CUDA COMPUTE LEVELS

CUDA COMPUTE LEVELS

- CUDA supports a number of compute levels.
- A full list of the differences between each compute level can be found in the NVIDIA CUDA Programming Guide

CUDA COMPUTE LEVELS

Compute 1.0

- Compute level 1.0 is found on the older graphics cards.
  - Ex: the original 8800 Ultras and many of the 8000 series cards as well as the Tesla C/D/S870s.
- The main features lacking in compute 1.0 cards are those for atomic operations.
- Atomic operations are those where we can guarantee a complete operation without any other thread interrupting.
- The hardware implements a barrier point at the entry of the atomic function and guarantees the completion of the operation (add, sub, min, max, logical and, or, xor, etc.) as one operation.
- Compute 1.0 cards are effectively now obsolete.

CUDA COMPUTE LEVELS

Compute 1.1

- Compute level 1.1 is found in many of the later shipping 9000 series cards, such as the 9800 GTX, which were extremely popular..
- One major change brought in with compute 1.1, is overlapped data transfer and kernel execution.
- The SDK call to cudaGetDeviceProperties() returns the deviceOverlap property, which defines if this functionality is available.
- This allows important optimization called **double buffering**.

CUDA COMPUTE LEVELS

Compute 1.1

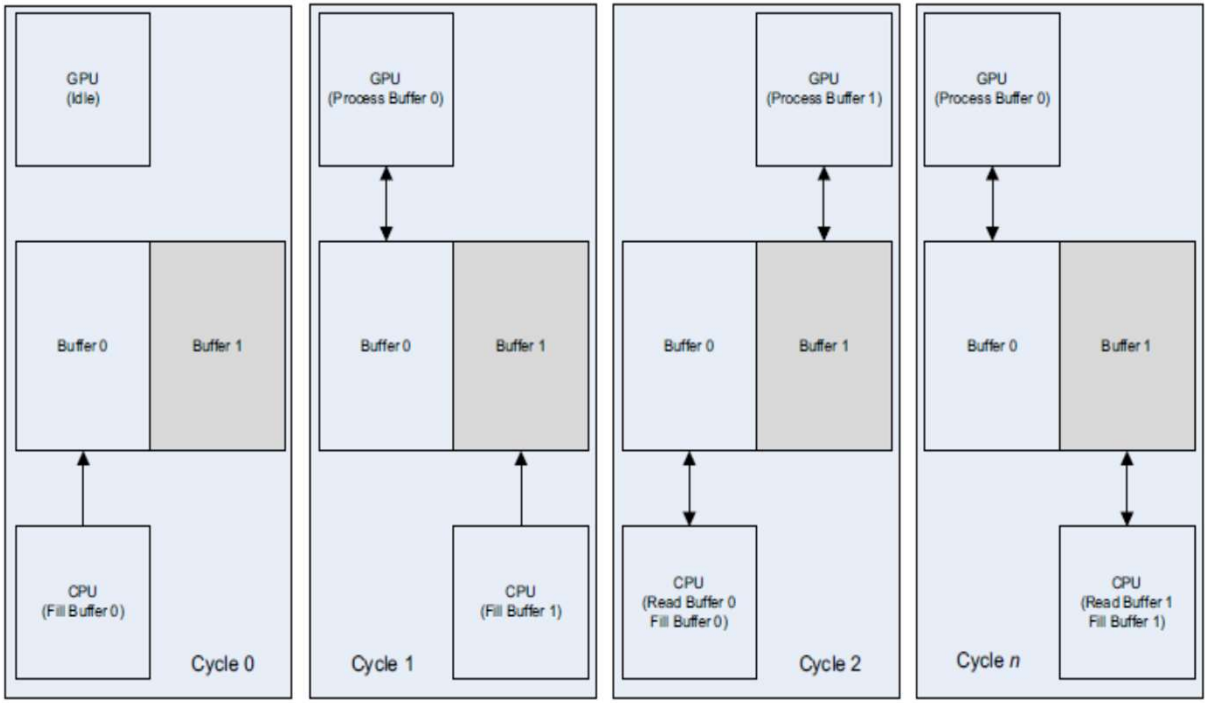

Fig: Double buffering with a single GPU

CUDA COMPUTE LEVELS

Compute 1.1

- Cycle 0: Having allocated two areas of memory in the GPU memory space, the CPU fills the first buffer.
- Cycle 1:
- a)The CPU then invokes a CUDA kernel (a GPU task) on the GPU, which returns immediately to the CPU.
- b)The CPU then fetches the next data packet, from a disk, the network, or wherever.
- c)Meanwhile, the GPU is processing away in the background on the data packet provided.
- When the CPU is ready, it starts filling the other buffer.

CUDA COMPUTE LEVELS

Compute 1.1

- Cycle 2: When the CPU is done filling the buffer, it invokes a kernel to process buffer 1.
- It then checks if the kernel from cycle 1, which was processing buffer 0, has completed.
- If not, it waits until this kernel has finished and then fetches the data from buffer 0 and then loads the next data block into the same buffer.
- During this time the kernel kicked off at the start of the cycle is processing data on the GPU in buffer 1.
- Cycle N:We then repeat cycle 2, alternating between which buffer we read and write to on the CPU with the buffer being processed on the GPU.

CUDA COMPUTE LEVELS

Compute 1.2

- Compute 1.2 devices appeared with the low-end GT200 series hardware. These were the initial GTX260 and GTX280 cards.
- With the GT200 series hardware, NVIDIA approximately doubled the number of CUDA core processors on a single card.
- NVIDIA increased the number of concurrent warps a multiprocessor could execute from 24 to 32.
- Warps are blocks of code that execute within a multiprocessor
- Issues with restrictions on coalesced access to the global memory bank conflicts in the shared memory found in compute 1.0 and compute 1.1 devices were greatly reduced.
- This make the GT200 series hardware far easier to program and it greatly improved the performance.

CUDA COMPUTE LEVELS

Compute 1.3

- The compute 1.3 devices were introduced with the move from GT200 to the GT200 a/b revisions of the hardware. Almost all higher-end cards from this era were compute 1.3 compatible.
- The major change that occurs with compute 1.3 hardware is the introduction of support for limited double-precision calculations.
- GPUs are primarily aimed at graphics and here there is a huge need for fast single-precision calculations, but limited need for double-precision ones.
- Typically, you see an order of magnitude drop in performance using double-precision as opposed to single-precision floating-point operations,.

CUDA COMPUTE LEVELS

Compute 2.0

- Compute 2.0 devices saw the switch to Fermi hardware.
- The original guide for tuning applications for the Fermi architecture can be found on the NVIDIA website at **http://developer.nvidia.com/cuda/**
- Some of the main changes in compute 2.x hardware are as follows:
  - Introduction of 16 K to 48 K of L1 cache memory on each SP.
  - Introduction of a shared L2 cache for all SMs.
  - Support in Tesla-based devices for ECC (Error Correcting Code)-based memory checking and error correction.
  - Support in Tesla-based devices for dual-copy engines.
  - Extension in size of the shared memory from 16 K per SM up to 48 K per SM.
  - For optimum coalescing of data, it must be 128-byte aligned.
  - The number of shared memory banks increased from 16 to 32.

CUDA COMPUTE LEVELS

Compute 2.0

- First, the introduction of the L1 cache. It is the fastest cache type. Compute 1.x hardware has no cache, except for the texture and constant memory caches.
- The introduction of a cache makes it much easier for many programmers to write programs that work well on GPU hardware
- To exploit the cache, the application either needs to have a sequential memory pattern or have at least some data reuse.
- The L2 cache is up to 768 K in size on Fermi and, importantly, is a unified cache.
- It is shared and provides a consistent view for all the SMs. This allows for much faster interblock communication through global atomic operations.

CUDA COMPUTE LEVELS

Compute 2.0

- Support for ECC memory is a must for data centres.
- ECC memory provides for automatic error detection and correction.
- ECC detects and corrects single-bit upset conditions that you may find in large data centers.
- The increase of shared memory banks from 16 to 32 bits.
- This is a major benefit over the previous generations.
- It allows each thread of the current warp (32 threads) to write to exactly one bank of 32 bits in the shared memory without causing a shared bank conflict.

CUDA COMPUTE LEVELS

Compute 2.1

- Compute 2.1 is seen on certain devices aimed specifically at the games market, such as the GTX460 and GTX560
- 48 CUDA cores per SM instead of the usual 32 per SM.
- Eight single-precision, special-function units for transcendental per SM instead of the usual four.
- Dual-warp dispatcher instead of the usual single-warp dispatcher.
- On compute 2.1 hardware, instead of the usual two instruction dispatchers per two clock cycles, we now have four.
- In the hardware, there are three banks of 16 CUDA cores, 48 CUDA cores in total, instead of the usual.