COMP 605: Introduction to Parallel Computing
Lecture : GPU Architecture

Mary Thomas

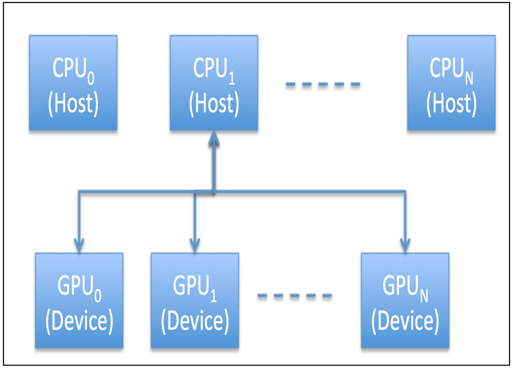
Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 04/10/17
Last Update: 04/10/17

Table of Contents

GPU Architecture

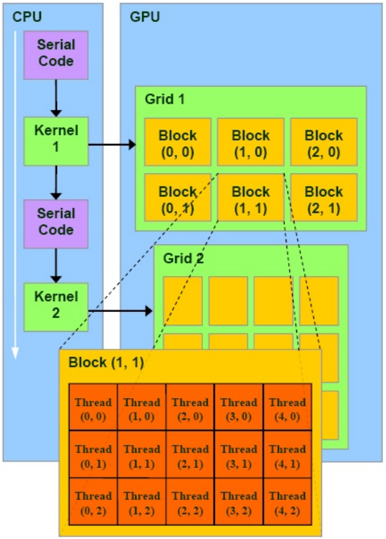
GPU Computing: Simplified Hardware View



- CPU = Central Processing Units
  - single or miltiple processing units (cores)
  - standalone or integrated into clusters
  - designed to run processes, supports threads
- GPU = Graphical Processing Units
  - Usually attached to a host CPU
  - Developed for games (think Sony, PS3), and visualization (OpenGl, think Pixar)
  - Designed to run lightweight threads, may have multiple PE's
  - Accessible via specialized libraries, compiler directives (OpenACC), and extensions to languages (C, C++ and Fortran).
- CUDA (Compute Unified Device Architecture)
  - a parallel computing platform and programming model created by NVIDIA.
  - extension of C programming language

GPU Architecture

- GPU is a highly threaded coprocessor to the host CPU and associated memory
- Kernels are sections of the application that are run on the GPU by a thread.
- A thread block is a batch of threads that can cooperate with each other by:
  - Sharing data through shared memory
  - Synchronizing their execution
  - Each block is organized as 3D array of threads: (*blockDim.x*, *blockDim.y*, and *blockDim.z*)
- Threads within a thread block must:
  - execute the same kernel
  - share data, so they must be issued to the same processor
- A grid is a collection of blocks:
  - A Grid is organized as a 2D array of blocks: (*gridDim.x* and *gridDim.y*)



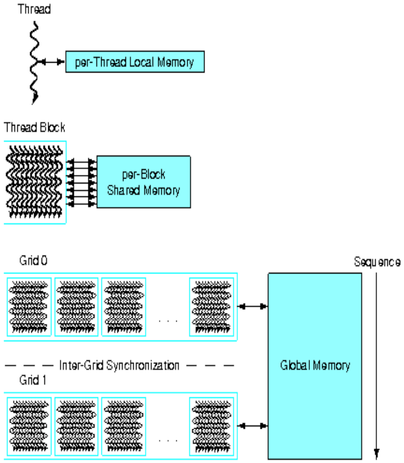
Source: http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview

GPU: Kernel

- Kernels are not full applications:
  - they are the parallel sections or critcal blocks
- They are executed by a grid of unordered thread blocks:
  - up to 512 threads per block.
  - Thread blocks start at the same instruction address, execute in parallel
  - Blocks can have different endpoints (divergence) but these are limited
  - Communicate through shared memory and synchronization barriers.
  - Must be assigned to the same processor

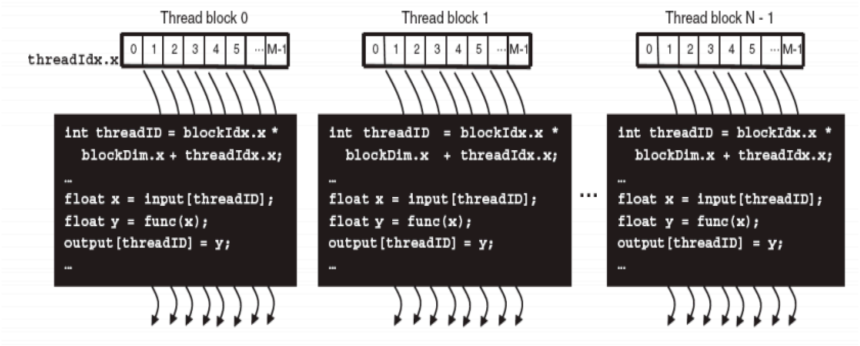
GPU: Threads

- A thread of execution is the smallest sequence of programmed instructions that can be managed independently by an operating system scheduler.
- A thread is a light-weight process.
- In most cases, a thread is contained inside a process.
- Multithreading generally occurs by time-division multiplexing (as in multitasking)
- Multiprocessor (including multi-core system): threads or tasks run at the same time - each processor or core runs a particular thread or task.



Source: `http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/gpus.pdf`

NVIDIA Thread Calculations



```
int threadID = blockIdx.x *
  blockDim.x + threadIdx.x;
…
float x = input[threadID];
float y = func(x);
output[threadID] = y;
…
```

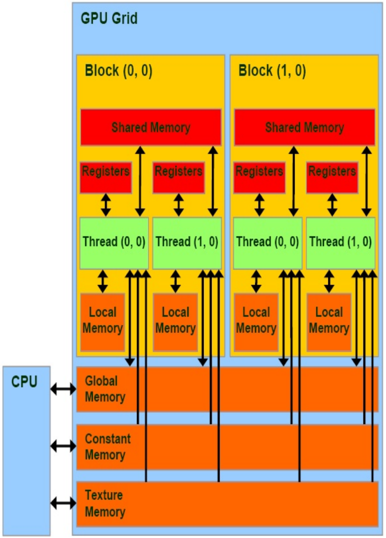
- Thread ID is unique within a block
- Bock ID is unique
- Can make unique ID for each thread per kernel using Thread and Block IDs.

GPU: Grid

- A collection of blocks that can (but are not required to) execute in parallel.
- Theres no synchronization at all between the blocks.
- Number of [concurrent] grids on a GPU:
  1 for GPU Cores < 2.0
  16 for $2.0 <= CC <= 3.0$
  32 for CC $= 3.5$
  . . .
  Need to use right API in order to avoid serialization.
- There is an API for querying the GPU system.

The GPU Memory Model

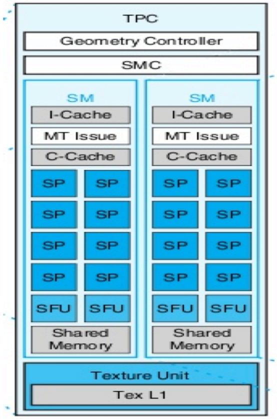
- Red is fast on-chip, orange is DRAM
- Register file & local memory are private for each thread
- Shared memory is used for communication between threads (appx same latency as regs)
- DRAM, Readonly:
  - Constant memory (64KB) used for random accesses (such as instructions)
  - Texture memory (large) and has two dimensional locality
- Global Memory: visible to an entire grid, can be arbitrarily written to and read from by the GPU or the CPU.



Source: NVIDIA

NVIDIA Hardware: GEForce 8800

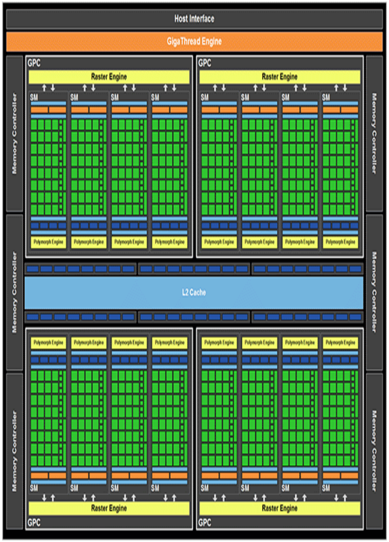
- Streaming Multiprocessors (SMs, also called nodes)
- 8 Stream Processors (SPs) (or cores): primary thread processor
- has 1000's of registers that can be partitioned among threads of execution
- Multiple caches:
    - shared memory for fast data interchange between threads,
    - constant cache for fast broadcast of reads from constant memory,
    - texture cache to aggregate bandwidth from texture memory,
    - L1 cache: reduce latency to memory
- warp schedulers: switch contexts between threads and instructions to warps;
- Execution cores:
    - Integer and floating point ops
    - Special Function Units (SFUs)



Source: `http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/gpus.pdf`

NVIDIA GPU GF100 High-Level Block Diagram (2010)

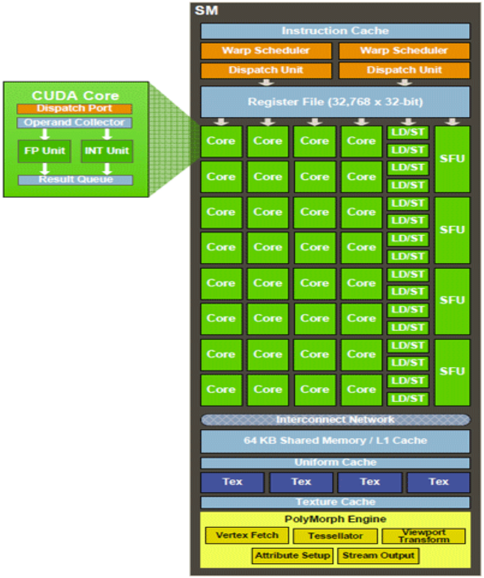
- CPU is the host
- GPU is the device
- The GF100 has 4 Graphics Processing Clusters (GPCs): laid out in (2x2) arrangement (also called "Raster Engine").
- Each GPC has 4 Streaming Multiprocessors, (SMs): NVIDIAs' term for multiprocessor (also called "Polymorph Engines").
  Arranged in 1x4 layout.
  Total number of SMs $= 4 * 4 = 16$
- Each SM has a block of Stream Processors (SPs) or Cores– also called execution units.
  Arranged in 8x4 layout.
  Total number of SPs on each SM $= 8 * 4 = 32$
- Total number of cores on the GPU
  $$\#Cores = \#SPs/SM \times \#SMs/GPC \times \#GPCs$$
  $$= 32 \times 4 \times 4 = 512$$



Source: `http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview`

NVIDIA GF100 SM Block Diagram

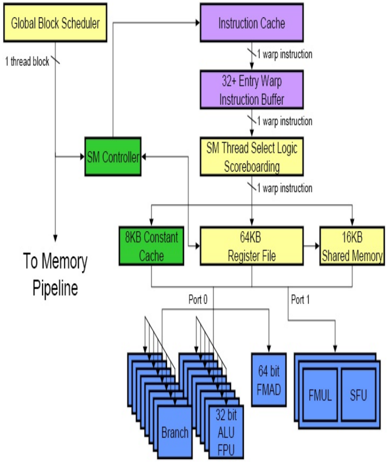
- each SM block in each GPC is comprised of 32 cores
- 48/16KB of shared memory (3 x that of GT200),
- 16/48KB of L1 (there is no L1 cache on GT200),



Source: http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview

NVIDIA GT200 SM Arch (2008)

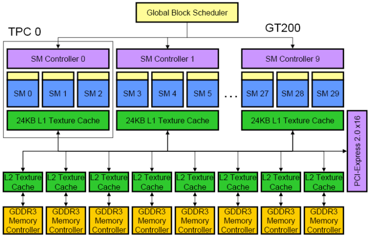
- highly threaded single-issue processor with SIMD/SIMT (single instruction multiple thread)
- 8 functional units
- Each SM can execute up to 8 thread blocks concurrently and a total of 1024 threads concurrently
- warp: a group of threads managed by SM thread scheduler
- Single Instruction, Multiple Thread (SIMT) programming model

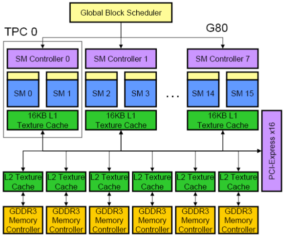


Source: `http://www.realworldtech.com/gt200`

GPU Global Scheduler (work distribution unit)

- Manages coarse grained parallelism at thread block level
- At kernel startup, information for grid sent from CPU (host) to GPU (device)
- Scheduler reads information and issues thread blocks to streaming multiprocessors (SM)
- Issues thread blocks in a round-robin fashion to SMs
- Uniformly distribute threads to SMs
- Key distribution factors:
  - kernel demand for threads per block
  - shared memory per block
  - registers per thread
  - thread and block state requirements
  - current availability resources in SM





http://www.realworldtech.com/gt200/6/