Introduction to CUDA

Week 9

Amittai Aviram – 10 November 2020 – Boston University

CS 591 A1 – Parallel Computing and Programming

The Graphics Processing Unit (GPU)
Specialized Hardware

- Designed to execute instructions typically needed for graphics
  - Massively parallel
  - Fast
  - Supporting a limited range of computations
- Used in graphics
  - Moving a line or shape over time (matrix multiplication)
  - Changing the shading or color over a region (vector addition or multiplication)
  - Applying a *texture* to a region.

CPU and GPU

High-Level Architecture

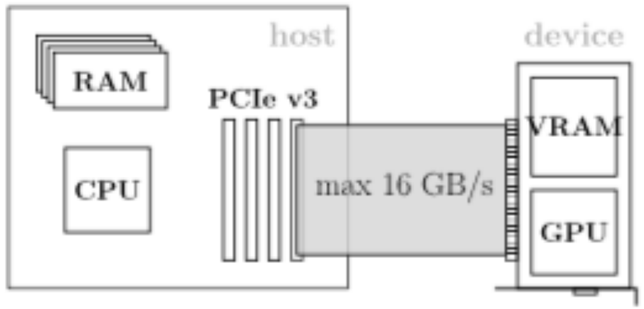


FIGURE 7.1

Schematic overview of the processors (CPU and GPU) and memory components for host and device. Memory transfers are executed over the PCIe v3 bus providing up to 16 GB/s bandwidth. Conceptually, the host and the device can be treated as independent compute platforms which communicate over an interconnection network.

GPU Components

Partial Low-Level Architecture

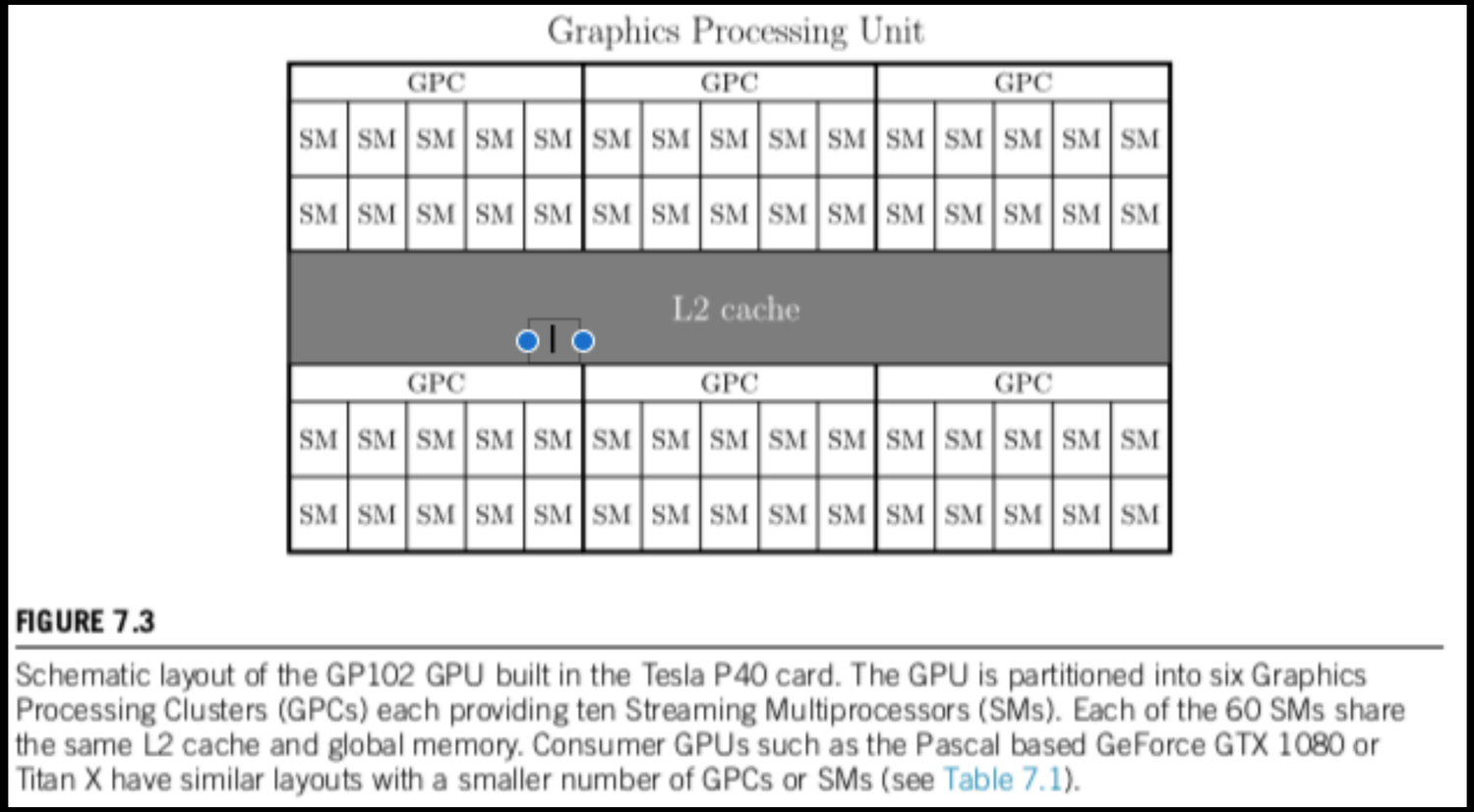


FIGURE 7.3

Schematic layout of the GP102 GPU built in the Tesla P40 card. The GPU is partitioned into six Graphics Processing Clusters (GPCs) each providing ten Streaming Multiprocessors (SMs). Each of the 60 SMs share the same L2 cache and global memory. Consumer GPUs such as the Pascal based GeForce GTX 1080 or Titan X have similar layouts with a smaller number of GPCs or SMs (see Table 7.1).

Compute Unified Device Architecture

Adaptation of the GPU for General Computing

- On NVIDIA GPUs only
- NVIDIA GPUs have their own Assembly instruction set
- CUDA exposes GPU instructions for general programming (in C++)
- Requires a special compiler or specialized compiler support
- Follows distinct processing and memory models.

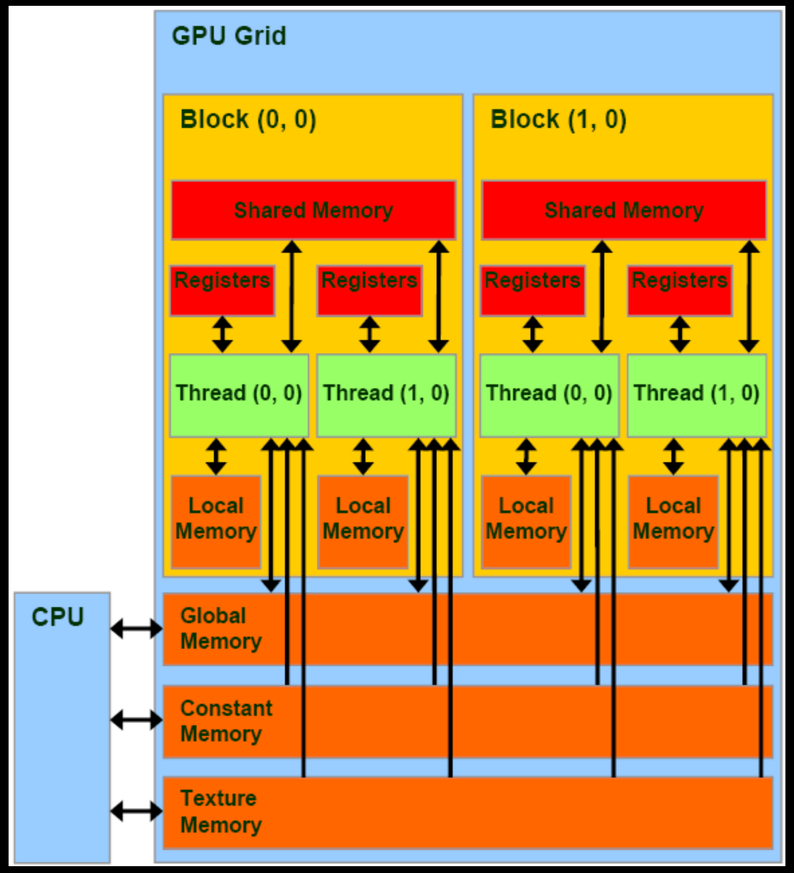
CUDA Execution Model

Terminology

- Thread—as usual
- Warp—a bundle of 32 threads
  - Mostly only visible to the compiler
  - All threads in the warp execute the same instruction (SIMD model).
- Block—a grouping of up to 1024 threads
  - These threads have access to the same shared memory.
- Grid—up to 2^{32} - 1 blocks.
- The SMs execute the blocks as they come available.

CUDA Memory Model

Choices and Levels



Programming Procedures
Host and Device

- *Kernel*: a function to run on the *device* (GPU).
- Annotations
  - __global__ — runs on the device, visible to the host and the device
  - __device__ — runs on the device, visible to the device (only)
  - __host__ (default) — runs on the host, visible to the host (only).
- A __global__ kernel cannot return a value. (It must have return type **void**.)

Programming Procedures
Movind Data

- Define objects in host memory as usual.
- Allocate GPU global memory for common objects using **cudaMalloc**.
- Copy input data from host memory to device memory using using **cudaMemcpy**.
- Call the kernel, passing to it the device-allocated objects.
  - Use **function<<<num_blocks, num_threads_per_block>>>(args)** notation.
  - Include an out-parameter (passed by pointer).
- Copy the data back from the device-allocated out-parameter to a host object.