Graphics Processing Unit

Computer Division
Bhabha Atomic Research Centre

Topics

- Evolution of Performance
- Applications of GPU
- CPU v/s GPU
- GPU as a Co-processor
- Heterogeneous Computing
- GPU Architecture
- Streaming Multiprocessor
- Memory Hierarchy
- Thread Hierarchy

Evolution of Performance (1)

Early Processors:

- Single core
- Increase clock speed
  - Number of cycles per second
  - Number of instructions per second
- Faster execution of individual instructions
- Limitations of Clock Speed
  - Physical limits on clock speeds
  - Power or heat dissipation

Evolution of Performance (2)

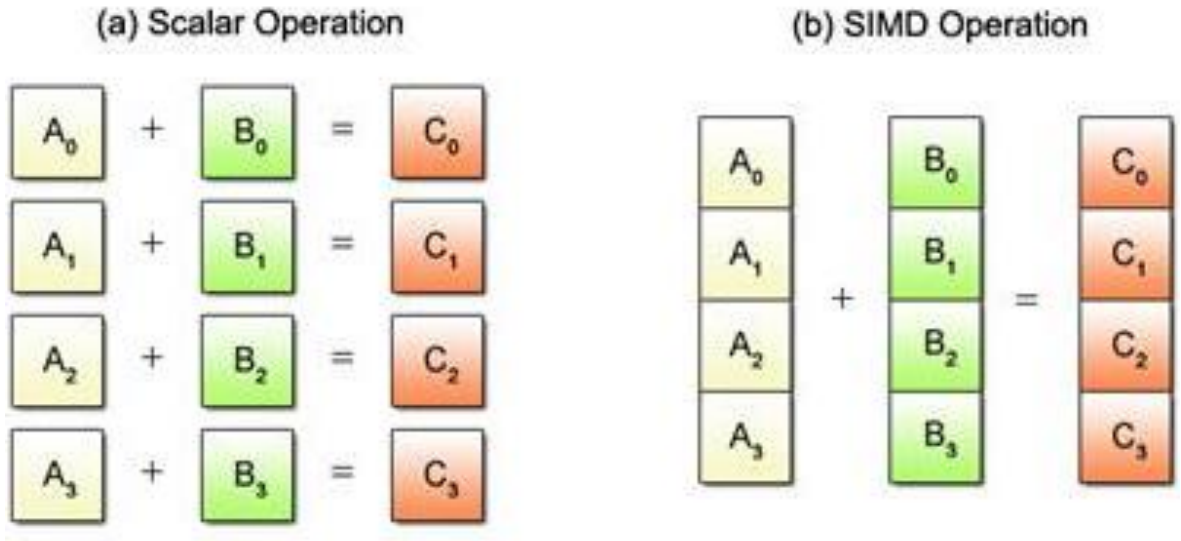
Multi Core Architecture:

- Multiple cores on Single chip
- Parallel execution of tasks
- Significant overall performance improvement
- Multiple cores – Different threads – Simultaneous execution
- Multi threaded tasks – Performance improvement
- Power Efficiency:
    - Individual cores
    - Lower clock speed
    - Better power efficiency

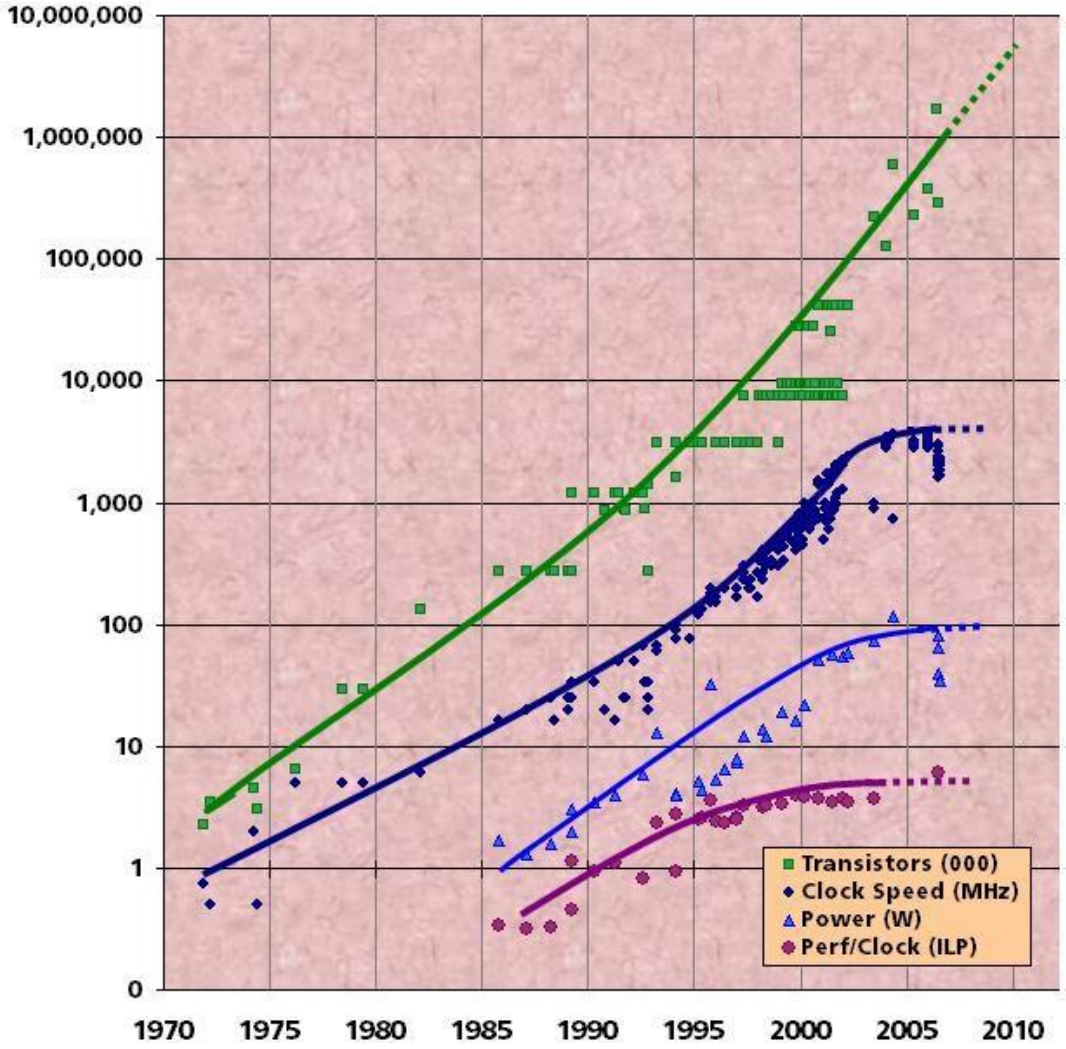
Evolution of Performance (3)

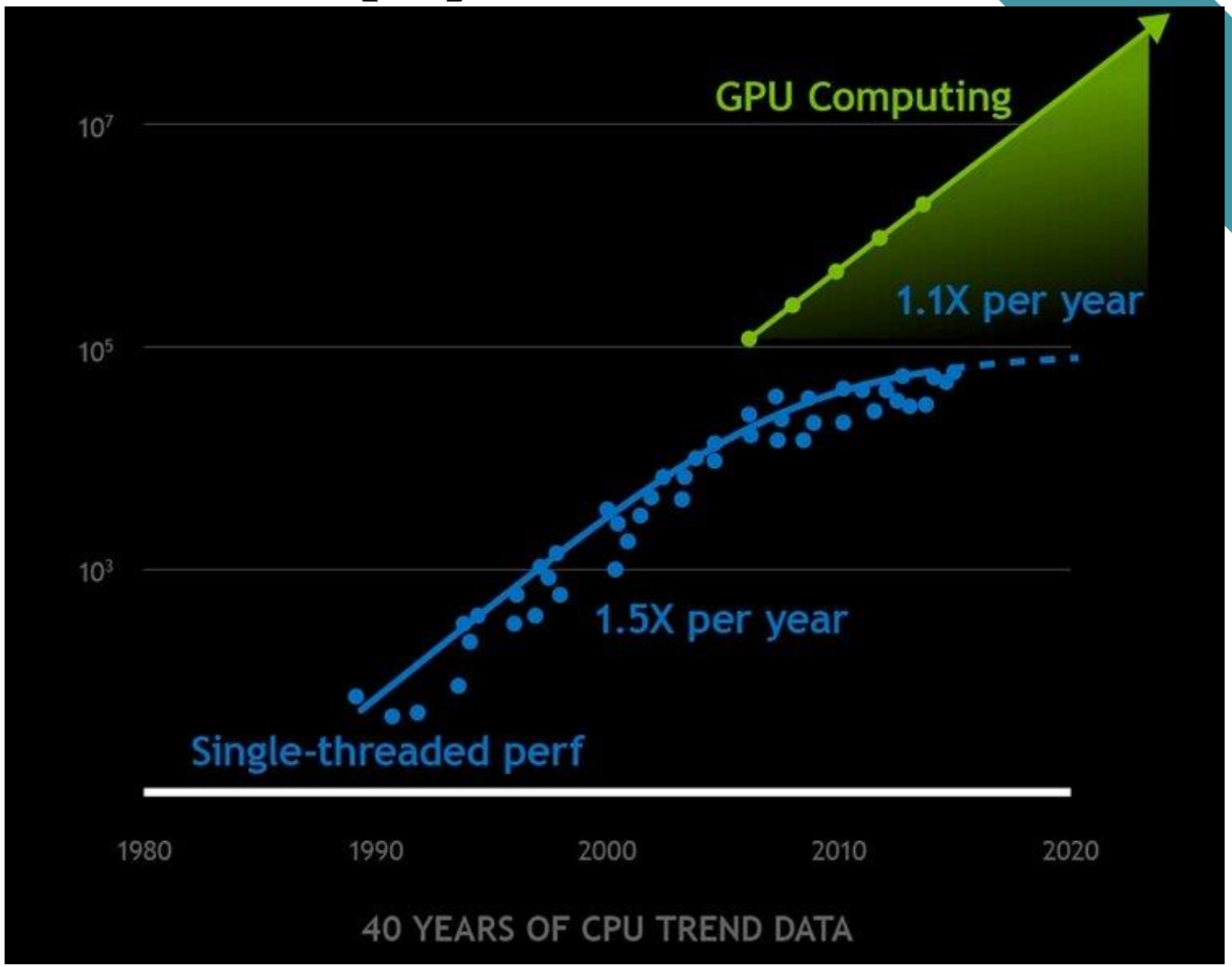
Heterogeneous Computing

- GPU – Graphics Processing Unit
- Specialized processor
  - Process large amounts of data
  - High performance
  - Many core processors
  - Very high memory bandwidth
  - TeraFLOPs peak performance
- Single Instruction Multiple Data (SIMD)
- Offload computationally intensive tasks to GPU



Evolution of Performance (4)





Applications of GPU

Computer Graphics – Image Synthesis

- Generates images – Video games, Animated movies
- X-ray computed tomography

Image Processing

- Image analysis – Template matching, SURF features

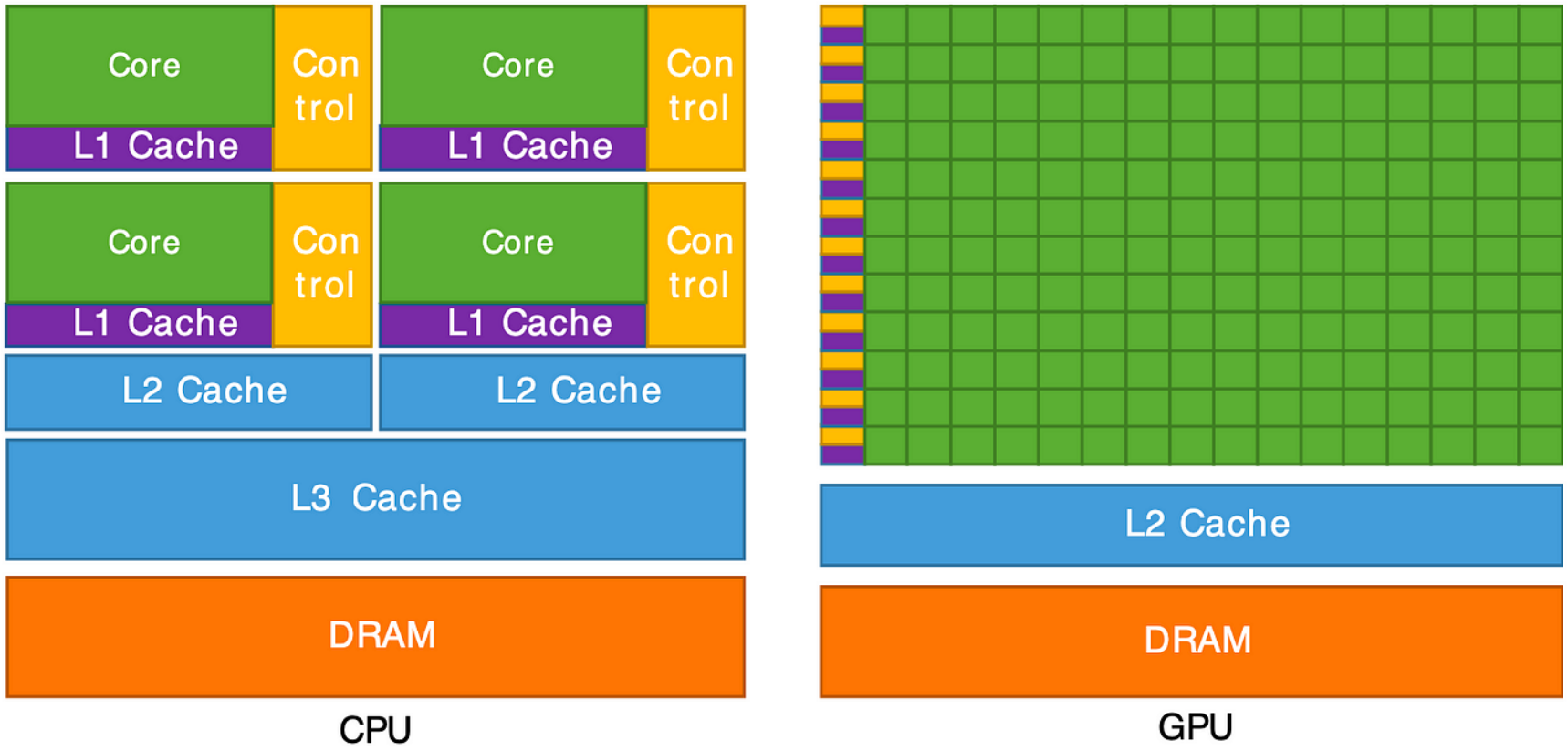
High Performance Computing

- Computational Fluid Dynamics (CFD), Molecular Dynamics

AI/ML

- Deep Learning – Object detection, object recognition, image segmentation
- Large Language Models (LLM) – ChatGPT, Llama

CPU v/s GPU


Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
L2 Cache
L2 Cache
L3 Cache
DRAM
CPU
L2 Cache
DRAM
GPU

CPU v/s GPU

Central Processing Unit – CPU

- Less number of cores (~100)
- Low compute density
- Low latency
- Higher clock speed
- Powerful cores
- Serial instruction processing
- Handful of operations at once
- Suitable for serial instruction processing
- Versatility - Can be used for tasks such as OS or I/O etc

Graphics Processing Unit – GPU

- Large number of cores (~5000)
- High compute density
- High throughput
- Lower clock speed
- Weak cores
- Parallel instruction processing
- Thousands of operations at once
- Not suitable for serial instruction processing
- Not Versatile – Cannot be used for tasks such as OS or I/O etc

GPU as Co-processor

- GPU as Compute device
  - Has its own DRAM
  - Can run multiple threads in parallel
- Application runs on host
- The compute intensive, data-parallel part is sent to GPU
  - Written as C functions called kernel
  - The kernel is executed on device simultaneously by multiple threads

Heterogeneous Computing

Serial Code

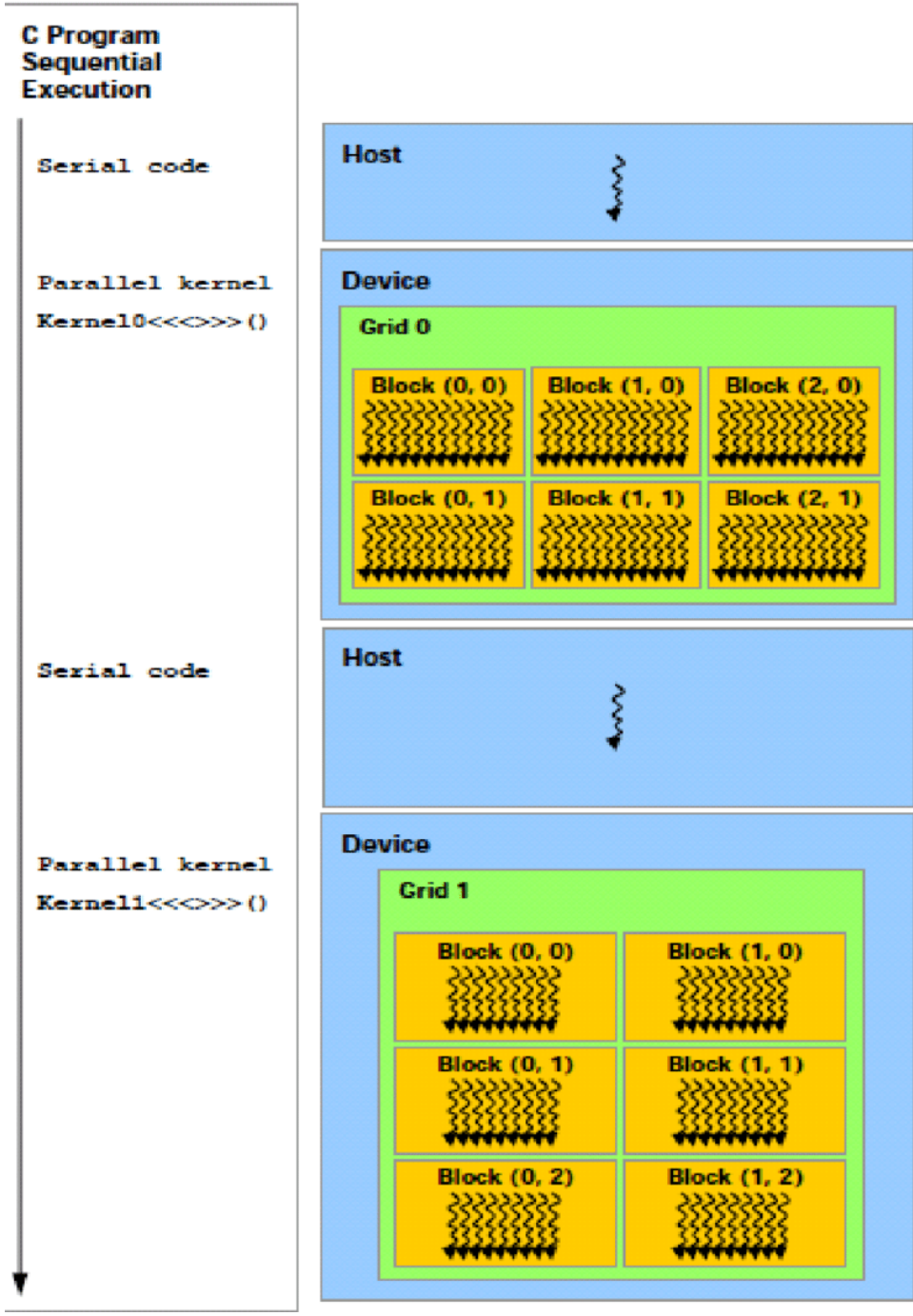
- Executed on CPU
- Host Code

Parallel Code

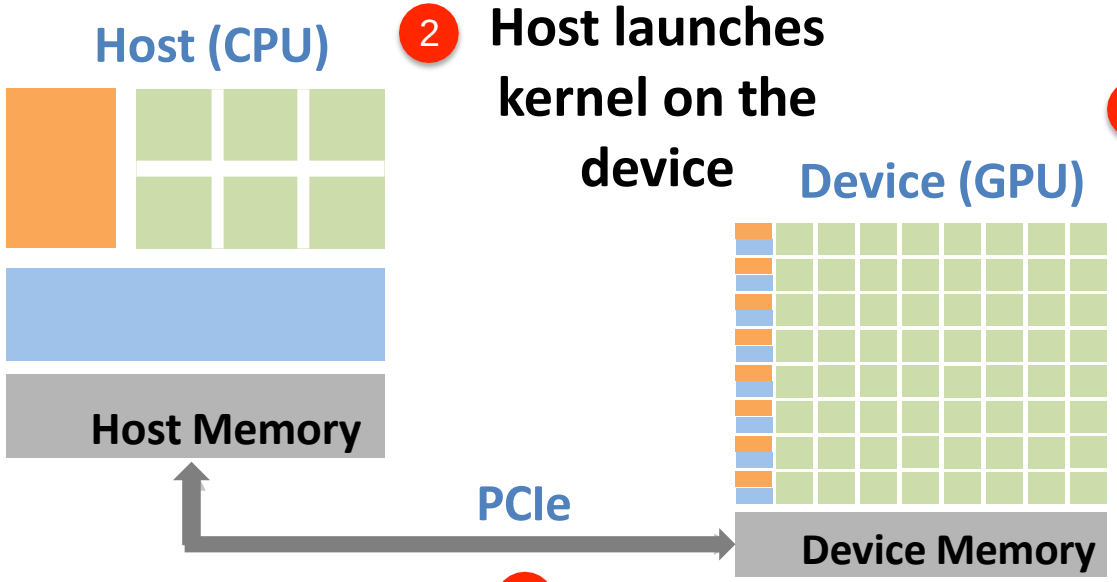
- Executed on GPU
- Device Code
- Kernel

CPU – Host

GPU – Co-processor

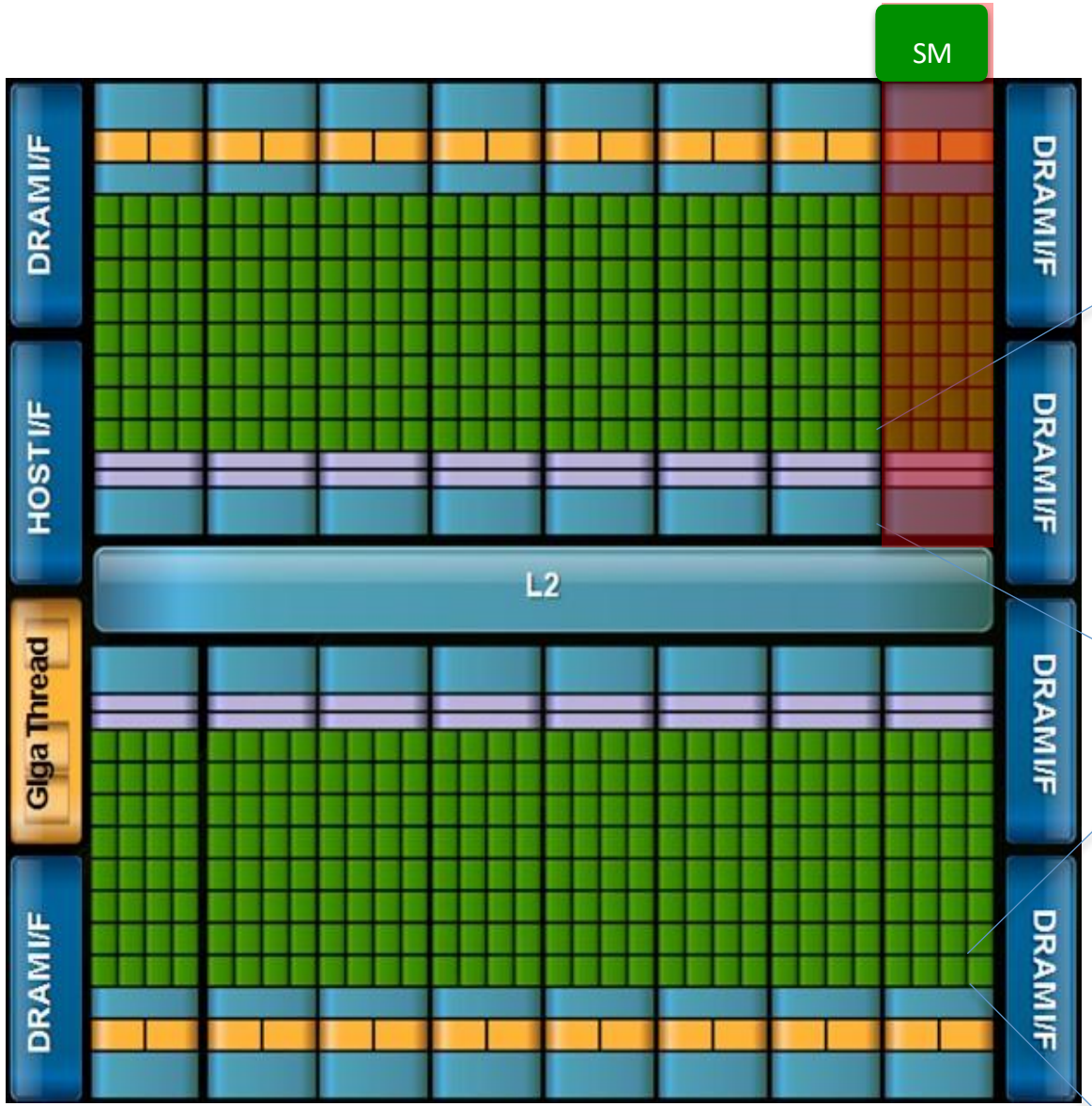


Typical Execution



1. Data is copied from the host memory to the device memory via PCIe Bus
2. Host launches kernel on the device
3. The kernel is executed by multiple threads concurrently
4. The data within the device is accessed by threads through memory hierarchy
5. The results are moved back to the device memory and are transferred back to the host via PCIe bus

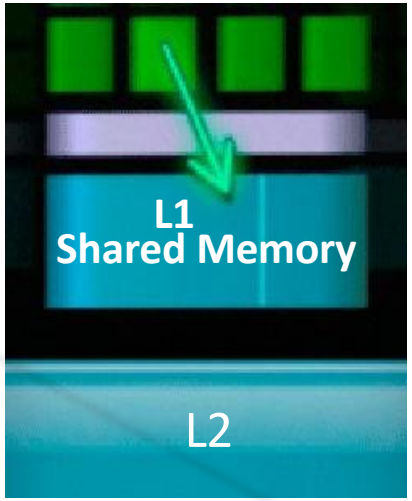
GPU Architecture



NVIDIA FERMI

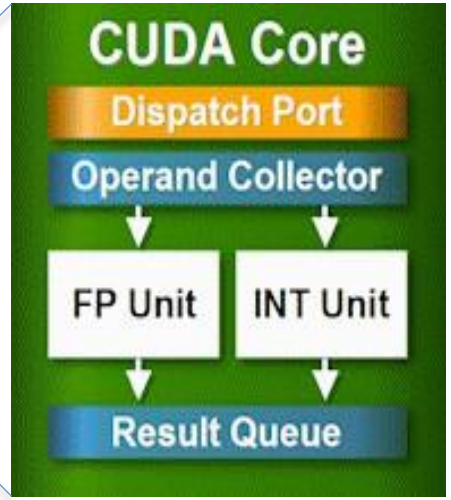
16 Stream Multiprocessors (SM) 512 CUDA cores (32/SM)
IEEE 754--2008 floating point (DP and SP)
6 GB GDDR5 DRAM (Global Memory)
ECC Memory support

Two DMA interface



Reconfigurable L1 Cache and Shared Memory
48 KB / 16 KB

L2 Cache 768 KB

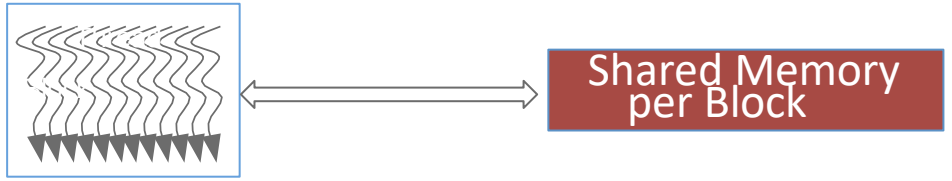


Load/Store address width 64 bits. Can calculate addresses of 16 threads per clock.

Memory Hierarchy

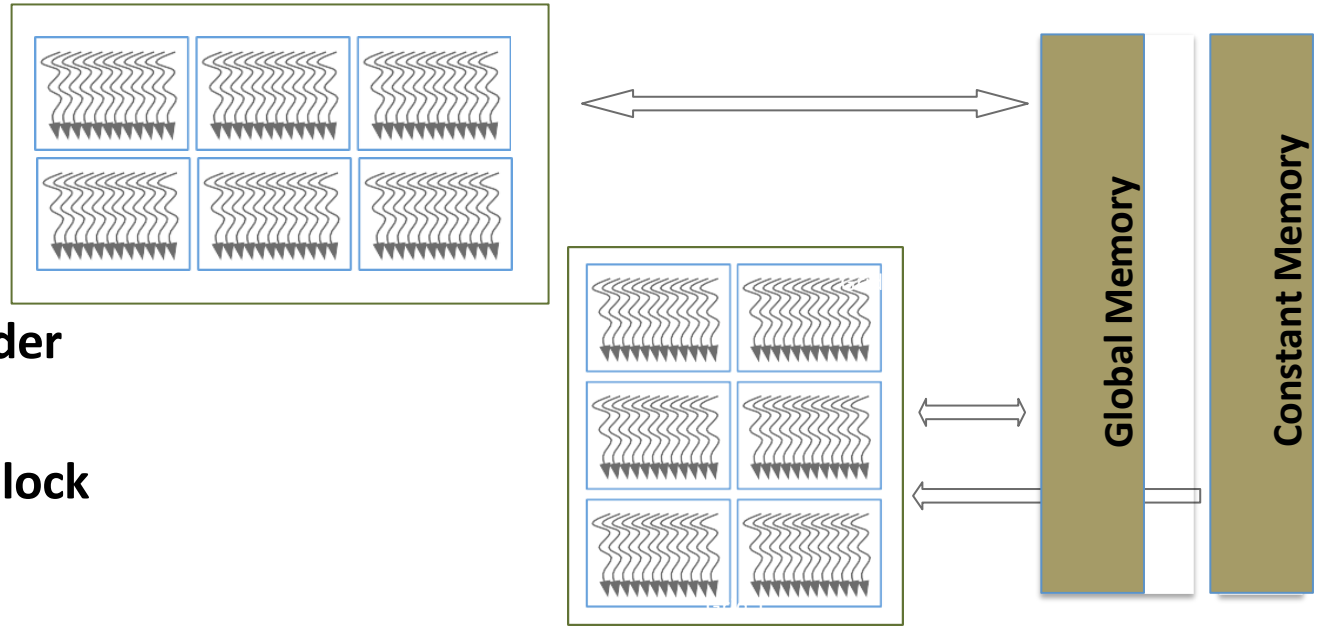
Private memory
Visible only to the thread

Shared memory
Visible to all the threads in a block



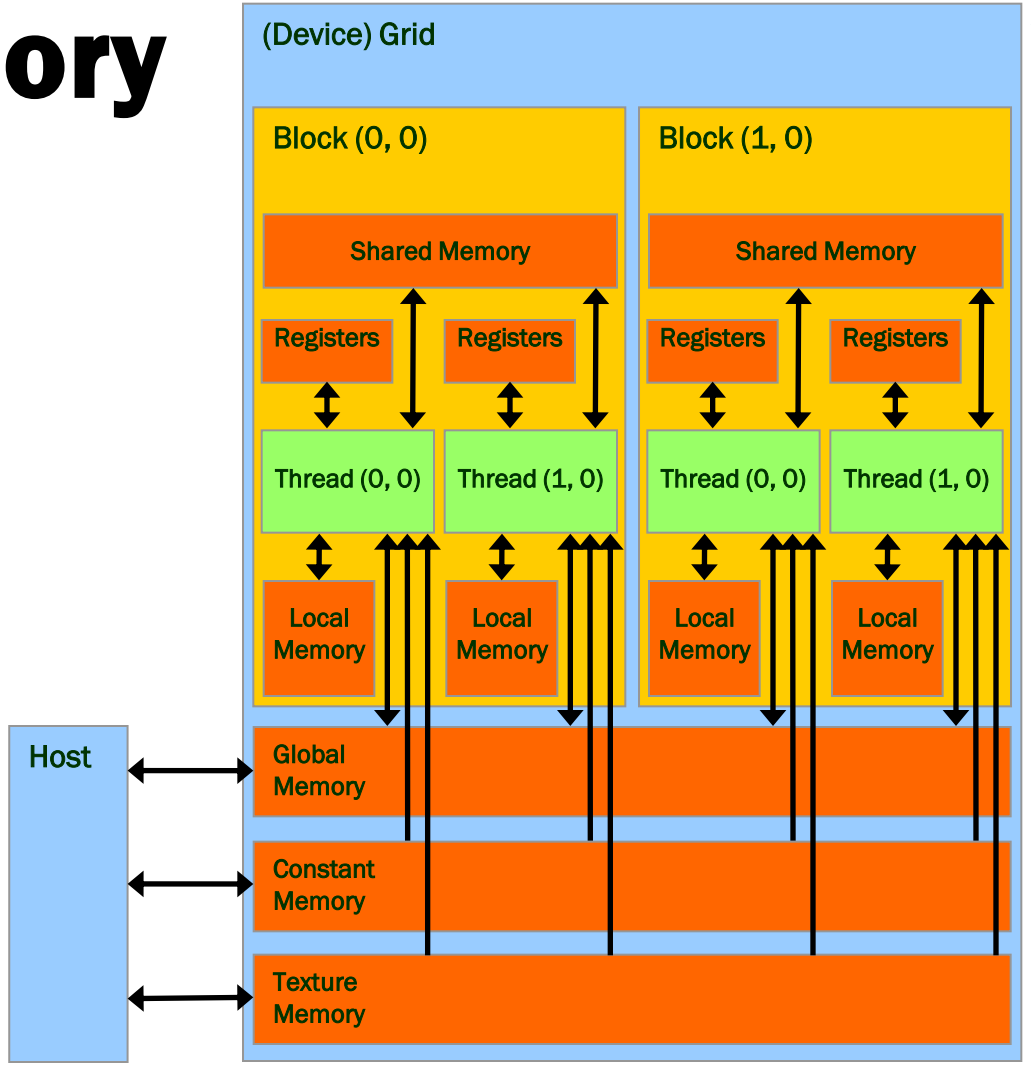
Global memory
Visible to all the threads
Visible to host
Accessible to multiple kernels
Data is stored in row major order

Constant memory (Read Only)
Visible to all the threads in a block

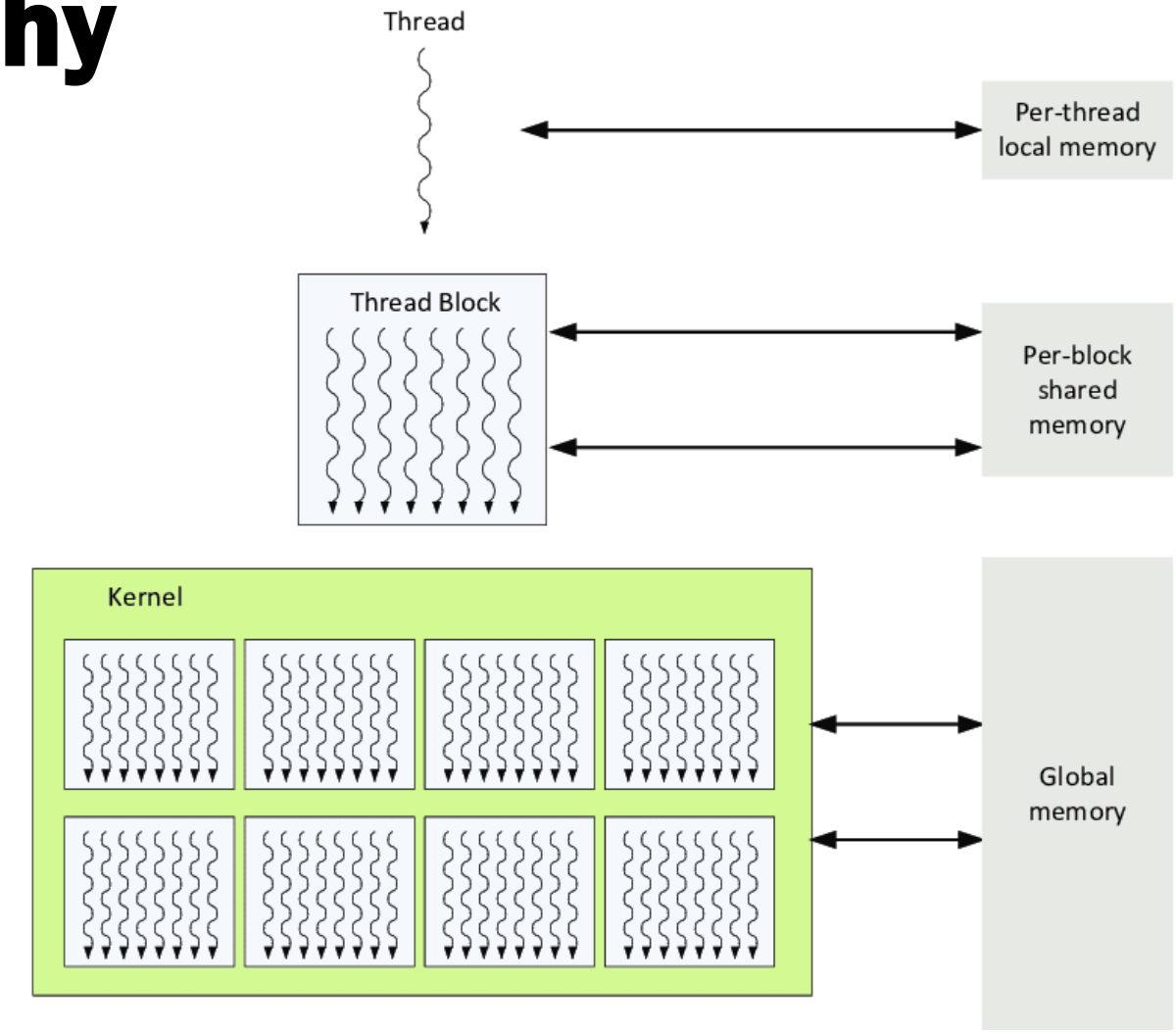


Host Accessible Memory

- Host have access to
  - Global memory
  - Constant memory
  - Texture memory



Thread Hierarchy

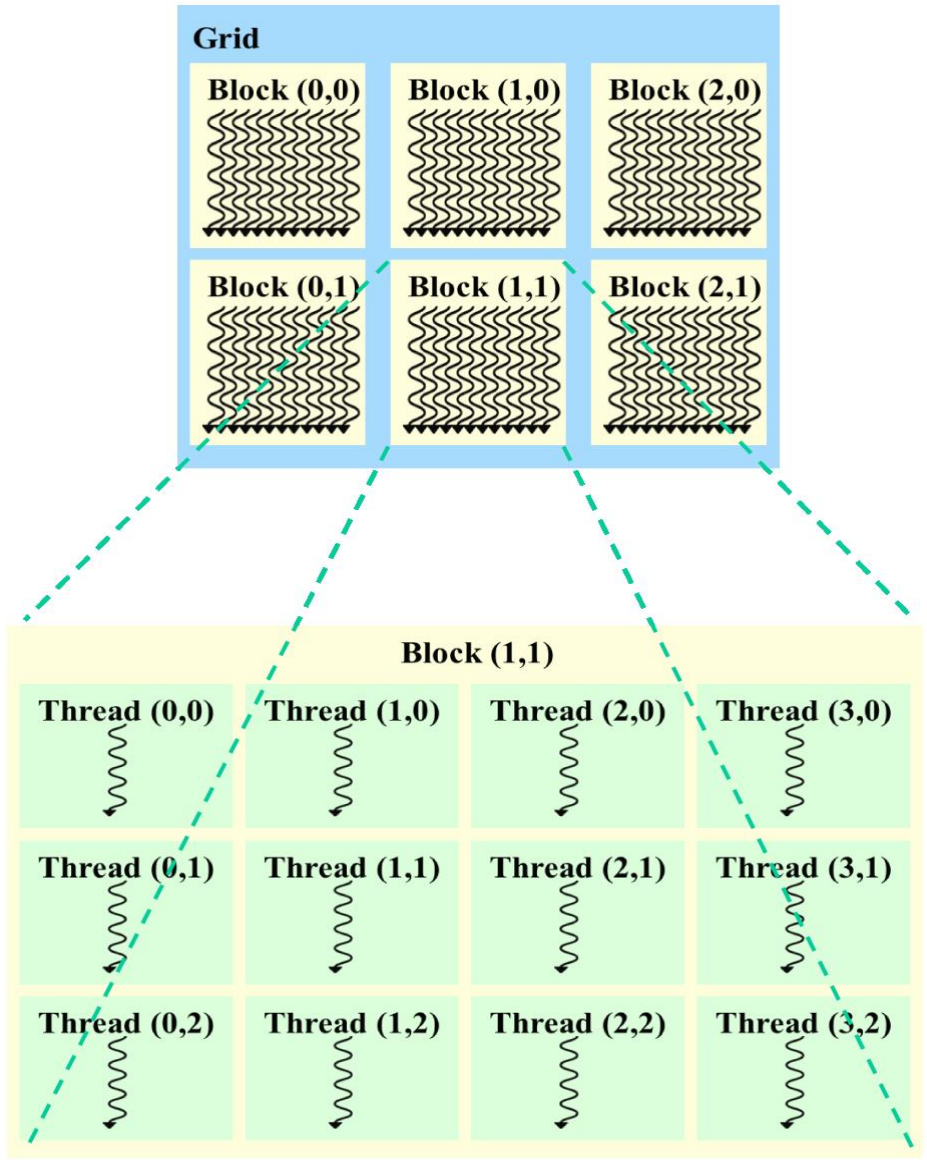


Thread Hierarchy

Threads

Thread Block

Grid of Thread Blocks



Questions?

THANK YOU