GPUs AS PARALLEL COMPUTERS

- GPUs is shaped by the fast growing video game industry
- ability to perform a massive number of floating-point calculations per video frame
- GPUs are designed as numeric computing engines, and they will not perform well on some tasks on which CPUs are designed to perform well;
- One should expect that most applications will use both CPUs and GPUs, executing the sequential parts on the CPU and numerically intensive parts on the GPUs.
- CUDA (Compute Unified Device Architecture) programming model, introduced by NVIDIA in 2007, is designed to support joint CPU/GPU execution of an application

ARCHITECTURE OF A MODERN GPU

- CUDA-capable GPU
- It is organized into an array of highly threaded streaming multiprocessors. Two SMs form a building block;
- Each SM in has a number of streaming processors (SPs) that share control logic and instruction cache.
- (GDDR) DRAM- Memory.
- The massively parallel G80 chip has 128 Sps- (16 SMs, each with 8 SPs)
- Each SP has a multiply–add (MAD) unit and an additional multiply unit.
- Intel CPUs support 2 or 4 threads, depending on the machine model, per core. But The G80 chip supports up to 768 threads
- per SM, which sums up to about 12,000 threads for this chip.

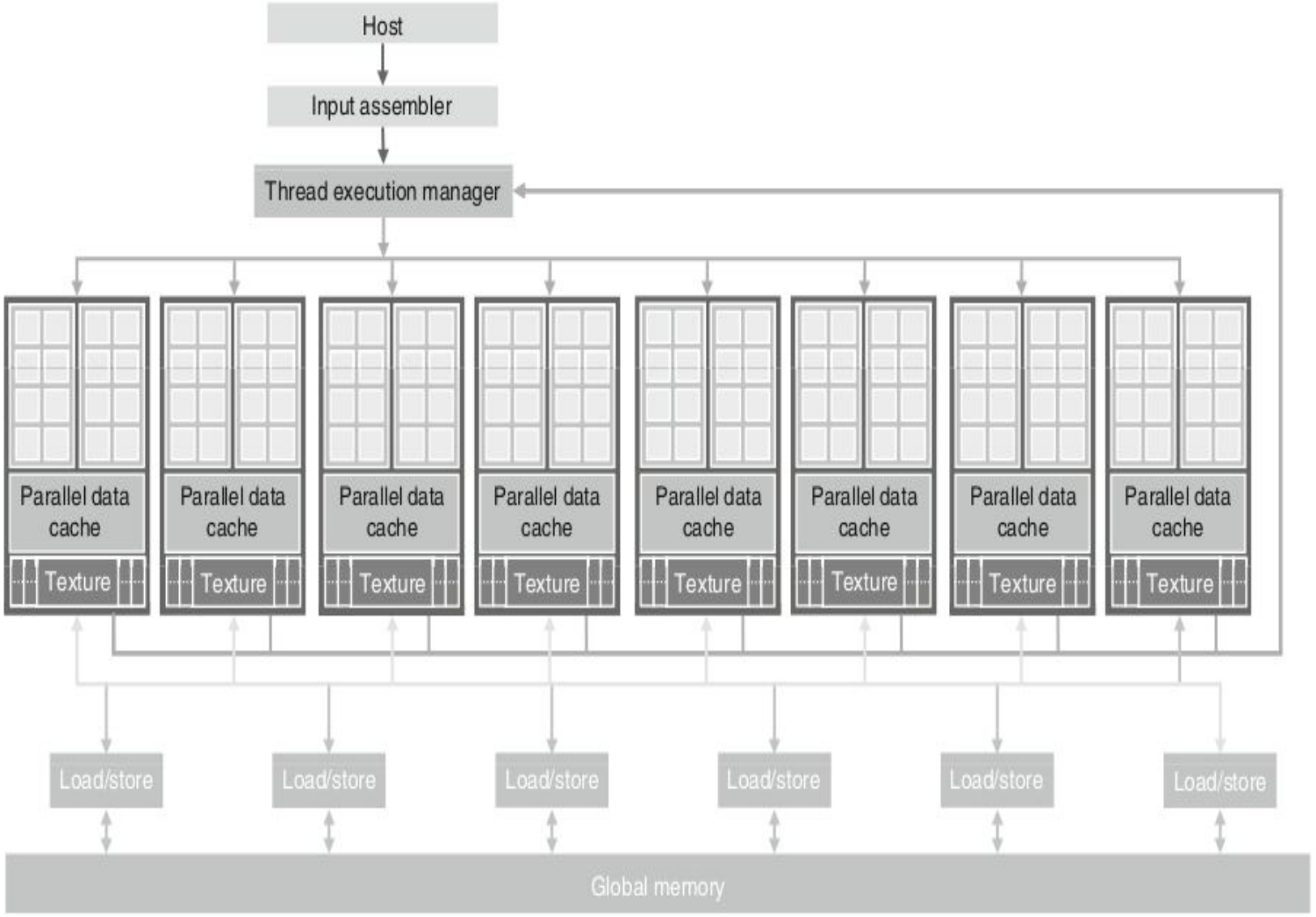

FIGURE 1.3

Architecture of a CUDA-capable GPU.

- WHY MORE SPEED OR PARALLELISM?
  - supercomputing applications, or superapplications.
  - molecular-level observation: computational model to simulate the underlying molecular activities with boundary conditions set.
  - video and audio coding and manipulation.
    - high-definition television (HDTV)
  - 3D imaging and visualization.
  - consumer electronic gaming
    - imagine driving a car in a game today; the game is, in fact, simply a prear-ranged set of scenes.

- **PARALLEL PROGRAMMING LANGUAGES AND MODELS**
  - The ones that are the most widely used are the Message Passing Interface (MPI) for scalable cluster computing and OpenMPI for shared-memory multiprocessor systems.
  - MPI has been successful in the high-performance scientific computing domain. Applications written in MPI have been known to run successfully on cluster computing systems
  - CUDA, on the other hand, provides shared memory for parallel execution in the GPU to address this difficulty.
  - CUDA achieves much higher scalability with simple, low-overhead thread management and no cache coherence hardware requirements.
  - OpenCL- Apple, Intel, AMD/ATI, and NVIDIA, have jointly developed a standardized program-ming mode
  - OpenCL programming model defines language extensions and runtime APIs to allow programmers to manage parallelism and data delivery in massively parallel proce_{ssors.}

- **WHY MORE SPEED OR PARALLELISM?**
  - supercomputing applications, or superapplications.
  - molecular-level observation: computational model to simulate the underlying molecular activities with boundary conditions set.
  - video and audio coding and manipulation.
    - high-definition television (HDTV)
  - 3D imaging and visualization.
  - consumer electronic gaming
    - imagine driving a car in a game today; the game is, in fact, simply a prear-ranged set of scenes.

History of GPU Computing

- The Era of Fixed-Function Graphics Pipelines

- The remarkable advancement of graphics hardware performance has
- been driven by the market demand for high-quality, real-time graphics in
- computer applications.

- Graphics application programming interface (API) libraries became popular

- An API is a standardized layer of software (i.e., a collection of library functions) that allows applications (such as games) to use software or hardware services and functionality.

- DirectX, Microsoft's proprietary API for media functionality

- Major API is OpenGL

Three dimensional (3D) graphics pipeline hardware evolved from large expensive systems of the early 1980s to small workstations and then PC accelerators in the mid to late 1990s.
During this period, the performance increased:

from 50 millions pixels to 1 billion pixels per second,

from 100,000 vertices to 10 million vertices per second.
This advancement was driven by market demand for high quality, real time graphics in computer applications.
The architecture evolved

From a simple pipeline for drawing wire frame diagrams.

To a parallel design of several deep parallel pipelines capable of rendering the complex interactive imagery of 3D scenes.
In the mean time, graphics processors became programmable

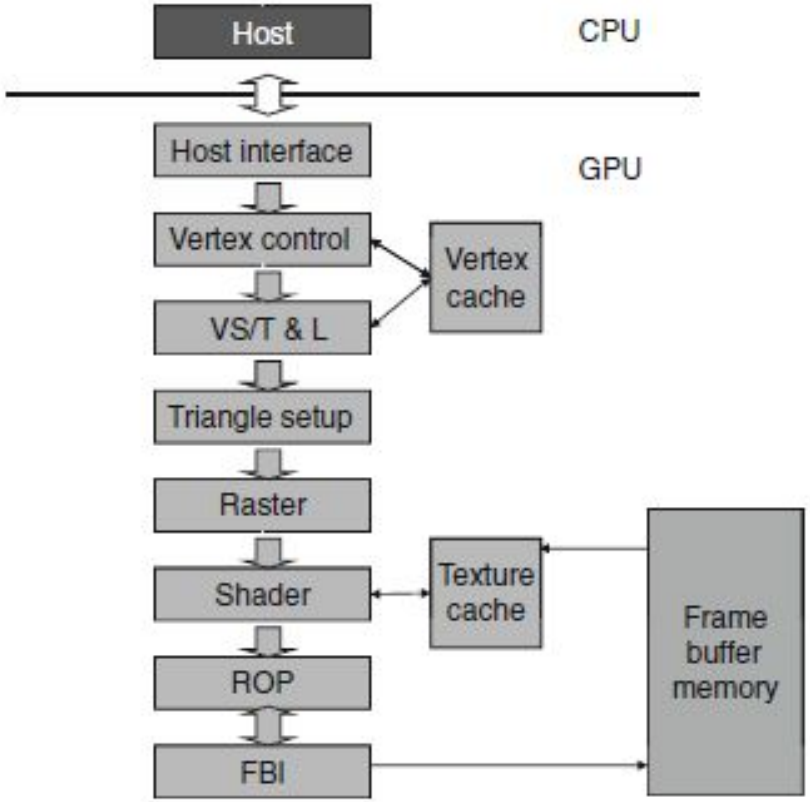

FIGURE 2.1

A fixed-function NVIDIA GeForce graphics pipeline.

Stages in the first part of the pipeline1 vertex

vertex control

This stage receives parametrized triangle data from the CPU.

The data gets converted and placed into the vertex cache.

VS/T & L (vertex shading, transform, and lighting)

The VS/T & L stage transforms vertices and assigns per-vertex values, e.g.: colors, normals, texture coordinates, tangents.

The vertex shader can assign a color to each vertex, but color is not applied to triangle pixels until later.

Triangle setup

Edge equations are used to interpolate colors and otherper-vertex data across the pixels touched by the triangle.

Raster

The raster determines which pixels are contained in each triangle.Per-vertex values necessary for shading are interpolated.

shader

The shader determines the final color of each pixel as a combined effect of interpolation of vertex colors, texture mapping, per-pixel lighting, reflections, etc.

ROP (Raster Operation)

The final raster operations blend the color of overlapping/adjacent objects for transparency and antialiasing effects.

For a given viewpoint, visible objects are determined and occluded pixels (blocked from view by other objects) are discarded.

FBI (Frame Buffer Interface)

The FBI stages manages memory reads from and writes to the display frame buffer memory

This data independence as the dominating characteristic is the key difference between the design assumption for GPUs and CPUs.
A single frame, rendered in 1/60th of a second, might have a million triangles and 6 million pixels.

Unified graphics and computing processors

Introduced in 2006, the GeForce 8800 GPU mapped the separate programmable graphics stages to an array of unified processors

High-clock-speed design made programmable GPU processor array ready for general numeric computing

Original GPGPU programming used APIs (DirectX or OpenGL): to a GPU everything is a pixel

Interpolation of vertex colors, texture mapping, per-pixel lighting mathematics, reflections.
Many effects that make the rendered images more realistic are incorporated in the shader stage.
Raster operation (ROP) stage in Figure 2.2 performs the final rasteroperations on the pixels

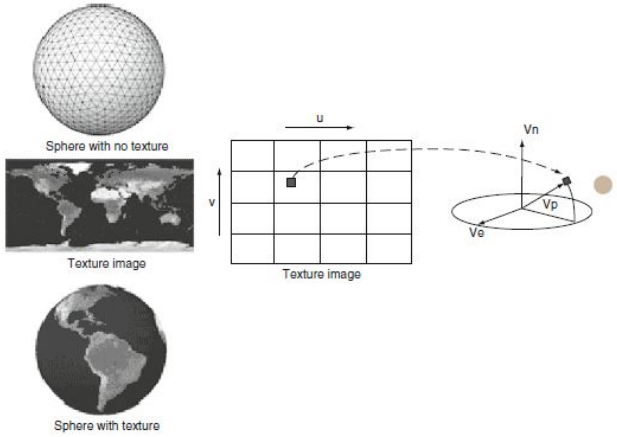

FIGURE 2.2
Texture mapping example: painting a world map texture image onto a globe object.

GPU computing

Drawbacks of the GPGPU model:

The programmer must know APIs and GPU architecture well
Programs expressed in terms of vertex coordinates, textures,shader programs, add to the complexity
Random reads and writes to memory are not supported
No double precision is limiting for scientific applications

Evolution of Programmable Real-Time Graphics

These programmable pixel shader processors were part of a general trend towards unifying the functionality of the different stages as seen by the application programmer
The GeForce 6800 and 7800 series were
built with separate processor designs dedicated to vertex and pixel processing.
The XBox 360 introduced an early unified processor GPU in 2005, allowing vertex and pixel shaders to execute on the same processor
Two particular programmable stages stand out: the vertex shader and the pixel shader.

Vertex shader programs map the positions of triangle vertices onto the screen, altering their position, color, or orientation