



Teach GPU Accelerated Computing with NVIDIA Teaching Kit for Educators

Wen-Mei Hwu (University of Illinois) and Joe Bungo (NVIDIA)

AGENDA

Introduction to NVIDIA's GPU Educators Program and GPU Teaching Kits

GPU Teaching Kit for Accelerated Computing Syllabus Overview

Teaching Kit Contents

UIUC Activities

Further Q&A

GPU EDUCATORS PROGRAM

Advancing STEM Education with Accelerated Computing

"What an amazing resource for educators in GPU computing! The GPU Teaching Kit has a wealth of resources that allow both experienced and new teachers in parallel computing easily incorporate GPUs into their current course or design an entirely new course."

Prof. John Owens, UC-Davis

"The GPU teaching kit covers all aspects of GPU based programming.. the epitome for educators who want to float a course on heterogeneous computing using graphics processors as accelerators."

Dr. Tajendra Singh, UCLA

"Teaching resources such as these will be invaluable in helping the next generation of scientists and engineers know how to fully harness the capability of this exciting technology."

Dr. Alan Gray, University of Edinburgh

"The Teaching Kit covers all the needed content of a GPU/computing course.. The projects and quiz designs are handy, saving a lot of time and effort. Moreover, the whole structure is well organized to lead students step by step in CUDA programming. I highly recommend integrating it into a related syllabus."

Dr. Bin Zhou, University of Science and Technology of China

FLAGSHIP OFFERING: GPU TEACHING KITS

Breaking the Barriers to GPU Education in Academia

Co-develop with academic partners

Comprehensive teaching materials

Lecture slides and notes

Lecture videos

Hands-on labs/solutions

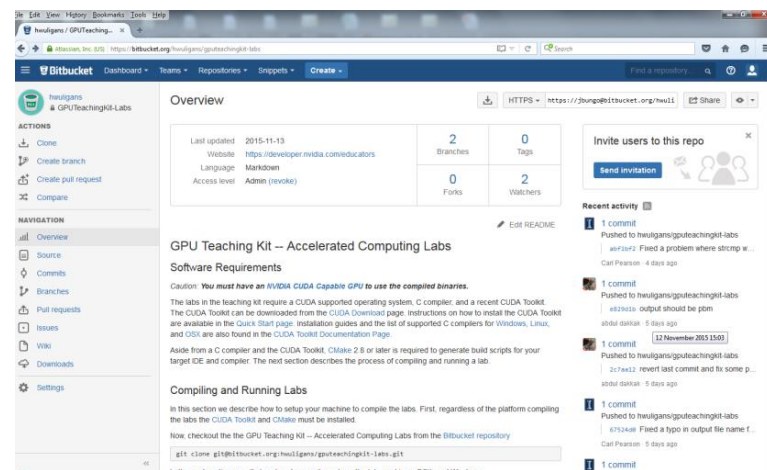
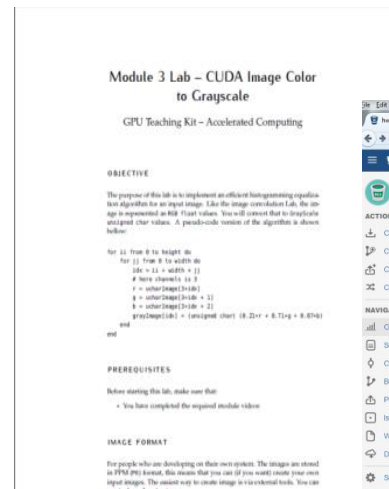
Larger coding projects/solutions

Quiz/exam questions/solution

GPU compute resource

Software tools

Textbooks and/or e-books



FLAGSHIP OFFERING: GPU TEACHING KITS

Breaking the Barriers to GPU Education in Academia

Different kits for different courses

Accelerated/parallel computing
Robotics

Machine/Deep learning

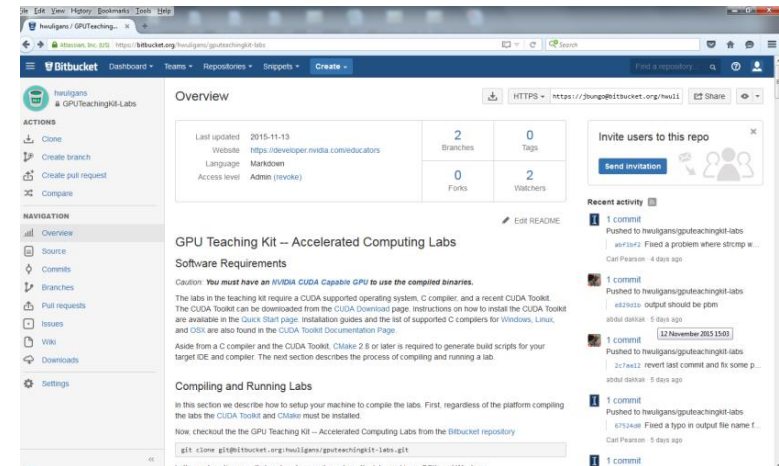
Computer vision

Computer architecture

Computational domain sciences

Etc.

Localizations/translations in progress



OTHER PROGRAM OFFERINGS

Collaborative Opportunities and Supporting Expertise

Instructor workshops, conferences, sponsorships and exhibits

Enablement web pages

Getting started guides/videos

Email updates

Feedback and enhancement requests



GPU Teaching Kit – Accelerated Computing

Available to Instructors Now!

developer.nvidia.com/educators

GPU TEACHING KIT - ACCELERATED COMPUTING

Course Goals

Learn to program heterogeneous parallel computing systems

High performance and energy-efficiency

Functionality and maintainability

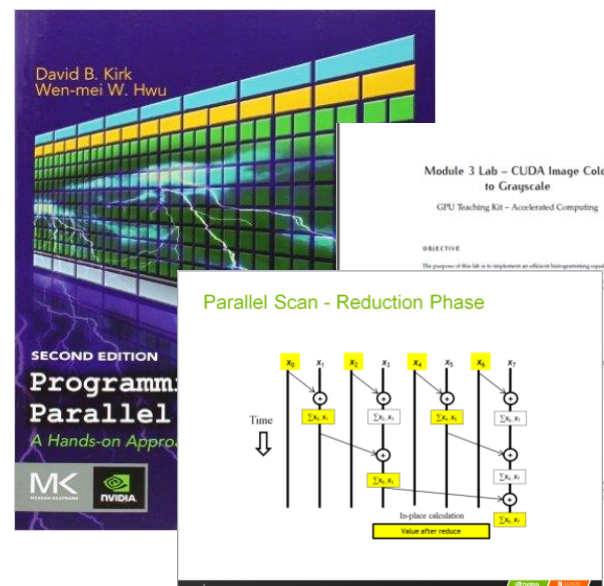
Scalability across future generations

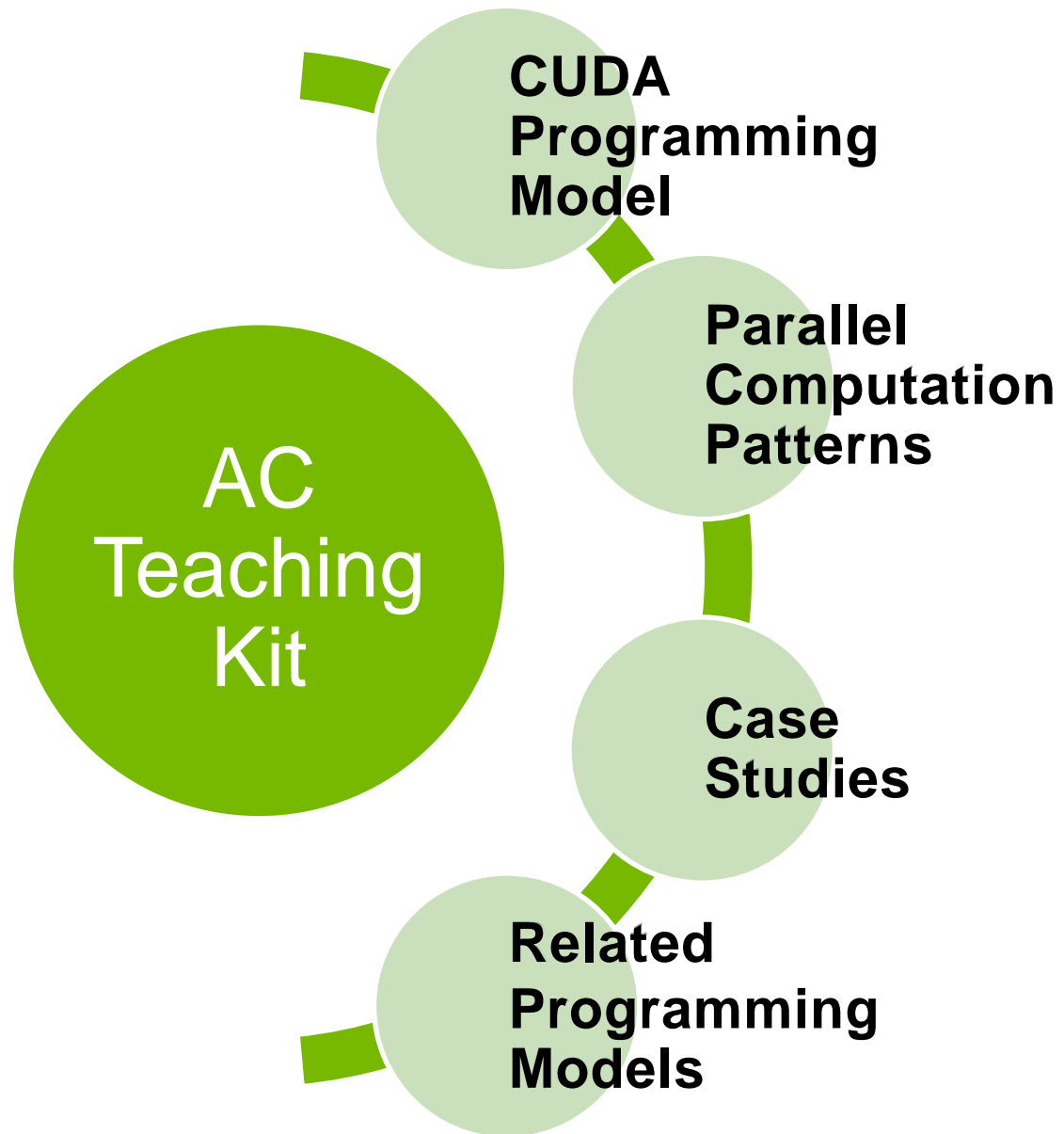
Technical subjects

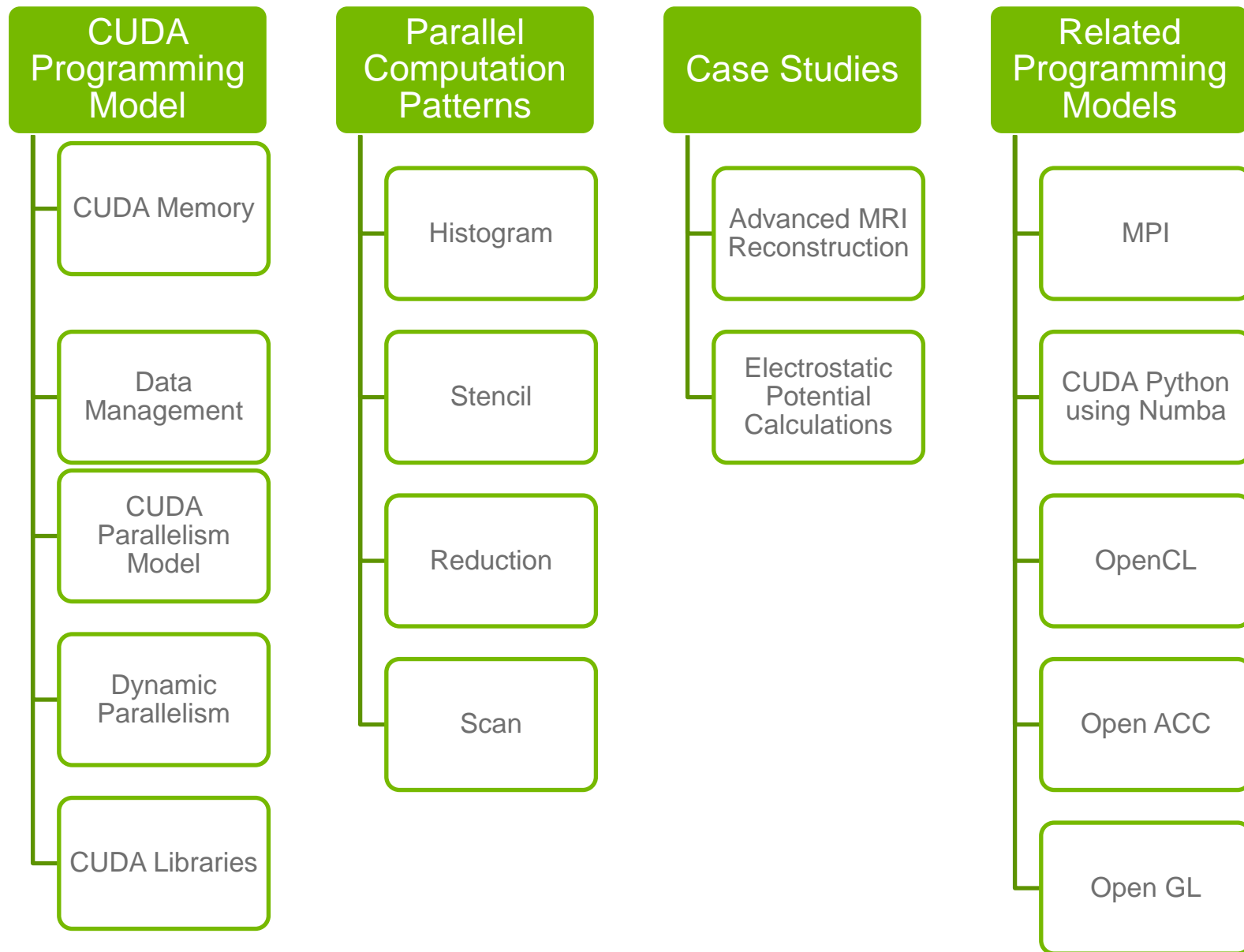
Parallel programming API, tools and techniques

Principles and patterns of parallel algorithms

Processor architecture features and constraints







AC TEACHING KIT CONTENTS

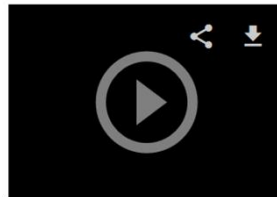
Syllabus

<http://syllabus.gputeachingkit.com/>

Module 1: Course Introduction

In this module we review course goals and syllabus and introduce the concepts of heterogeneous and parallel programming.

Lectures and Videos



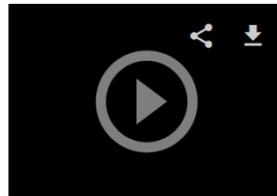
1.1 Course Introduction and Overview



PDF Slides
Lecture-1-1-overview.pdf



PowerPoint Slides
Lecture-1-1-overview.pptx



1.2 Introduction to Heterogeneous Parallel Computing



PDF Slides
Lecture-1-2-heterogeneous-computing.pdf



PowerPoint Slides
Lecture-1-2-heterogeneous-computing.pptx

AC TEACHING KIT CONTENTS

E-book Chapters

Chapters from “Programming Massively Parallel Processors: A Hands-on Approach”
2nd and 3rd edition, by David Kirk and Wen-Mei Hwu

.pdf format

Publisher 30% discount and free worldwide shipping on additional copies



AC TEACHING KIT CONTENTS

Lecture Slides

Supplement e-book chapters

Embedded audio narration

Current Release: 49 slide decks from 17 modules

.pptx and .pdf format

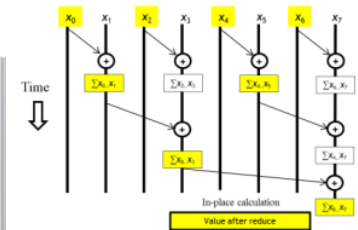


blockIdx and threadIdx

- Each thread uses indices to decide what data to work on
 - blockIdx: 1D, 2D, or 3D (CUDA 4.0)
 - threadIdx: 1D, 2D, or 3D

device

Parallel Scan - Reduction Phase



Kernel Regions

- Kernel constructs are descriptive of programmer intentions.
 - The compiler has a lot of flexibility in its use of the information
- This is in contrast with Parallel, which is prescriptive of the action for the compile follow

```
#pragma acc kernels
{
  #pragma acc loop gang(1024)
  for (int i=0; i<2548; i++) {
    A[i] = B[i];
  }
  #pragma acc loop gang(1024)
  for (int i=0; i<2548; i++) {
    A[i] = A[i]*2;
  }
  for (int k=0; k<2048; k++) {
    B[k] = A[k];
  }
}
```

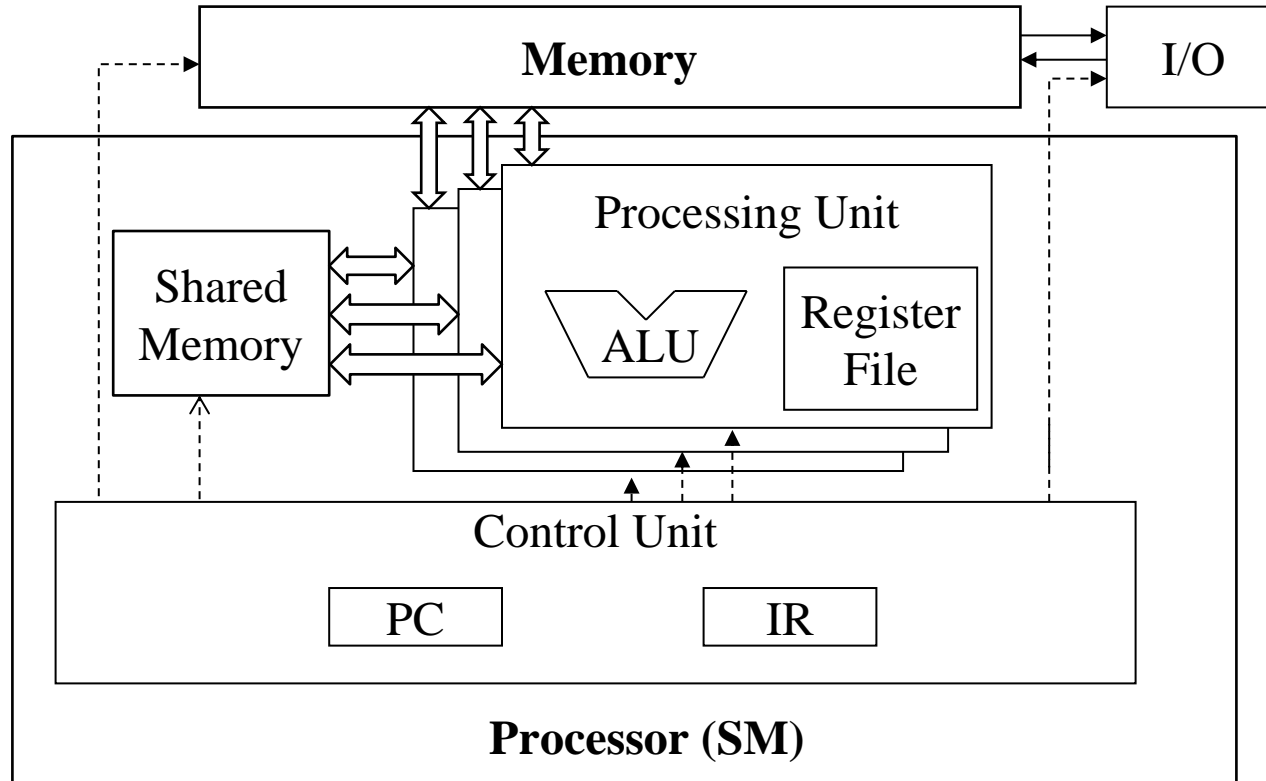


Module 5.1 – Thread Execution Efficiency

Warps and SIMD Hardware

SMs are SIMD Processors

- Control unit for instruction fetch, decode, and control is shared among multiple processing units
 - Control overhead is minimized (Module 1)



SIMD Execution Among Threads in a Warp

- All threads in a warp must execute the same instruction at any point in time
- This works efficiently if all threads follow the same control flow path
 - All if-then-else statements make the same decision
 - All loops iterate the same number of times

Control Divergence

- Control divergence occurs when threads in a warp take different control flow paths by making different control decisions
 - Some take the then-path and others take the else-path of an if-statement
 - Some threads take different number of loop iterations than others
- The execution of threads taking different paths are serialized in current GPUs
 - The control paths taken by the threads in a warp are traversed one at a time until there is no more.
 - During the execution of each path, all threads taking that path will be executed in parallel
 - The number of different paths can be large when considering nested control flow statements

AC TEACHING KIT CONTENTS

Lecture Videos

Supplement e-book chapters

Useful for “flipped” course and self-paced learning

Current Release: 49 slide decks from **17** modules

Stream individually or download as .mp4 from
<http://syllabus.gputeachingkit.com>

GPU Teaching Kit — Accelerated Computing

PDF Slides
Lecture 4-4 tiled-matrix-multiplication-kernel.pdf
PowerPoint Slides
Lecture 4-4 tiled-matrix-multiplication-kernel.pptx

Handling Arbitrary Matrix Sizes in Tiled Algorithms

PDF Slides
Lecture 6-5 tile-boundary-condition-2015.pdf
PowerPoint Slides
Lecture 6-5 tile-boundary-condition-2015.pptx

Labs

- Basic Matrix Multiplication
Module 4-6 BasicMatrixMultiplication.pdf
- CUDA Tiled Matrix Multiplication
Module 4-6 TiledMatrixMultiplication.pdf

Book Chapters

- Chapter 4 - Memory and Data Locality
3rd Edition-Chapter04-memory-model-2-2-2016.pdf

Module 5: Thread Execution Efficiency

In this module we explore how CUDA threads execute on SIMD Hardware and how to analyze the performance impact of control divergence.

Lectures and Videos

Warps and SIMD Hardware

PDF Slides
Lecture 5-1-warps-simd.pdf
PowerPoint Slides
Lecture 5-1-warps-simd.pptx

Performance Impact of Control Divergence

PDF Slides
Lecture 5-2-control-divergence.pdf
PowerPoint Slides
Lecture 5-2-control-divergence.pptx

Quiz

- Module 5 Quiz
Module 5 Quiz.pdf

Book Chapters

Some Observations (1)

```
1. void computeBlock(float *P, const float *M, const float *N, int Mx, int Nx, int Ny)
2. {
3.     // Arguments are parallel loop copies of [0, Mx*Nx], [0, Mx*Nx], [0, Mx*Nx]
4.     for (int i=0; i<Mx; i++)
5.     {
6.         // Arguments are temp
7.         float sum = 0;
8.         for (int k=0; k<Nx; k++)
9.         {
10.            float x = N[i*Nx+k];
11.            sum += x*x;
12.        }
13.        P[i*Nx] = sum;
14.    }
15. }
```

The code is almost identical to the sequential version, except for the two lines with `sum` at line 3 and line 5.

RGB Color Image Representation

- Each pixel in an image is an RGB value
- The format of an image's row is (r g b) (r g b) ... (r g b)
- RGB ranges are not distributed uniformly
- Many different color spaces, here we show the constants to convert to AdobeRGB color space

an

ing block

```
- Convert recurrences from sequential:
for (j=1; j<out[j]; j++)
out[j] = out[j-1] + f(j);

- into parallel:
forall(j) { temp[j] = f(j); }
scan(out, temp);

- Useful for many parallel algorithms:
* Reduce sort      * Polynomial evaluation
* Quickselect      * Schlegel's algorithm
* Binary comparison * Two-pointers
* Linear analysis   * Histograms, ...
* Stream comparison
```

AC TEACHING KIT CONTENTS

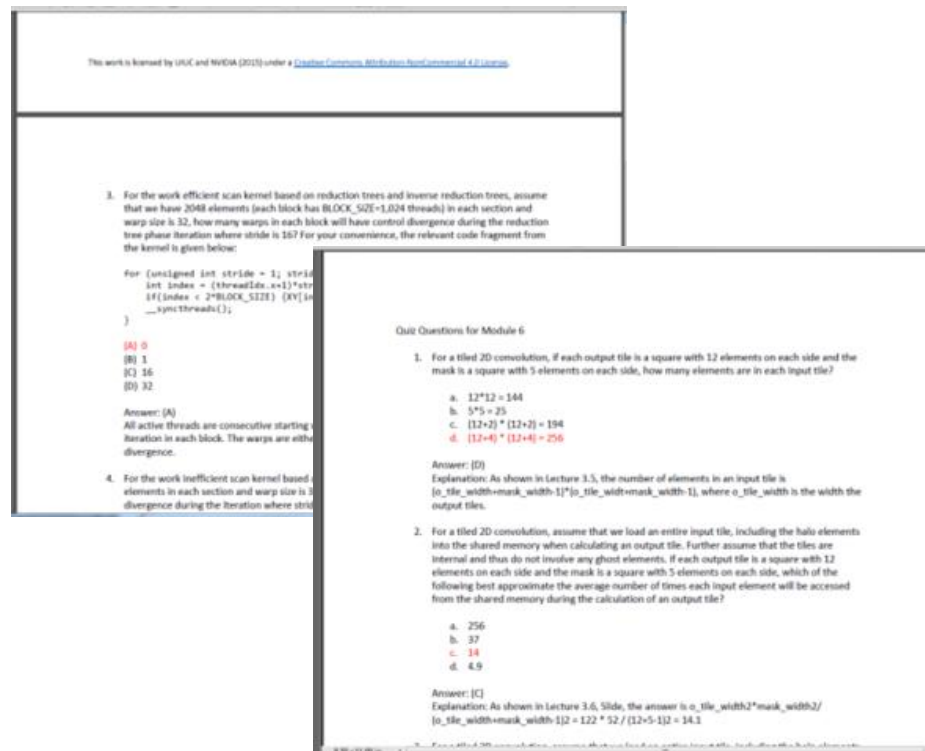
Quiz questions/answers

Multiple choice, including rationale for answers

Students should be able to answer from lecture content

Current Release: 16 quizzes from **17** modules

.docx and .pdf formats



AC TEACHING KIT CONTENTS

Hands-on Labs/solutions

1-2 week assignments

Includes description, objectives, prerequisites and open-ended questions, and solution code templates

Latest source code and instructions always on a private Git Repository [BitBucket]

Current Release: 17 Labs/solutions from 17 modules

.docx and .pdf formats



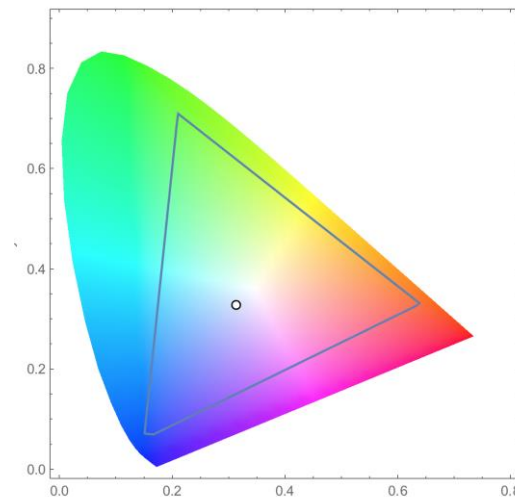
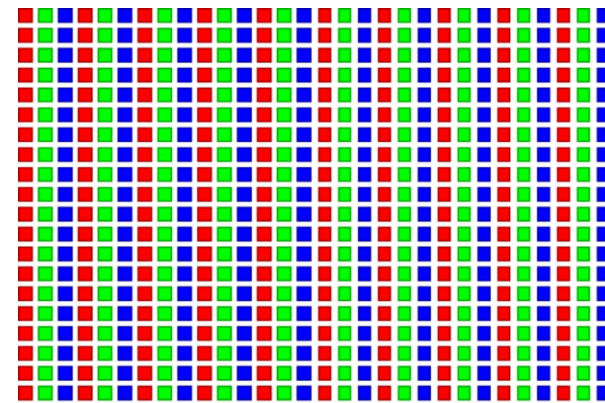
Image Color to Grayscale Conversion



A grayscale digital image is an image in which the value of each pixel carries only intensity information.

Image Color to Grayscale Conversion

- Each pixel in an image is an RGB tuple
- Since colors are not distributed uniformly, there are different color spaces, here we show the constants to convert to AdobeRGB color space
- The vertical axis (y value) and horizontal axis (x value) show the fraction of the pixel intensity that should be allocated to G and R. The remaining fraction ($1-y-x$) of the pixel intensity is B
- The triangle contains colors representable in this color space

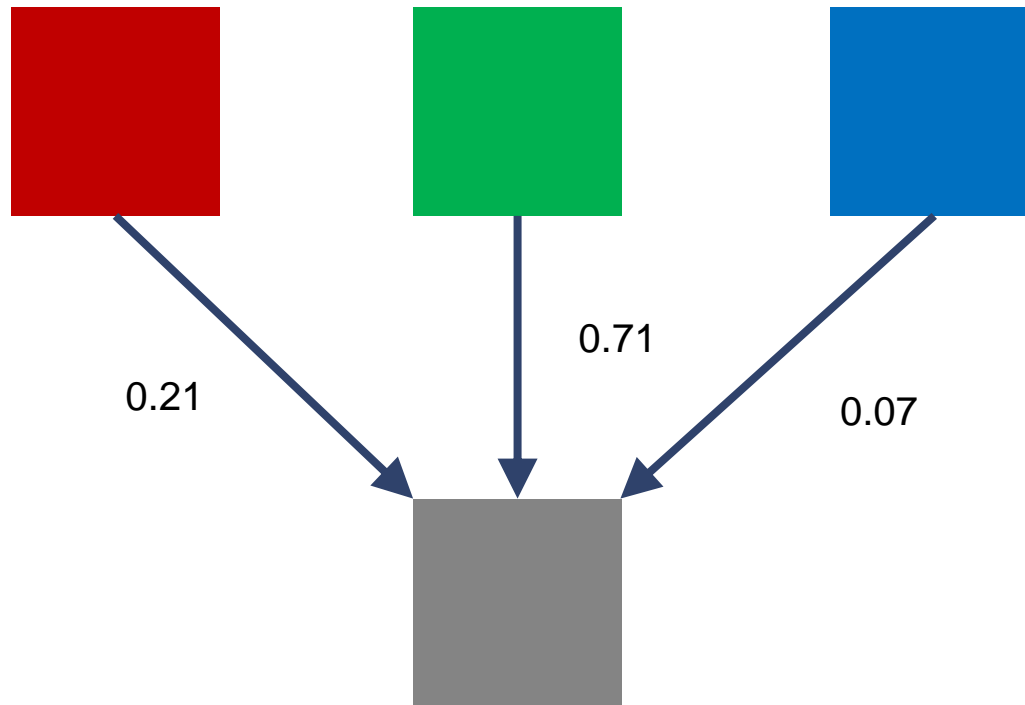


Transformation Formula

For each pixel (r g b) at (I, J) do:

$$\text{grayPixel}[I,J] = 0.21*r + 0.71*g + 0.07*b$$

This is a dot product $\langle [r,g,b], [0.21,0.71,0.07] \rangle$ with the constants specific to the RGB space



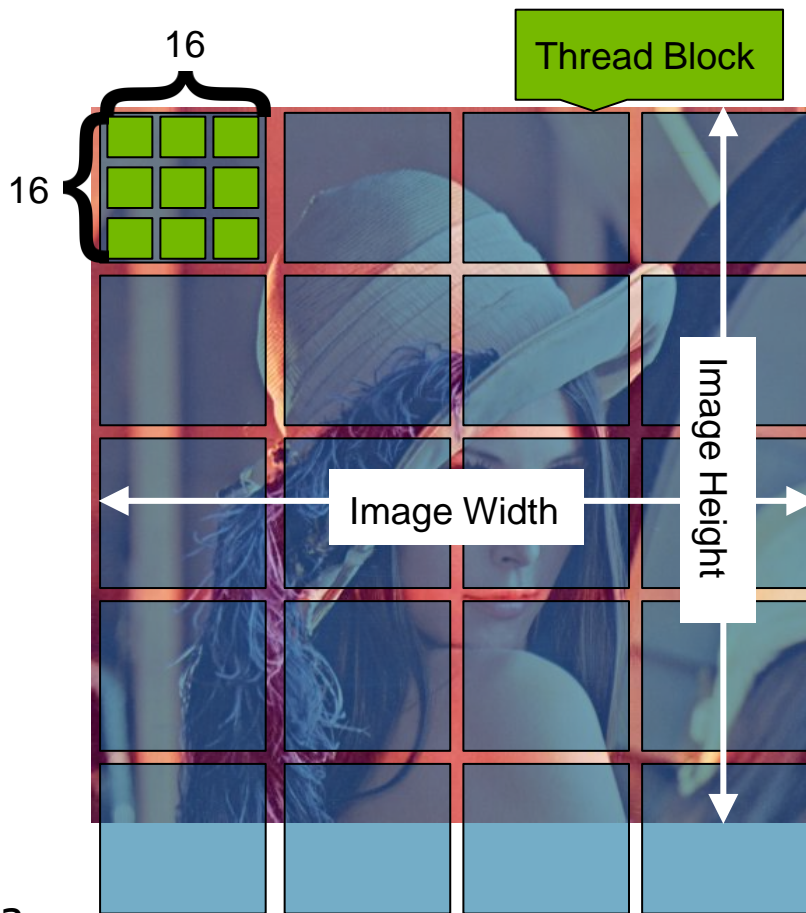
RGB to Grayscale Host Code

```
#define TILE_WIDTH 16
```

```
dim3 dimBlock(TILE_WIDTH, TILE_WIDTH, 1);
```

```
dim3 dimGrid(ceil((float)imageWidth / TILE_WIDTH),  
             ceil((float)imageHeight / TILE_WIDTH));
```

```
rgb2gray<<<dimGrid, dimBlock>>>(deviceOutputImageData,  
                                 deviceInputImageData, imageChannels,  
                                 imageWidth, imageHeight);
```



RGB to Grayscale Conversion Code

```
#define TILE_WIDTH 16
__global__ void rgb2gray(float *grayImage, float *rgbImage, int channels,
                        int width, int height) {
    int x = threadIdx.x + blockIdx.x * blockDim.x;
    int y = threadIdx.y + blockIdx.y * blockDim.y;

    }
```

RGB to Grayscale Conversion Code

```
#define TILE_WIDTH 16
__global__ void rgb2gray(float *grayImage, float *rgbImage, int channels,
                        int width, int height) {
    int x = threadIdx.x + blockIdx.x * blockDim.x;
    int y = threadIdx.y + blockIdx.y * blockDim.y;

    if (x < width && y < height) {
        // get 1D coordinate for the grayscale image
        int grayOffset = y * width + x;
        // one can think of the RGB image having
        // CHANNEL times columns than the gray scale image
        int rgbOffset = grayOffset * channels;
        float r = rgbImage[rgbOffset]; // red value for pixel
        float g = rgbImage[rgbOffset + 1]; // green value for pixel
        float b = rgbImage[rgbOffset + 2]; // blue value for pixel
        // perform the rescaling and store it
        // We multiply by floating point constants
        grayImage[grayOffset] = 0.21f * r + 0.71f * g + 0.07f * b;
    }
}
```


RGB to Grayscale Conversion Code

```
#define TILE_WIDTH 16
__global__ void rgb2gray(float *grayImage, float *rgbImage, int channels,
                        int width, int height) {
    int x = threadIdx.x + blockIdx.x * blockDim.x;
    int y = threadIdx.y + blockIdx.y * blockDim.y;

    if (x < width && y < height) {
        // get 1D coordinate for the grayscale image
        int grayOffset = y * width + x;
        // one can think of the RGB image having
        // CHANNEL times columns than the gray scale image
        int rgbOffset = grayOffset * channels;
        float r = rgbImage[rgbOffset]; // red value for pixel
        float g = rgbImage[rgbOffset + 1]; // green value for pixel
        float b = rgbImage[rgbOffset + 2]; // blue value for pixel
        // perform the rescaling and store it
        // We multiply by floating point constants
        grayImage[grayOffset] = 0.21f * r + 0.71f * g + 0.07f * b;
    }
}
```

AC TEACHING KIT CONTENTS

Larger coding projects/solutions

3-4 week, open-ended, multidisciplinary,
final semester projects

Real projects from real UIUC students

Not tied to specific modules

Current Release: 10 projects/solutions

.docx and .pdf formats

Solutions in source code

The collage displays three key components of the AC Teaching Kit:

- CUDA Code Snippet:** A snippet of C++ code using CUDA, showing memory allocation and kernel launches.
- GPU Teaching Kit: Project Guidelines:** A document outlining the project's purpose, goals, and a project outline. It includes a table comparing CPU and GPU recovery times.
- Results:** A document showing performance results, including a table of execution times for different data sizes.

Recovery	CPU Recovery Time (s)	GPU Recovery Time (s)
johnson	17.058	1.081

Series	0.018	0.043
johnson	17.058	1.081

OTHER RESOURCES

qwikLABS

Live, hands-on, self-paced learning environment to reinforce the concepts contained in the Teaching Kit

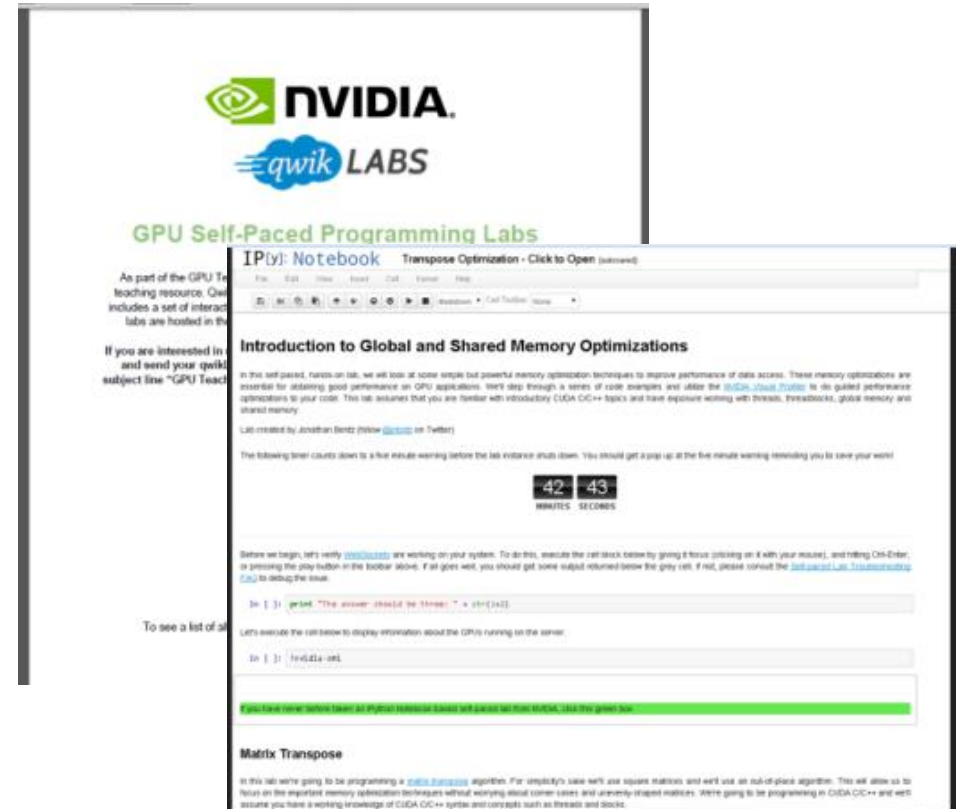
Labs include interactive instructions, coding and Q/A

Hosted in the cloud

Students only need a web-browser and internet access

Labs are timed

Free tokens with Teaching Kit



UIUC Activities

GPU Computing

UIUC ECE408/CS483

Semester calendar, 15 weeks

Uses 18 modules

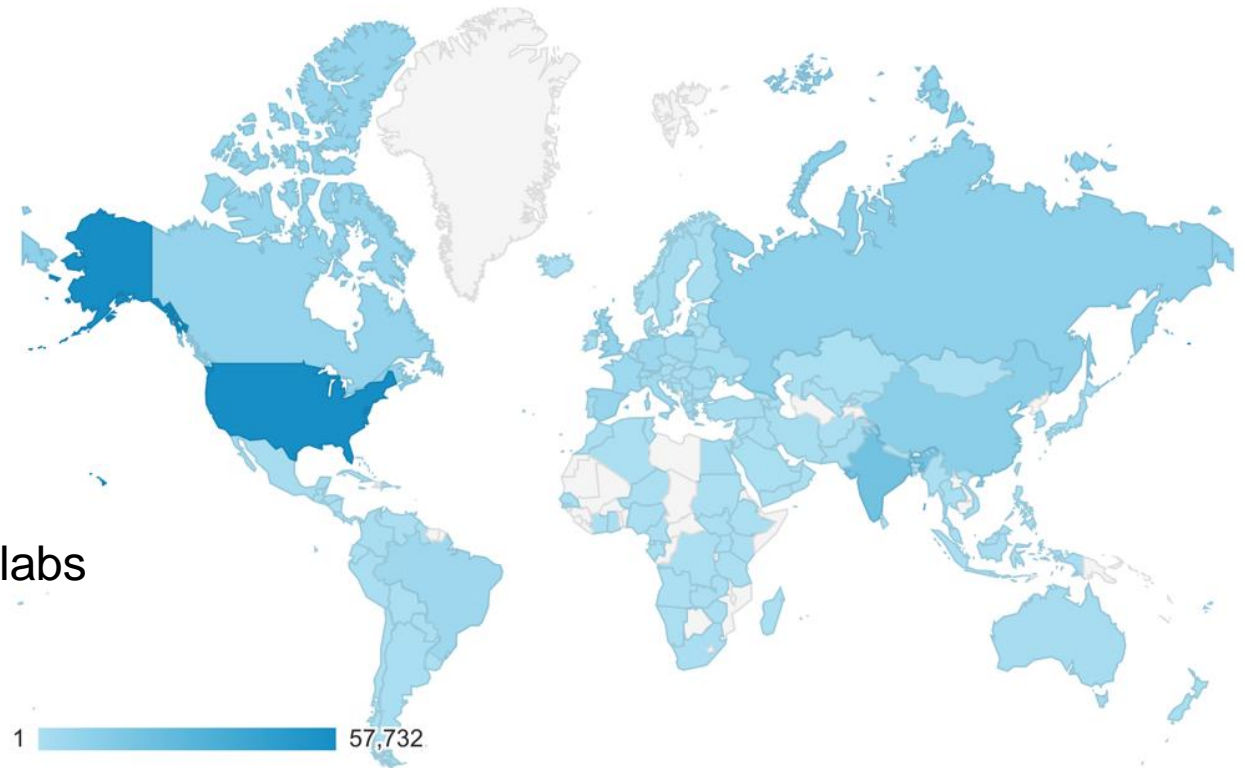
Lecture slides, quizzes, labs

Coursera HPP

7 weeks

Uses 10 modules

Lecture videos, lecture slides, quizzes, labs



➔ ECE408/CS483

Around 100 students from UIUC

➔ CS 598 HK

Around 80 students for UIUC and collaborating institutions

➔ Summer School

Around 100 students from all over the world

➔ Coursera HPP

Around 20,000 students worldwide

More Advanced Courses

- CS 598 HK (Parallel Algorithmic Techniques)**

- Taught this fall in conjunction with 8 US institutions. If interested in next year's class, contact aschuh@illinois.edu.

- Summer School**

- PUMPS concluded last week. If interested in next year's week long summer course held in Barcelona, contact aschuh@illinois.edu.

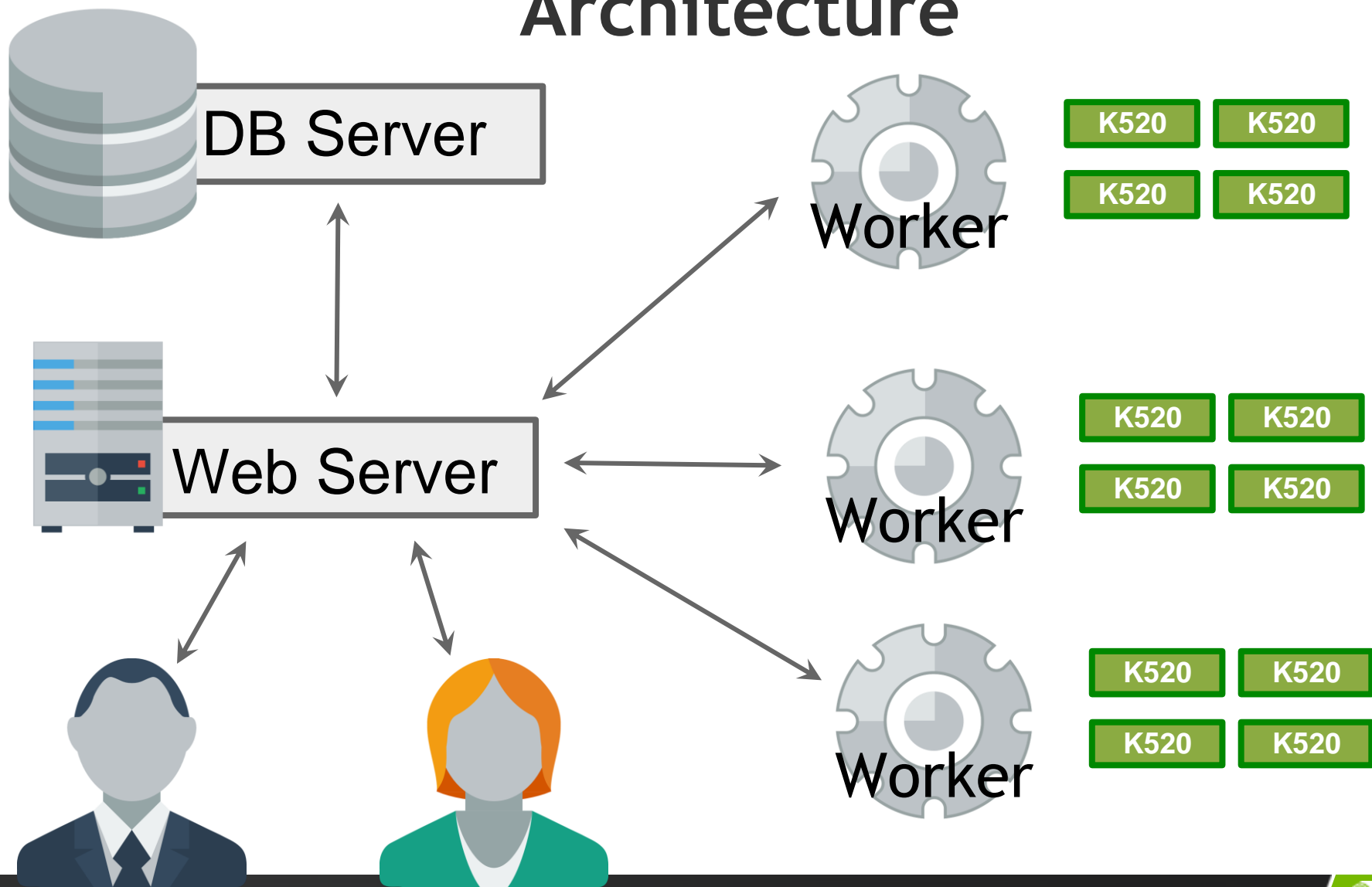
WebGPU.com

A System for Online GPU Development

- An online IDE for GPU development
- Used intensively at UIUC for the past 4 years
- Essential tool for the Coursera courses offered as well as the introductory and advanced teaching courses at UIUC
- Over 15,000 registered users
- Used as a vehicle for program analysis research

Published at EduPar 2016 "*WebGPU: A Scalable Online Development Platform for GPU Programming Courses*" - A. Dakkak, C. Pearson and W. Hwu

Architecture





Questions?

GPU Teaching Kit – Accelerated Computing

Available to Instructors Now!

developer.nvidia.com/educators