be scheduled on the same SM.
- There are no guarantees on the order of scheduling of blocks so we should make no assumptions here. the device schedules dynamically → this is why GPU results with floating point operations are non-deterministic.
- However, within threads within the same block, we can coordinate between threads using synchronization primitives.
- `__syncthreads()` acts as a barrier and will make the thread wait for all other threads in the same block (**not** across threads in different blocks).
- GPU architecture (→ SM equivalent to an Intel processor core with SMT?)



- when we say a GPU has a few gigs of memory, the device memory (GDDR5 above) is the memory we are talking about → called global memory
- local variables etc are by default thread local memory
- here is the details of each memory above

Memory Space	Managed by	Physical Implementation	Scope on GPU	Scope on CPU	Lifetime
Registers	Compiler	On-chip	Per Thread	Not visible	Lifetime of a thread
Local	Compiler	Device Memory	Per Thread	Not visible	
Shared	Programmer	On-chip	Block	Not visible	Block lifetime
Global	Programmer	Device Memory	All threads	Read/Write	Application or until explicitly freed
Constant	Programmer	Device Memory	All threads Read-only	Read/Write	

- on chip is always faster than device (cuz its nearer) by around an order or two of magnitude
- shared memory can be allocated statically or dynamically using the `__shared__` keyword.
- the key is on chip memory is cheaper than device memory and registers are faster than shared on chip memory. so bring your data in as close as possible before computing → it makes a huge difference in speed!