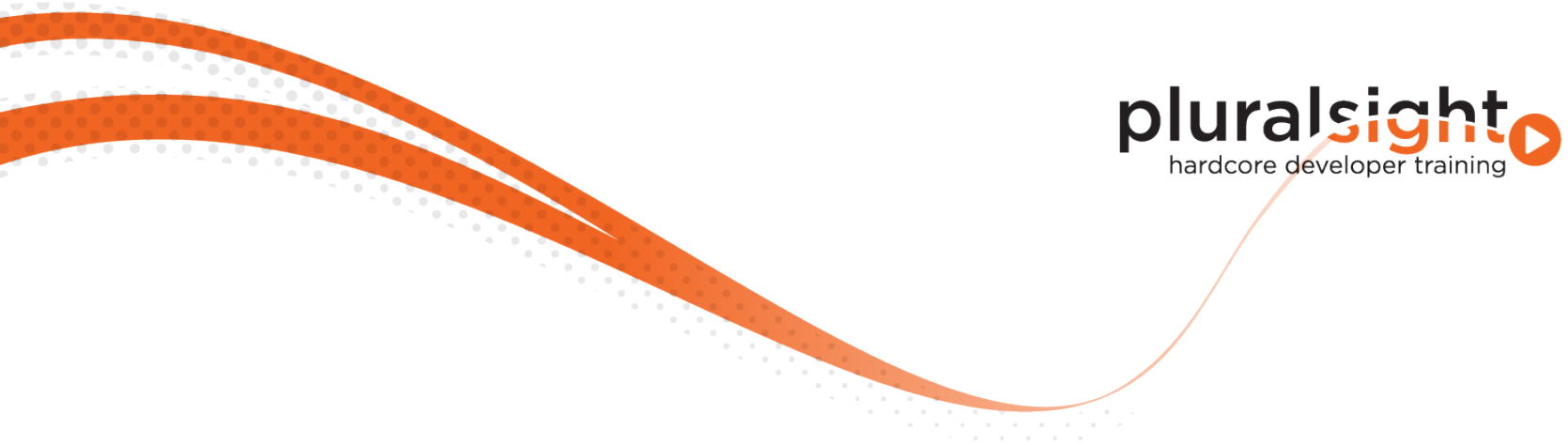
Parallel Computing with CUDA

Dmitri Nesteruk
@dnesteruk



GPU Architecture Overview

Dmitri Nesteruk
@dnesteruk

Course Outline

- GPU Architecture Overview (this module)
- Tools of the Trade
- Introduction to CUDA C
- Patterns of Parallel Computing
- Thread Cooperation and Synchronization
- The Many Types of Memory
- Atomic Operations
- Events and Streams
- CUDA in Advanced Scenarios

Note: to follow this course, you require a CUDA-capable GPU.

History of GPU Computation

No GPU

- CPU handled graphic output

Dedicated GPU

- Separate graphics card (PCI, AGP)

Programmable GPU

- Shaders

Shaders

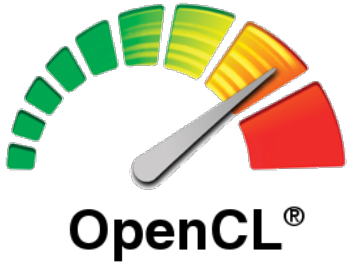
- **Small program that runs on the GPU**
- **Types**
  - Vertex (used to calculate location of a vertex)
  - Pixel (used to calculate the color components of a single pixel)
- **Shader languages**
  - High Level Shader Language (HLSL, Microsoft DirectX)
  - OpenGL Shading Language (GLSL, OpenGL)
  - Both are C-like
  - Both are intended for graphics (i.e., not general-purpose)
- **Pixel shaders used for math**
  - Convert data to texture
  - Run texture through pixel shader
  - Get result texture and convert to data

Why GPGPU?

- **General-Purpose Computation on GPUs**
- **Highly parallel architecture**
  - Lots of concurrent threads of execution (SIMT)
  - Higher throughput compared to CPUs
    - Even taking into account many cores, hypethreading, SIMD
  - Thus more FLOPS (floating-point operations per second)
- **Commodity hardware**
  - Commonly available (mainly used by gamers)
  - Relatively cheap compared to custom solutions (e.g., FPGAs)
- **Sensible programming model**
  - Manufacturers realized GPGPU market potential
  - Graphics made optional
  - NVIDIA offers dedicated GPU platform “Tesla”
    - No output connections

GPGPU Frameworks

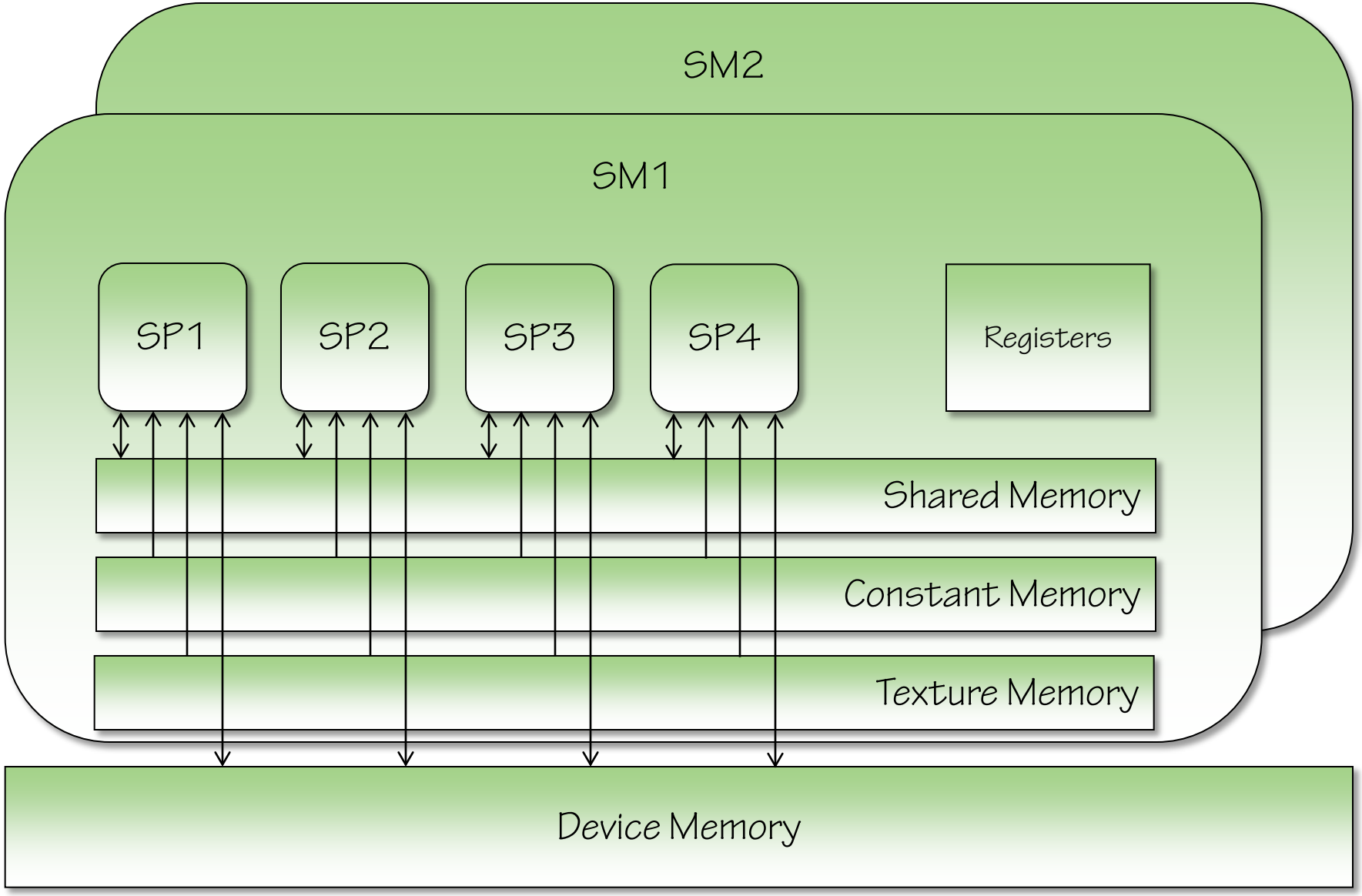
- **Compute Unified Driver Architecture (CUDA)**
  - Developed by NVIDIA Corporation
  - Extensions to programming languages (C/C++)
  - Wrappers for other languages/platforms (e.g., FORTRAN, PyCUDA, MATLAB)
- **Open Computing Language (OpenCL)**
  - Supported by many manufacturers (inc. NVIDIA)
  - The high-level way to perform computation on ATI devices
- **C++ Accelerated Massive Programming (AMP)**
  - C++ superset
  - A standard by Microsoft, part of MSVC++
  - Supports both ATI and NVIDIA
- **Other frameworks and cross-compilers**
  - Alea, Aparapi, Brook, etc.



Graphics Processor Architecture

- **Warning: NVIDIA terminology ahead**
- **Streaming Multiprocessor (SM)**
  - Contains several CUDA cores
  - Can have >1 SM on card
- **CUDA Core (a.k.a. Streaming Processor, Shader Unit)**
  - # of cores per SM tied to compute capability
- **Different types of memory**
  - Means of access
  - Performance characteristics

Graphics Processor Architecture



Compute Capability

- **A number indicating what the card can do**
  - Current range: 1.0, 1.x, 2.x, 3.0, 3.5
- **Affects both hardware and API support**
  - Number of CUDA cores per SM
  - Max # of 32-bit registers per SM
  - Max # of instructions per kernel
  - Support for double-precision ops
  - … and many other parameters
- **Higher is better**
  - See http://en.wikipedia.org/wiki/CUDA

Feature support (unlisted features are supported for all compute capabilities)	Compute capability (version)						
	1.0	1.1	1.2	1.3	2.x	3.0	3.5
Integer atomic functions operating on 32-bit words in global memory	No	Yes					
atomicExch() operating on 32-bit floating point values in global memory							
Integer atomic functions operating on 32-bit words in shared memory	No	Yes					
atomicExch() operating on 32-bit floating point values in shared memory							
Integer atomic functions operating on 64-bit words in global memory							
Warp vote functions							
Double-precision floating-point operations	No			Yes			
Atomic functions operating on 64-bit integer values in shared memory	No				Yes		
Floating-point atomic addition operating on 32-bit words in global and shared memory							
_ballot()							
_threadfence_system()							
__syncthreads_count(), __syncthreads_and(), __syncthreads_or()							
Surface functions							
3D grid of thread block							
Warp shuffle functions	No					Yes	
Funnel shift	No						Yes
Dynamic parallelism							

Choosing a Graphics Card

- **Look at performance specs (peak flops)**
  - Pay attention to single vs. double
  - E.g. http://www.nvidia.com/object/tesla-servers.html
- **Number of GPUs**
- **Compute capability/architecture**
  - COTS cheaper than dedicated
- **Memory size**
- **Ensure PSU/motherboard is good enough to handle the card(s)**
- **Can have >1 graphics card**
  - YMMV (PCI saturation)
- **Can mix architectures (e.g. NVIDIA+ATI)**