CUDA Programming Model Overview

05 sep ,IISc SERC

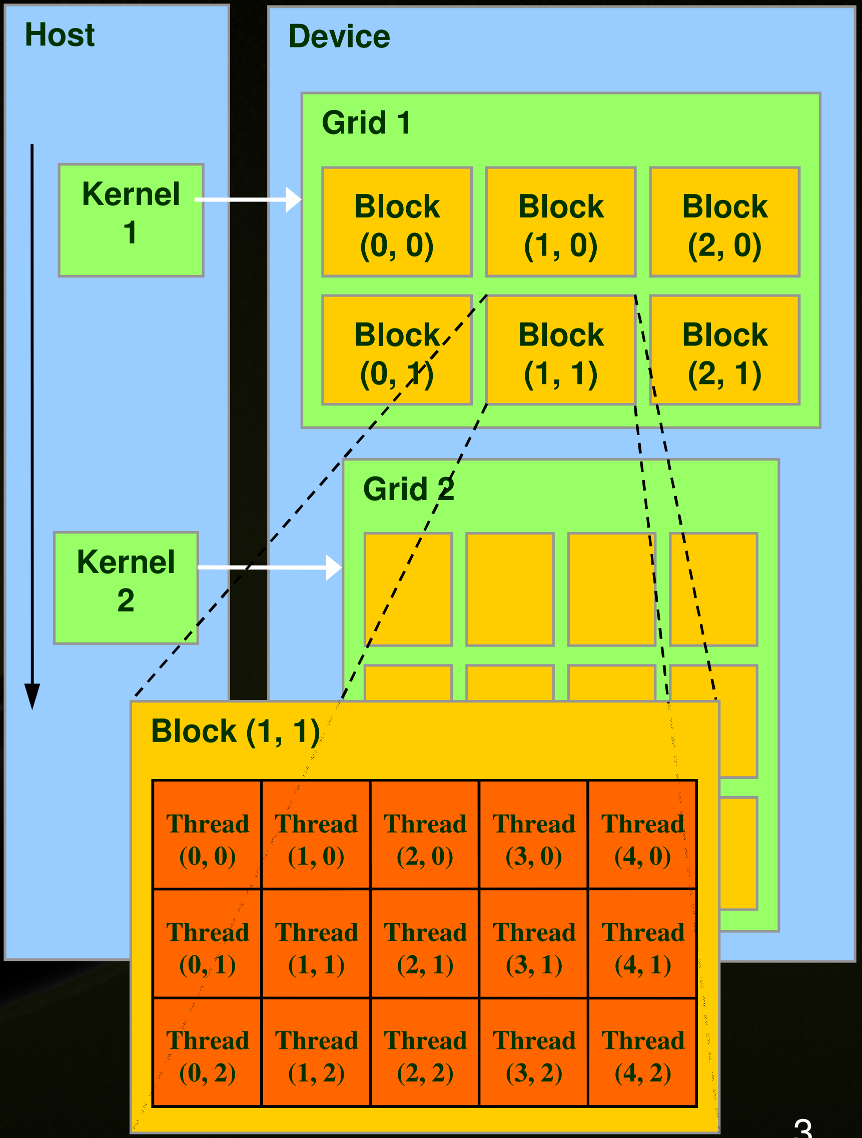
Punit Kishore

CUDA Programming Model

- Parallel portions of an application are executed on the device as kernels
  - One kernel is executed at a time
  - Many threads execute each kernel
- Differences between CUDA and CPU threads
  - CUDA threads are extremely lightweight
    - Very little creation overhead
    - Instant switching
  - CUDA uses 1000s of threads to achieve efficiency
    - Multi-core CPUs can use only a few

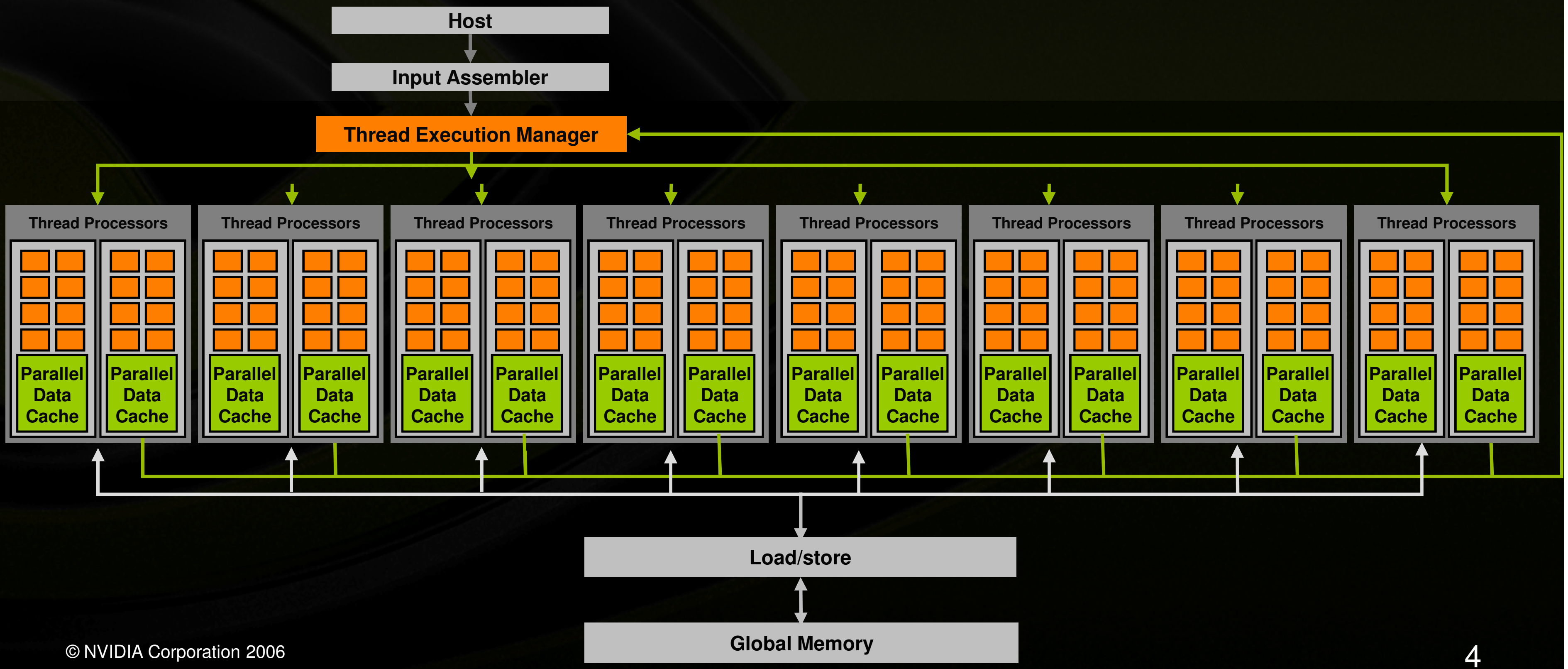
Programming Model

- A kernel is executed as a grid of thread blocks
- A thread block is a batch of threads that can cooperate with each other by:
  - Sharing data through shared memory
  - Synchronizing their execution
- Threads from different blocks cannot cooperate



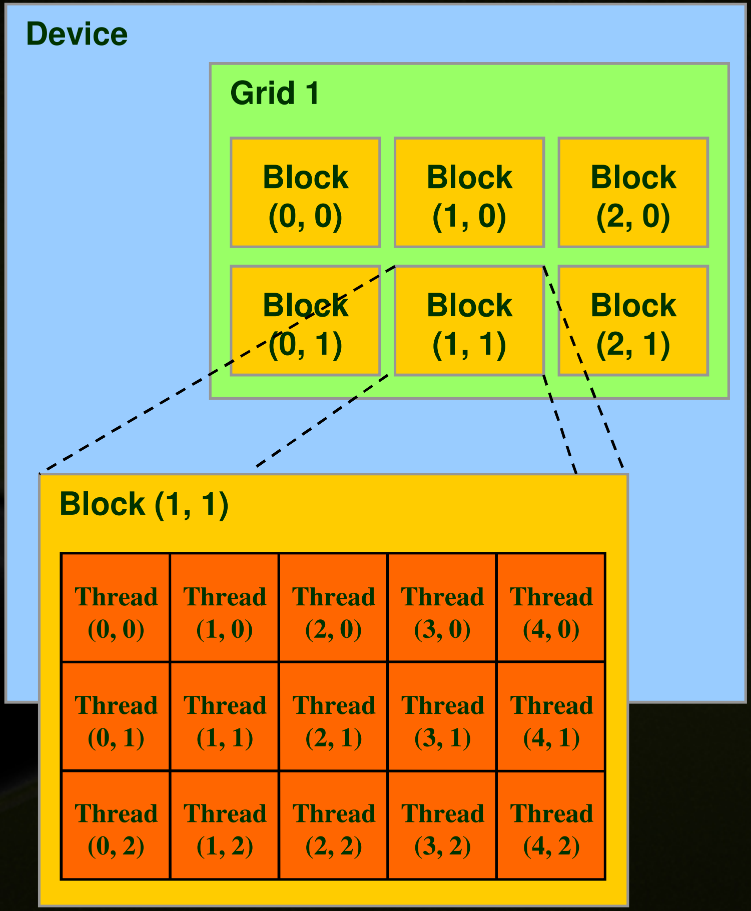
G80 Device

- Processors execute computing threads
- Thread Execution Manager issues threads
- 128 Thread Processors
- Parallel Data Cache accelerates processing



Programming Model

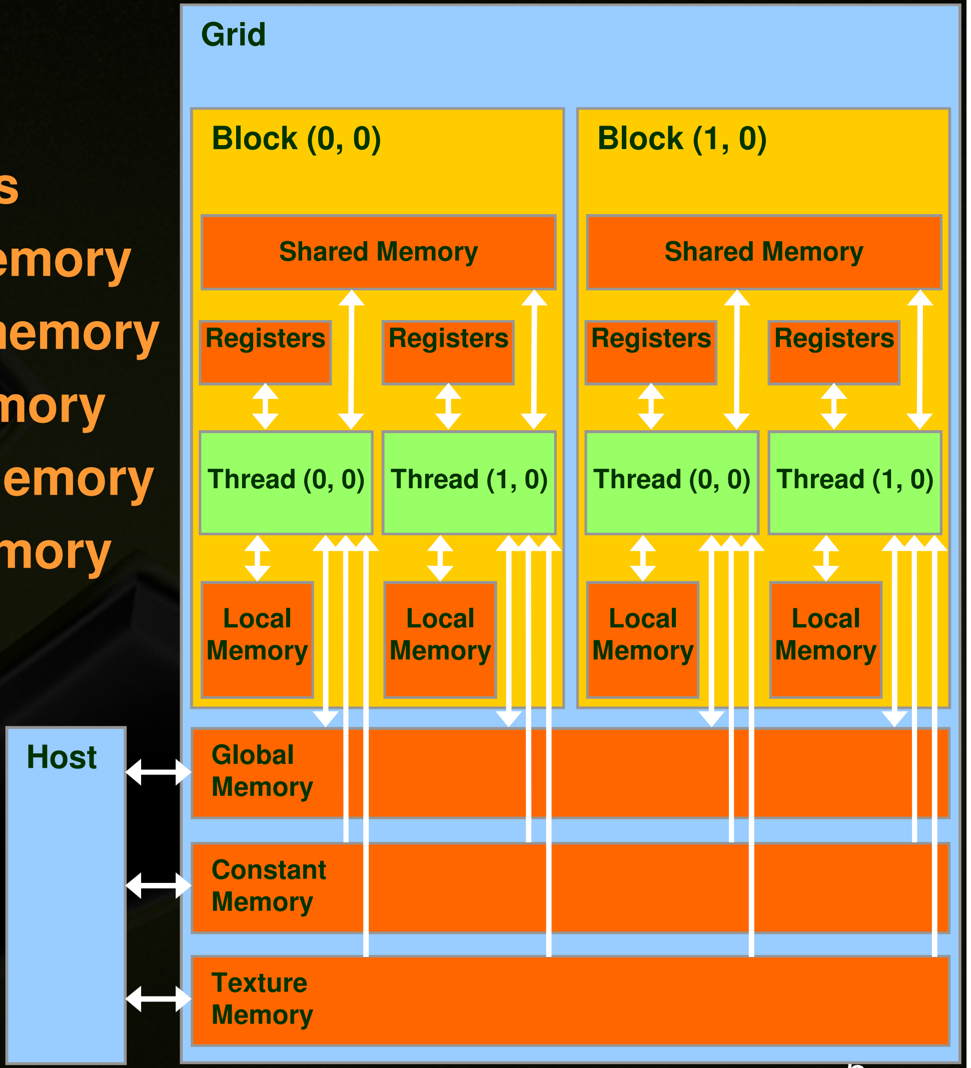
- Threads and blocks have IDs
  - So each thread can decide what data to work on
- Block ID: 1D or 2D
- Thread ID: 1D, 2D, or 3D
- Simplifies memory addressing when processing multidimensional data
  - Image processing
  - Solving PDEs on volumes



Programming Model: Memory Spaces

- Each thread can:
  - Read/write per-thread **registers**
  - Read/write per-thread **local memory**
  - Read/write per-block **shared memory**
  - Read/write per-grid **global memory**
  - Read only per-grid **constant memory**
  - Read only per-grid **texture memory**

- The host can read/write global, constant, and texture memory (stored in DRAM)



Execution Model

- Kernels are launched in grids
  - One kernel executes at a time
- A block executes on one multiprocessor
  - Does not migrate
- Several blocks can execute concurrently on one multiprocessor
  - Control limitations:
    - At most 8 concurrent blocks per SM
    - At most 768 concurrent threads per SM
  - Number is further limited by SM resources
    - Register file is partitioned among the threads
    - Shared memory is partitioned among the blocks

CUDA Advantages over Legacy GPGPU

- Random access to memory
  - Thread can access any memory location
- Unlimited access to memory
  - Thread can read/write as many locations as needed
- User-managed cache (per block)
  - Threads can cooperatively load data into SMEM
  - Any thread can then access any SMEM location
- Low learning curve
  - Just a few extensions to C
  - No knowledge of graphics is required
- No graphics API overhead

CUDA Model Summary

- Thousands of lightweight concurrent threads
  - No switching overhead
  - Hide instruction latency
- Shared memory
  - User-managed L1 cache
  - Thread communication within blocks
- Random access to global memory
  - Any thread can read/write any location(s)
- Current generation hardware:
  - Up to 128 streaming processors

Memory	Location	Cached	Access	Who
Local	Off-chip	No	Read/write	One thread
Shared	On-chip	N/A	Read/write	All threads in a block
Global	Off-chip	No	Read/write	All threads + host
Constant	Off-chip	Yes	Read	All threads + host
Texture	Off-chip	Yes	Read	All threads + host