

OPENCL

Episode I - Introduction to OpenCL

David W. Gohara, Ph.D.

Center for Computational Biology

Washington University School of Medicine, St. Louis

email: sdg0919@gmail.com

WHAT IS OPENCL?

- **OpenCL - Open Computing Language**
 - Open Specification
 - Proposed by Apple
 - Specification developed by a number of companies
 - Specification is maintained by the Khronos Group



WHAT IS OPENCL?

- OpenCL needs to be implemented by “someone”
 - Similar to OpenGL
- OpenCL as an open standard means it can be adopted by “anyone”
- For an implementation to be compliant it must conform to the specification

WHY OPENCL?

- Computational performance has shifted from clock speed to cores
- Multiple CPUs and programmable GPUs
- Need a programming interface that allows users to take advantage of all of the system resources
- Supports **general purpose** parallel computations

WHY OPENCL?

- OpenCL is device agnostic
- As an open standard, code should be portable across implementations
- No single company controls the specification - vendor neutral

OPENCL DEVICES

- Commonly CPUs and GPUs
- Could be any type of hardware that is compliant
 - Multimedia chip
 - FPGA
 - Embedded processors



GOALS OF OPENCL?

- Simple computing model - clean API
- ANSI-C99 language support
 - Additional qualifiers, data types and built-in functions
- Thread management framework
 - Application and thread-level synchronization

GOALS OF OPENCL?

- Easy to use and needs to be lightweight and efficient
- Need to be able to use all computational resources in computer
- IEEE-754 compliant in rounding behavior
- Minimum errors for math functions
- Provides guidelines for new hardware requirements

USES OF OPENCL

- Where OpenCL can be used
 - Image, video and audio processing
 - Simulations and scientific calculations
 - Medical imaging
 - Financial models
- Data-parallel algorithms that are computationally significant

DATA PARALLEL COMPUTING

- Many types of “parallel” computing
 - Granularity
- Grid Computing
- MPI/OpenMPI
- OpenMP/threads
- SIMD

Course

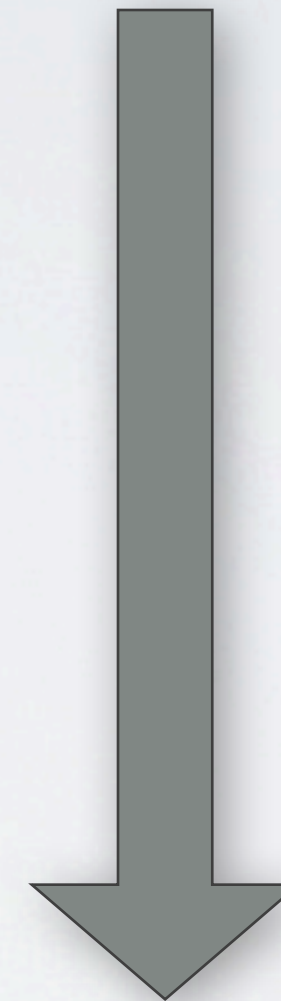
Grid Computing

MPI/OpenMPI

OpenMP/threads

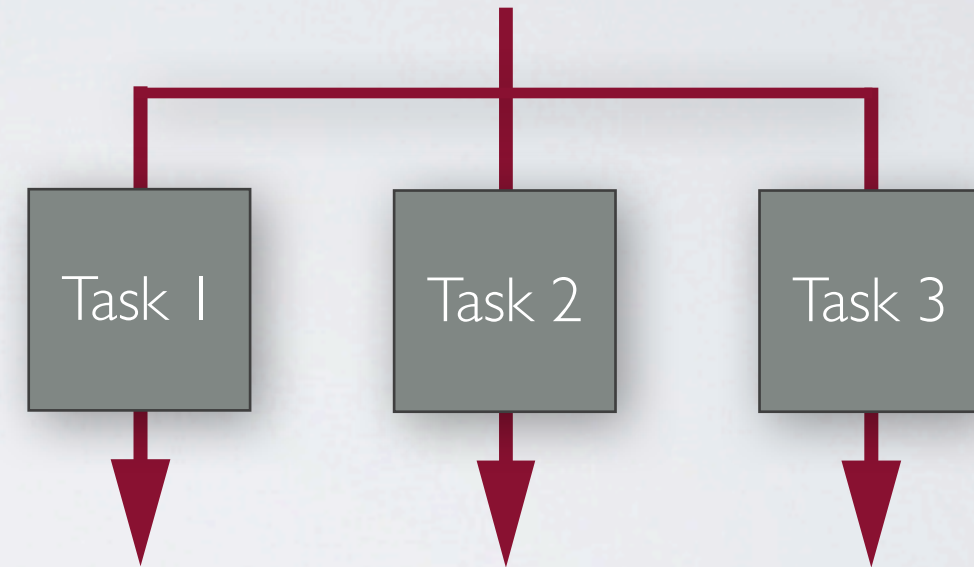
Fine

SIMD



DATA PARALLEL COMPUTING

- “Coarse” grain distribution in single system - Task Parallelism
- Same operation (or set of operations) is performed on multiple data elements independently - Data Parallelism



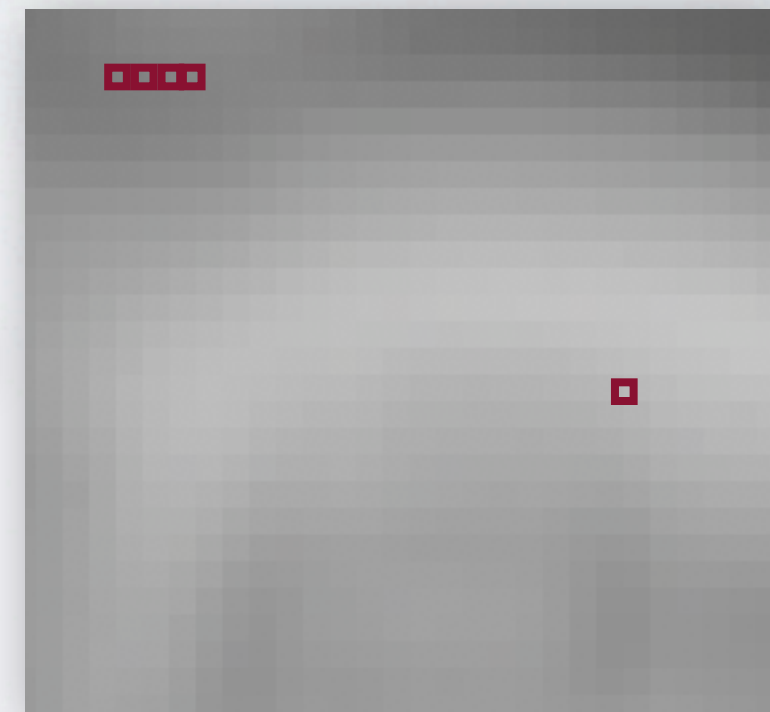
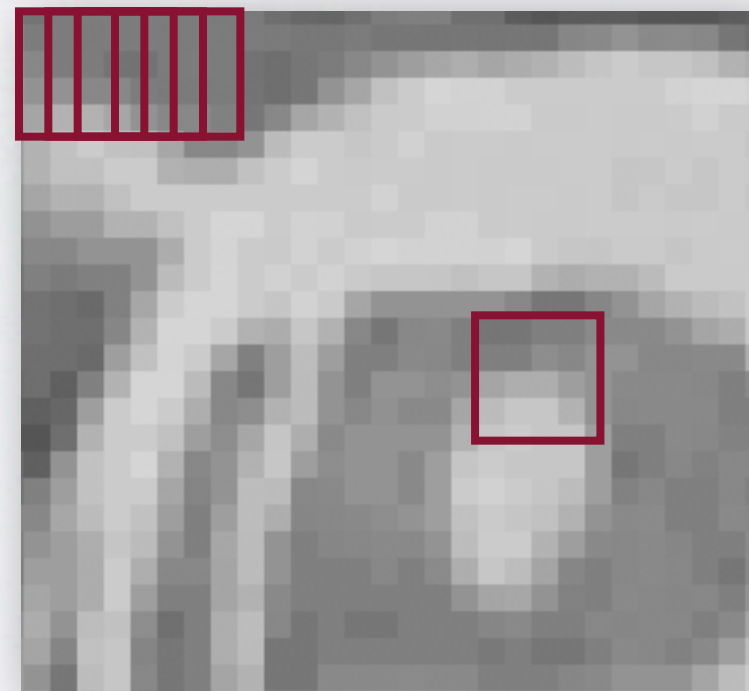
0	1	-5	-3	4	-8	2	7
---	---	----	----	---	----	---	---

↓ **abs()**

0	1	5	3	4	8	2	7
---	---	---	---	---	---	---	---

OPENCL DATA PARALLEL EXAMPLE

- Box Filter
 - Operation on each pixel can be performed independent of the others
 - Results can be stored to separate location
 - A good amount of work needs to be done



OPENCL AND OPENGL

- OpenCL plays well with OpenGL
- Data can be shared between computations (OpenCL) and display (OpenGL)
- OpenCL on GPU incurs little to no performance hit when integrated with OpenGL

WHAT ISN'T RIGHT FOR OPENCL

- Sequential problems
- Calculations that require a lot of pointer chasing or constant data permutation
- Calculations that require a lot of communication and result updates
- Device dependent limitations

Not everything will work well with OpenCL
(and that's OK)

WHAT ABOUT CUDA?

- CUDA is an extremely powerful, advanced GPGPU programming interface
- Integrates well with existing code (C/C++)
- However, CUDA is **not** device agnostic
 - Only works on NVIDIA hardware
 - Only (currently) targets the graphics card
 - Vendor controlled

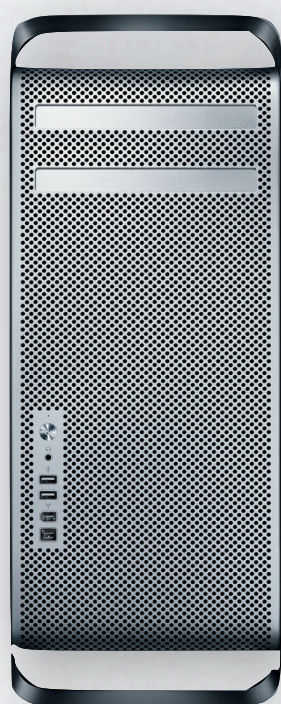
OPENCL IMPLEMENTATIONS

- Mac OS X Snow Leopard (10.6)
 - System Framework
- NVIDIA
- AMD

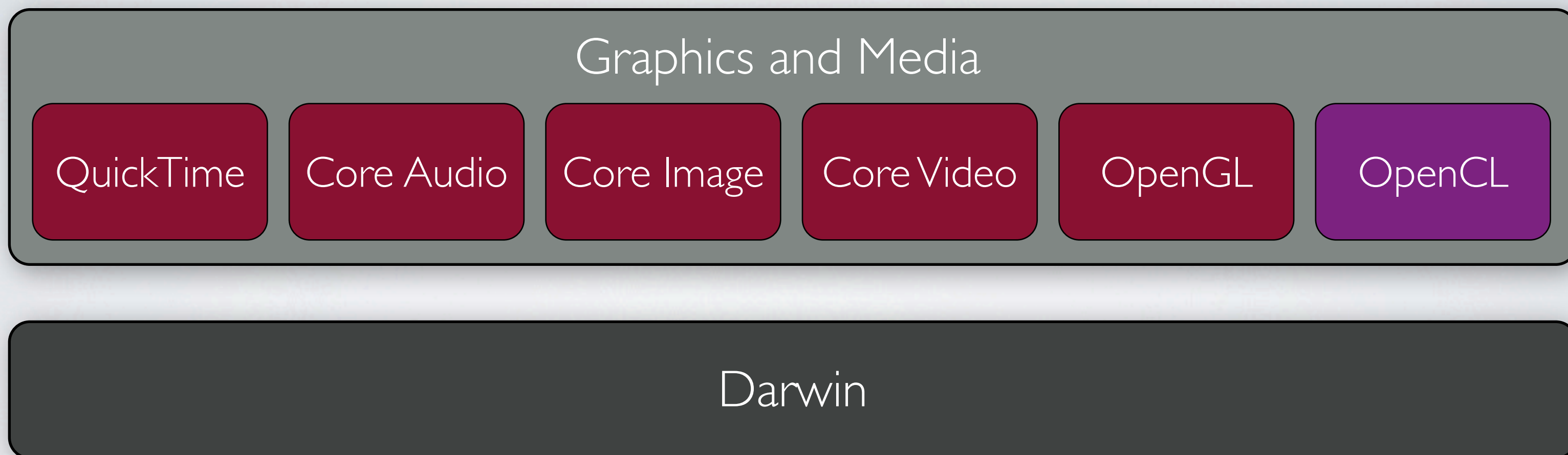


OPENCL-CAPABLE COMPUTERS

- Any computer with a CPU
- Macs - GPU
 - Mac Pro
 - 24" iMac
 - MacBook Pro
- NVIDIA - All currently shipping cards



OPENCL IN SNOW LEOPARD



WHY THE FUSS OVER THE GPU?

- GPUs are crazy fast floating-point number crunchers
- GPUs are designed for highly scalable parallelism
- GPU performance is increasing at a rate faster than CPU performance
- GPU bandwidth (in-card, memory to ALU) is much larger than the CPU

Core 2 Duo

- 150 W
- 4 cores
- 8.5 GB/s
- 45 GF/s

NVIDIA GTX 285

- 204 W (max)
- 240 cores
- 159 GB/s
- ~1 TFLOP/s

GPU LIMITATIONS

- System to GPU bandwidth is limited
- GPUs aren't very smart and don't handle errors well
- GPU programs can be difficult to debug
- GPUs like their data arranged in very specific ways for optimal efficiency

3-4 GB/s transfer = 1 byte/clock tick

During the time to copy 16 bytes:

- 64 adds
- 64 multiplies
- Loaded 256 bytes into register from L1
- Stored 256 bytes from register to L1
- Copied 16 vector registers to another register

OPENCL

- Open specification
- Device agnostic
- Treats all supported devices as peers
- Portable
- Low barrier to entry
- Easy access to GPGPU programming
- Allows more complete utilization of system resources
- Standard part of Mac OS X Snow Leopard (10.6)
- Available for other platforms and Operating Systems

OPENCL DEMO

MORE INFORMATION

- MacResearch.org
 - OpenCL - <http://www.macresearch.org/opencv>
 - Cocoa for Scientists - http://www.macresearch.org/cocoa_for_scientists
 - Xgrid Tutorials - http://www.macresearch.org/the_xgrid_tutorials
 - MacResearch Amazon Store - <http://astore.amazon.com/macreseorg-20>