



ECE 508

Manycore Parallel Algorithms

Lecture 1: Introduction,
Logistics and Topics

Course Goals

- master common algorithmic techniques and computational thinking skills for high-performance, scalable manycore programming
- specifically, to understand
 - GPU computing hardware limitations and constraints,
 - desirable and undesirable computation patterns, and
 - practical techniques to convert undesirable patterns into desirable ones

Started as a Multi-University Graduate Course

- U. of Illinois/NCSA - Lead
 - Prof. Wen-mei Hwu
- U. of Minnesota
 - Prof. John Sartori
- U. of Tennessee, Knoxville
 - Prof. Greg. Peterson
- U. of Oklahoma
 - Prof. Ron Barnes
- NC State U.
 - Prof. Huiyang Zhou
- Michigan State U.
 - Prof. Dirk Colbry
- New Mexico State U.
 - Prof. Alla Kammerdiner
- Wake Forest State U.
 - Prof. Damian Valles
- U. of Houston, Clear Lake
 - Prof. Liwen Shih
- Clarkson U.
 - Prof. Brian Helenbrook



Professor

- Instructor: Prof. Steven S. Lumetta
 - 209 CSL, lumetta@illinois.edu
 - use ECE508 to start your e-mail subject line
 - (use Piazza discussions for anything not personal!)
 - Office hours: 1:30 – 3:30 pm Tuesdays
- Expert contributors: Carl Pearson, Simon Garcia, Abdul Dakkak, Cheng Li, Mert Hidayetoglu, Andy Schuh



**WARNING:
maintain 2m
separation!**

Brief Bio of Steve Lumetta

- Ph.D. in CS from University of California, 1998
(clusters of symmetric multiprocessors)
- research
 - optical network architecture and reliability
 - processor architecture
 - digital system testing / test compression
 - accelerator architectures (GPU, FPGA)
 - computational genomics
 - programming education
- teaching
 - major contributor to CompE undergrad core (120, 220, 391)
 - Director of CompE Programs for ZJUI campus
- other stuff
 - co-founded company based on packetized optics in datacenters
 - contributed to other major efforts (ex: Blue Waters)

TA

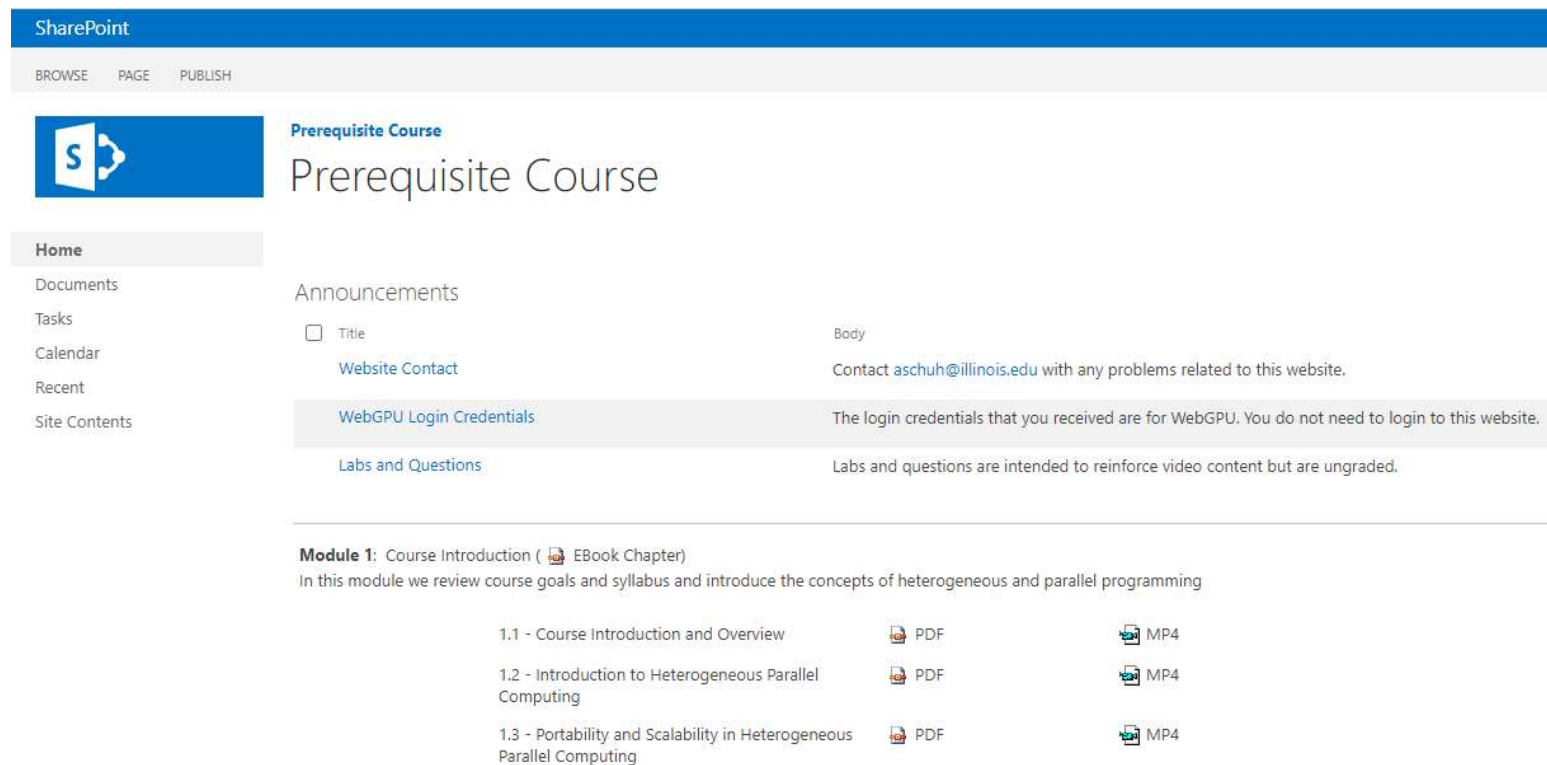
- Instructor: Kun Wu
 - kunwu2@illinois.edu
 - use ECE508 to start your e-mail subject line
 - Office hours: TBD
- Kun
 - advised by Wen-mei Hwu
 - took this class from me in F21



WARNING:
do not infect
your only TA!

508 Should Not be Your First GPU Class

(Take ECE408 or equivalent first!)



The screenshot shows a SharePoint page with a blue header bar containing the text 'SharePoint'. Below the header is a navigation bar with 'BROWSE', 'PAGE', and 'PUBLISH' links. The main content area has a blue sidebar on the left with a 'Home' button and links to 'Documents', 'Tasks', 'Calendar', 'Recent', and 'Site Contents'. The main content area is titled 'Prerequisite Course' and features a 'Prerequisite Course' icon. Below the title is an 'Announcements' section with a table of links and descriptions. The table has two columns: 'Title' and 'Body'. The first row contains a link to 'Website Contact' and the text 'Contact aschuh@illinois.edu with any problems related to this website.' The second row contains a link to 'WebGPU Login Credentials' and the text 'The login credentials that you received are for WebGPU. You do not need to login to this website.' The third row contains a link to 'Labs and Questions' and the text 'Labs and questions are intended to reinforce video content but are ungraded.' Below the table is a section titled 'Module 1: Course Introduction (EBook Chapter)' with a description: 'In this module we review course goals and syllabus and introduce the concepts of heterogeneous and parallel programming'. Below the description is a table of links and descriptions. The table has two columns: 'Title' and 'Body'. The first row contains a link to '1.1 - Course Introduction and Overview' and the text 'PDF'. The second row contains a link to '1.2 - Introduction to Heterogeneous Parallel Computing' and the text 'PDF'. The third row contains a link to '1.3 - Portability and Scalability in Heterogeneous Parallel Computing' and the text 'PDF'.

SharePoint

BROWSE PAGE PUBLISH

Prerequisite Course

Prerequisite Course

Home

Documents

Tasks

Calendar

Recent

Site Contents

Announcements

Title	Body
Website Contact	Contact aschuh@illinois.edu with any problems related to this website.
WebGPU Login Credentials	The login credentials that you received are for WebGPU. You do not need to login to this website.
Labs and Questions	Labs and questions are intended to reinforce video content but are ungraded.

Module 1: Course Introduction (EBook Chapter)

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

1.1 - Course Introduction and Overview	PDF	MP4
1.2 - Introduction to Heterogeneous Parallel Computing	PDF	MP4
1.3 - Portability and Scalability in Heterogeneous Parallel Computing	PDF	MP4

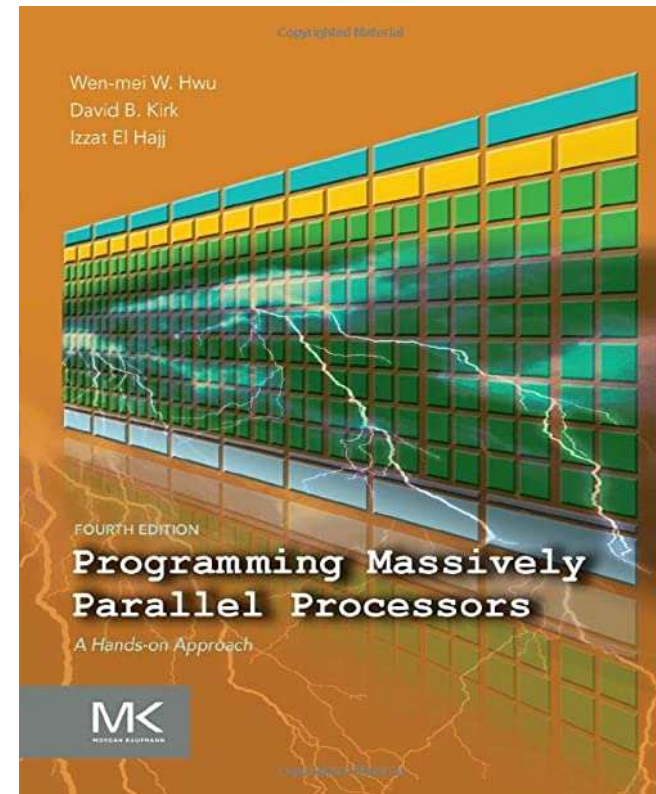
Course Web Page

<http://lumetta.web.engr.illinois.edu/508/>

- announcements
- handouts and lecture slides/recordings
- readings, documentation, software
(also see Github README and slides)
- discussion forum
 - try Piazza first
 - post ALL NON-PERSONAL questions there
 - do not post solutions, fragments of solutions, etc.

Check Web Page for Materials

- On the class web page:
 - lecture notes and recordings
 - additional readings
- Recommended background:
Hwu, Kirk and El Hajj,
*Programming Massively
Parallel Processors*, 4th ed.
- API Guides: NVIDIA, NVidia
CUDA 12.0 Programming
Guide (reference)



Use RAI for All Assignments

- submission system for project development
 - use both for MPs and final project
 - (there is no WebGPU!)
- You should have already received email with instructions for downloading RAI client and usage (be sure to get the right version!).
- MP0 is device query using RAI.

2/3 of Your Grade is Assignments

This is a lab-oriented course!

- Quizzes: 5%
- Exam: 30%
- Labs: 35%
- Project: 30%
 - Design Document: 25%
 - Project Presentation: 25%
 - Demo/Performance/Report: 50%

Let Us Know About Conflicts Early

- University has clear rules for conflicts (online)
 - Finals: **Section 3-201 of Student Code**
 - N.B. Not same as midterms; we have none.
- Finals rules
 - depend on class sizes;
 - if you can't tell, ask Jamie Smith in advising.
- If you have a conflict, **let me know early!**
(at least one week before the exam)

No Late Lab Submissions

- However, they will be posted early so you can plan your interviews, etc.
- I plan to modify labs.
 - Not less than two weeks before they are due.
 - Be sure that you know how to update from the Git repo! (and that you do so before starting)
 - Watch web page / repo for notice of changes.

Academic Integrity Policy Based on Student Code

- See **Section 1-402 of the UIUC Academic code.**
- You are allowed and encouraged to discuss assignments/MPs with other students in the class.
 - Getting verbal advice/help from people outside the class is also fine.
 - Any copying or exchange of code is unacceptable, including from public sources (other than the assignments themselves).

More on Academic Integrity

- Giving/receiving help on an exam from anyone other than the course staff (and team members in the final project ONLY) is unacceptable
- Minimum penalties for academic dishonesty:
 - 0 on the assignment / exam
 - reduction of final letter grade by one full letter
 - final grade reduced to F for second infractions

Final Project

- Most important product of the semester
- Apply some of the algorithm techniques to a challenging problem
- Sample topics provided in a few weeks
- Feel free to make suggestions and/or draw on your research (keep in mind that you will have partners and cannot claim all work as your own, if that matters to your advisor).

Projects in Teams of Three or Four

- Work can be divided up between team members in any way that works for you
- However, each team member will need to answer questions on the entire design
 - Rationale: if you don't know enough about the whole design to answer questions on it, you aren't involved enough

Massive Data Parallelism Key to GPU Performance

How can we leverage throughput-oriented architectures (GPUs) for performance?

- multiple Teraflops of execution throughput
- requires massive data parallelism
 - parallel threads doing similar work
 - regular computation and data accesses
- must avoid resource conflicts
 - example: off-chip DRAM bandwidth
 - conflicting parallel updates to memory

Architecture-Friendly Algorithms are Not New

Parallel computing has a long history

- starting in the late 1950s
- (or 1842, if one prefers).

During that history,

- **many** parallel **architectures performed well on** only a **few applications**,
- but **new algorithms were developed** to adapt applications to the architectures.
- The **Connection Machine (CM and CM-2)** is one interesting example from the 1980s.

Early GPUs were also Limited

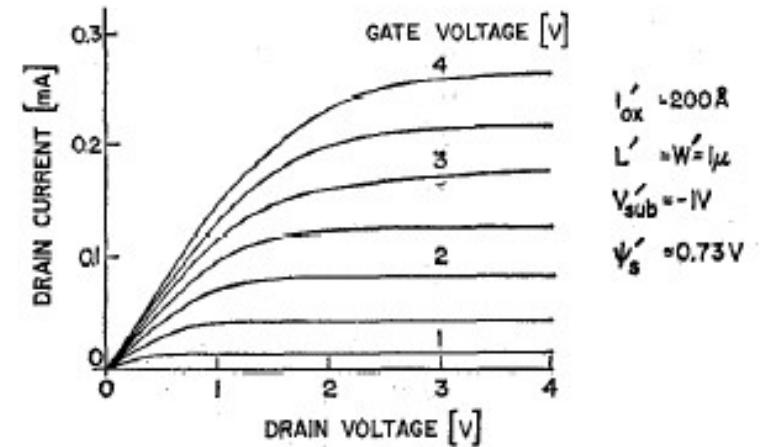
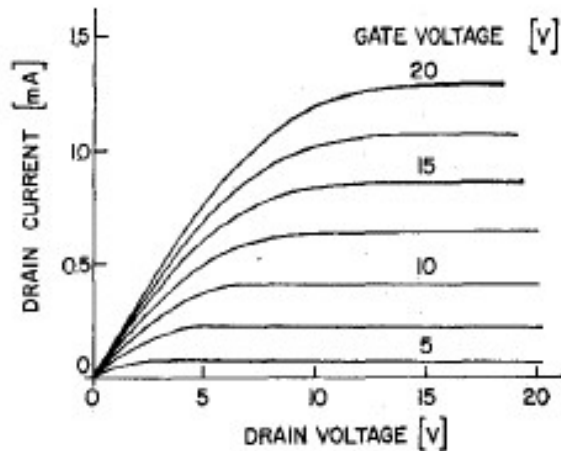
When **“general-purpose” GPUs appeared in 2007**,

- (and were first programmed by UIUC students taught by Hwu and Kirk!),
- the **architecture and CUDA** programming model **did not fit all applications**.

But **GPUs and CPUs both have important markets**, and

- with the decrease in single-core performance growth,
- parallelism became a necessity (see following slides).

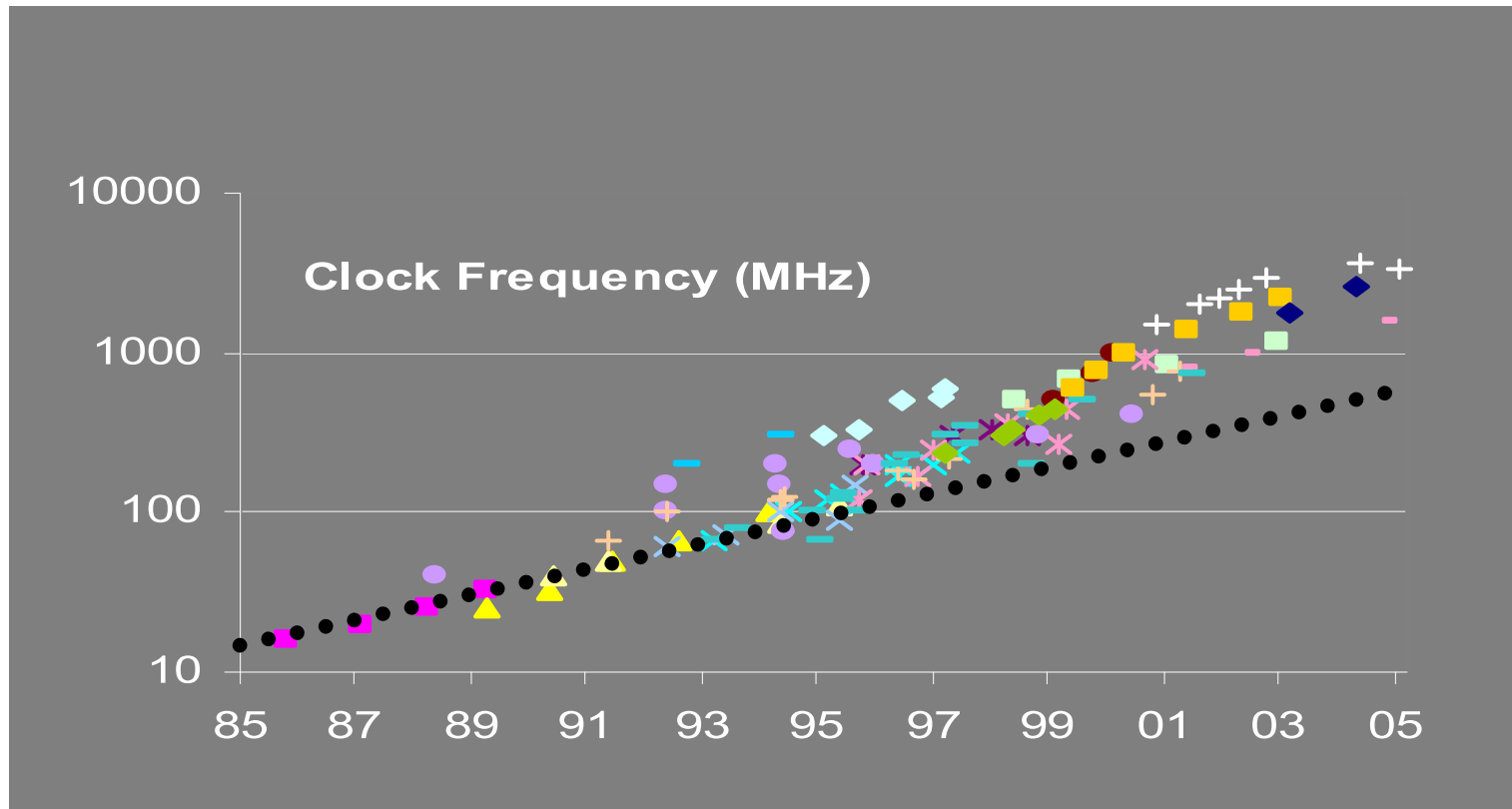
Dennard Scaling of MOS Devices



JSSC Oct 1974, page 256

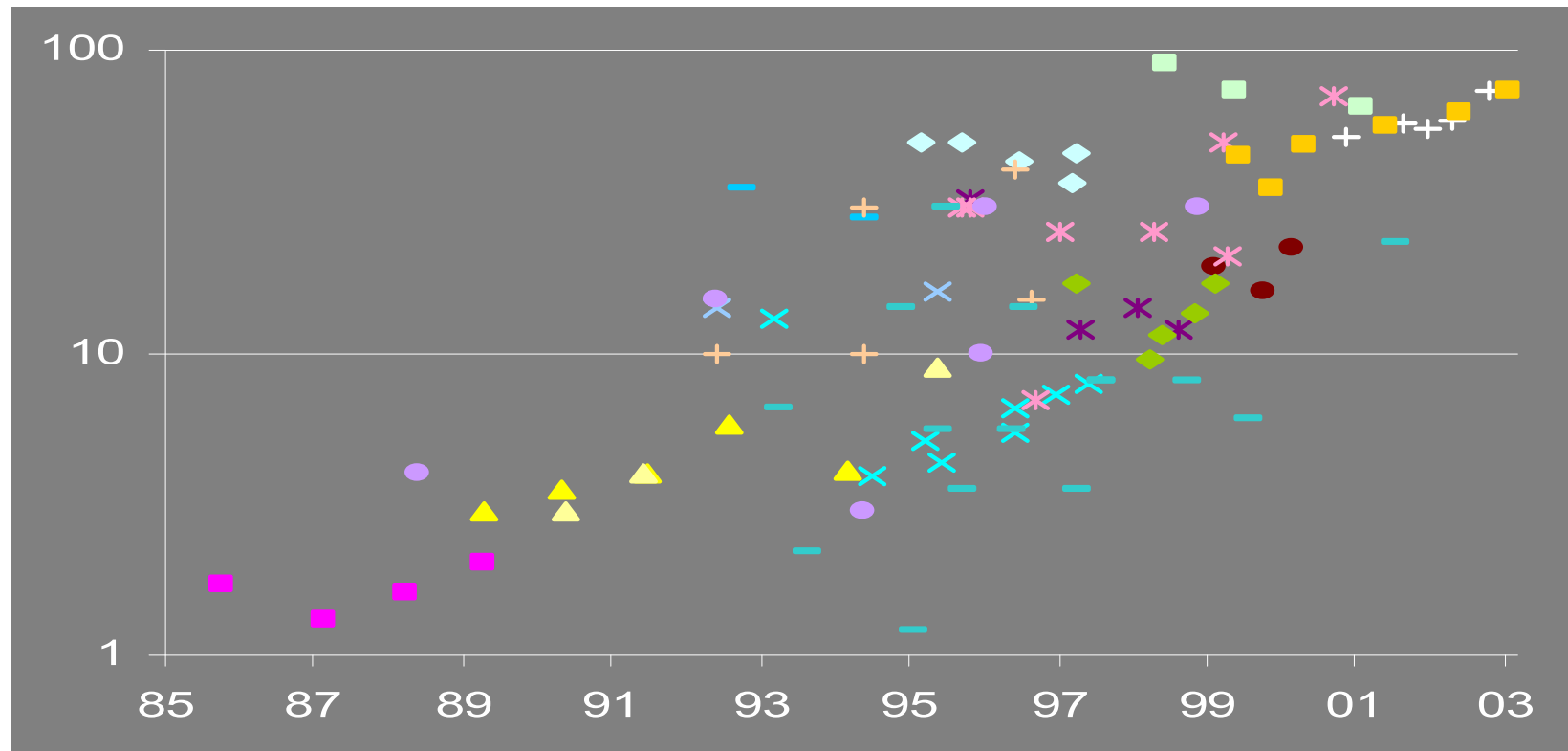
- In this ideal scaling, as $L \rightarrow \alpha * L$
 - $V_{DD} \rightarrow \alpha * V_{DD}$, $C \rightarrow \alpha * C$, $I \rightarrow \alpha * I$
 - Delay = CV_{DD}/I scales by α , so $f \rightarrow 1/\alpha$
 - Power for each transistor is $CV^2 * f$ and scales by α^2 , keeping total power constant for same chip area

Frequency Scaled Too Fast 1993-2003



Total Processor Power Increased

(super-scaling of frequency and chip size)




CPU and GPU are Ubiquitous

In less than a decade, **every system offered both CPU and GPU**

- from **supercomputers like Blue Waters**, which coupled high-end AMD processors with NVIDIA GPUs
- to **\$100 cell phones**, which combine low-power CPUs and GPUs in multi-chip modules.

Since 2007,

- **GPU architecture** has **evolved** to be easier to use (we'll see the impact on performance across generations), and
- **algorithmic techniques** have been **developed** (the **main focus of this course!**).



“CUDA is an elegant solution to the problem of representing parallelism in algorithms, not all algorithms, but enough to matter.”

V. Natoli, Kudos for CUDA,
HPCWire, 2010

Another Goal: Computational Thinking Skills

- The ability to translate/
formulate domain problems into
computation methods that can
execute efficiently with
available computing resources
 - Understanding the
relationship between the
domain problem and the
computational methods
 - Understanding the
strength and limitations
of the computing
devices
 - Designing the method
implementations to
steer away from the
limitations

Massive Parallelism - Regularity



Let's Look at the Scalability Challenges

What are some of the main challenges in developing scalable algorithms?

- conflicting accesses
- global memory bandwidth
- computational efficiency
- load balance

Conflicting Accesses Cause Delay

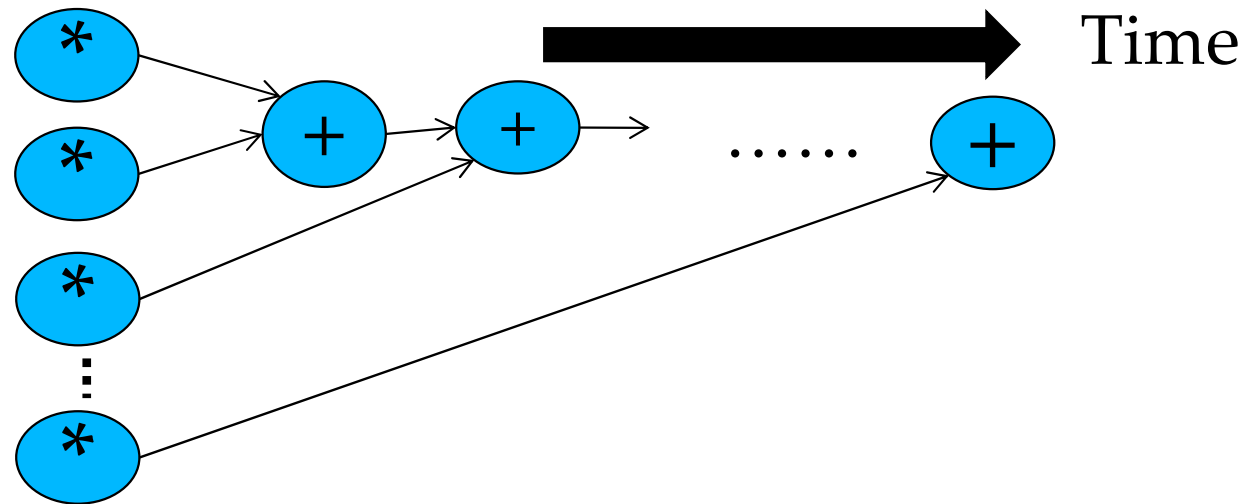
- Massively parallel execution cannot afford serialization
- Contentions in updating critical data causes serialization and delays



Simple Example Based on Resource Limits

Consider an **inner product** of two **vectors with one million elements**.

- All **multiplications** can be done **in parallel (one time unit)**.
- Only one adder, so **additions are serial (one million time units)**.



How Much can Conflicts Hurt?

- Amdahl's Law: if a **fraction F** of a computation is **serialized**, parallel **speedup cannot exceed $1/F$** .
- In the previous example, **$F = 50\%$** (the additions)
 - 50% of computation is serialized
 - **No more than $2\times$ speedup**, no matter how many multipliers are used!

Global Memory Bandwidth

Ideal



Reality



Balancing a System in 2013

Kepler (in Blue Waters, 2013)

- **1.3 TFLOPS** DPFP* peak throughput, and
- **250 GB/s** peak off-chip memory access bandwidth,
- which means **31.25 G DPFP operands per second**.

To achieve peak throughput, a program must perform

- **$1,300/31.25 = \sim 41.6$ FP** arithmetic operations
- **for each operand fetched** from off-chip memory.

*double-precision floating-point

Balancing a System in 2018

Volta (2018)

- **7.0 TFLOPS DFPF** peak throughput, and
- **900 GB/s** peak off-chip memory access bandwidth (HBM),
- which means **112.5 G DFPF operands per second**.

To achieve peak throughput, a program must perform

- **$7,000/112.5 = \sim 62.2$ FP** arithmetic operations
- **for each operand** value fetched from off-chip memory.

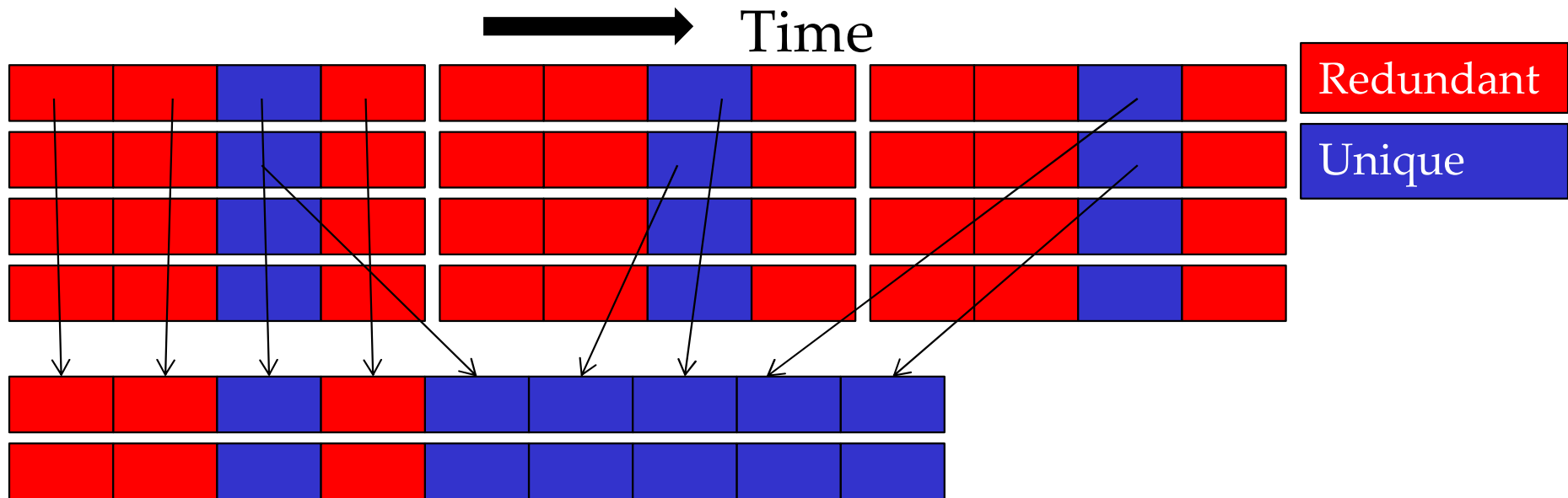
Trend: decreasing BW per FLOP; more required reuse.

Computational Efficiency Can Be Important

Parallel execution sometimes **requires redundant work**,
which may result in longer execution time.

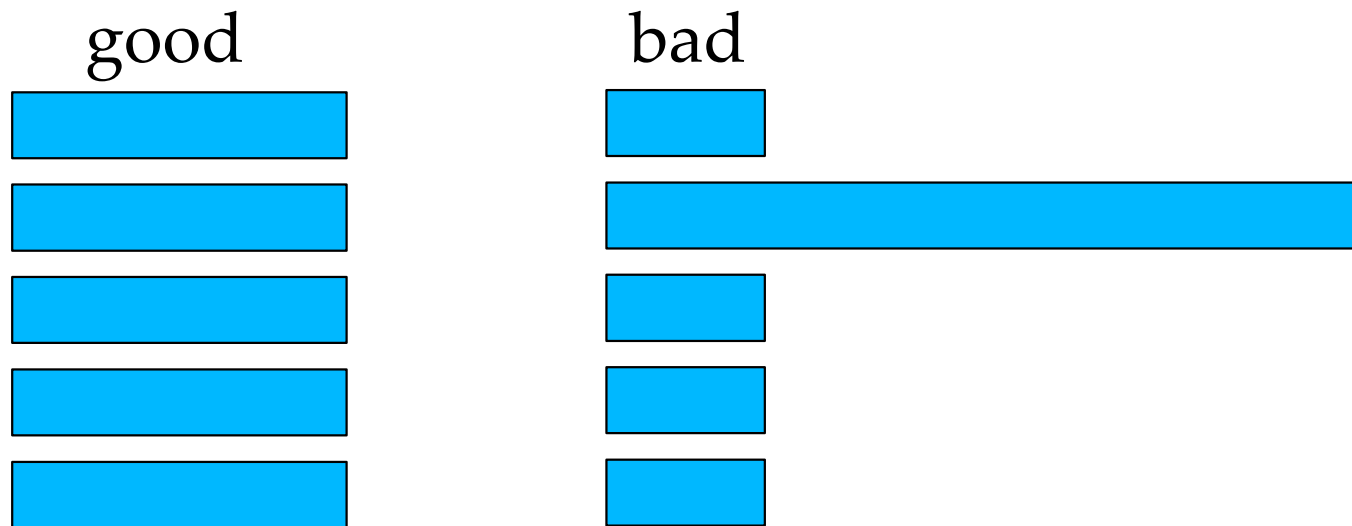
4-way
parallel

2-way
parallel



Load Balance: Must Wait for the Last Processor!

All threads must finish before a parallel job is complete.



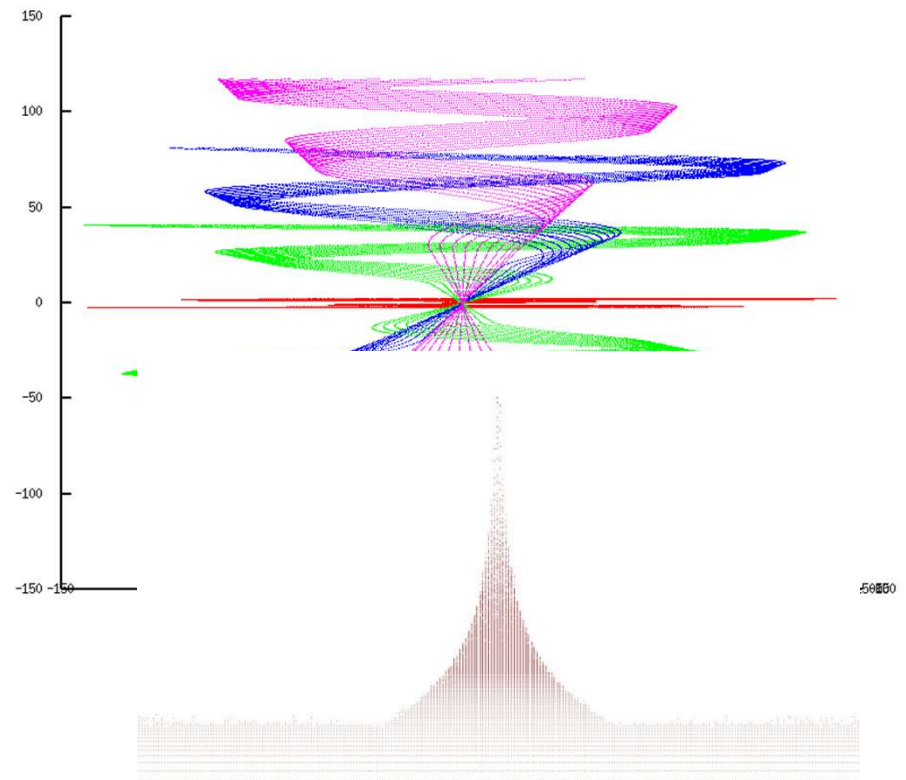
How Does Load Imbalance Affect Performance?

Say a job takes **100 units of time for one person** to finish.

- If we
 - break up the job into **10 parts of 10 units** each, and
 - have **10 people** to do it in parallel,
 - we can get a **10× speedup**.
- If we
 - break up the job into **50, 10, 5, 5, 5, 5, 5, 5, 5, 5 units**,
 - the same **10 people** will take **50 units to finish**,
 - with 9 of them idling for most of the time.
 - We get only a **2× speedup**.

What Causes Load Imbalance?

- Non-uniform data distributions
 - Highly concentrated spatial data areas
 - Astronomy, medical imaging, computer vision, rendering, ...
- If each thread processes the input data of a given spatial volume unit, some will do a lot more work than others



Eight Algorithmic Techniques

Technique	Contention	Bandwidth	Locality	Efficiency	Load Imbalance	CPU Leveraging
Tiling		X	X			
Privatization	X		X			
Regularization				X	X	X
Compaction		X				
Binning		X	X	X		X
Data Layout Transformation	X		X			
Thread Coarsening	X	X	X	X		
Scatter to Gather Conversion	X					

IEEE Computer, Stratton et al., August 2012



ANY QUESTIONS?