Lecture 26:
Class Wrap Up

CMU 15-418: Parallel Computer Architecture and Programming (Spring 2012)

Announcements

- Please complete course and TA evaluations

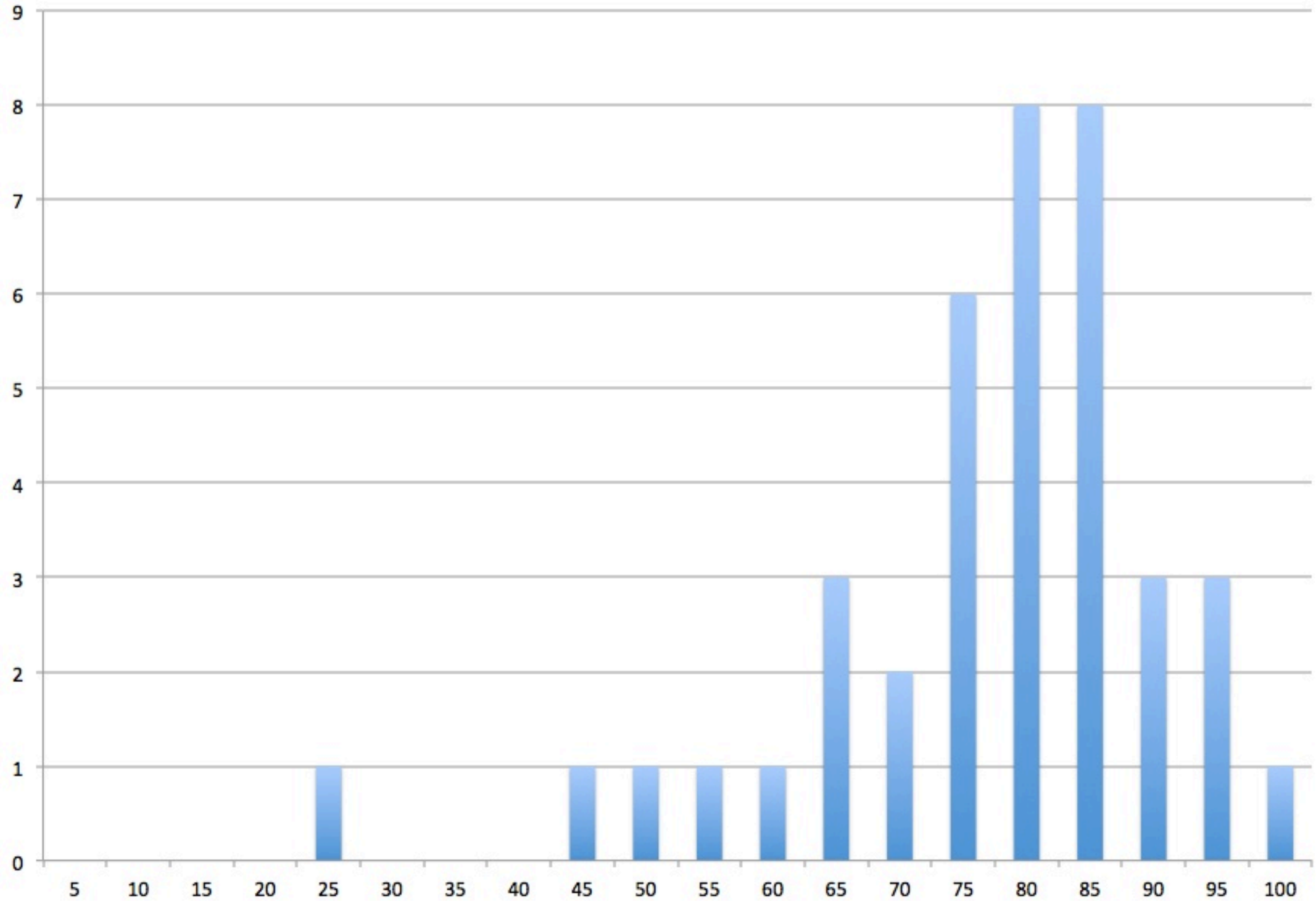
- Parallelism competition
  - Thursday May 10th, 8:30-11:30 AM
  - Scaife Hall 125

Today

- Exam 2 discussion
- Parallelism competition hints/guidelines
- Wrap up: a few final comments about parallelism

Exam 2

Average: 74 (std-dev: 15)



Exam 2 discussion

(whiteboard)

Project presentation tips

Presentation format

- Each group has 6.5 minutes
- Presentation format is up to you: slides, code demo, etc.
  - Start with name(s) of students, title of project
  - Both students in a 2-person group should speak
- Present off your own laptop, or make arrangements with staff
- Judges will make their way from group to group
- Rest of the class can follow along and listen, or make their own rounds (like a poster session, without posters)

Your #1 priority should be to be clear, rather than be comprehensive

(the writeup is for completeness)

Everything you say should be understandable by someone in this class. If you don't think the audience will understand, leave it out or change. (spend the time saying something we will understand)

This will be much harder than it seems.

Here are some tips to help.

Tip #1: consider your audience

- Everyone in the audience knows about parallel programming
  - CS terminology/concepts need not defined

- Most of the audience knows little-to-nothing about the specific application domain or problem you are trying to solve
  - Application-specific terminology should be defined or avoided

- The judges are trying to figure out the "most interesting" thing that you found out or did (your job is to define most interesting for them)

Tip #2: basic setup

- What is the problem?
  - What are the inputs, and the outputs?

- Why does this problem stand to benefit from optimization?
  - "Real-time performance could be achieved"
  - "Researchers could run many more trials, changing how science is done"
  - "It is 90% of the execution time in this particular system"

- What are the fundamental challenges to optimization?
  - What turned out to be the hardest part of the problem?
  - This may involve describing a few key characteristics of the workload (e.g., overcoming divergence, increasing arithmetic intensity)

Tip #3: pick a focus

- In this class, different projects should stress different results

- Some projects may wish to show a flashy demo and describe how it works (proof by "it works")

- Other projects may wish to show a sequence of graphs (path of progressive optimization) and describe the optimization that took system from performance A to B to C

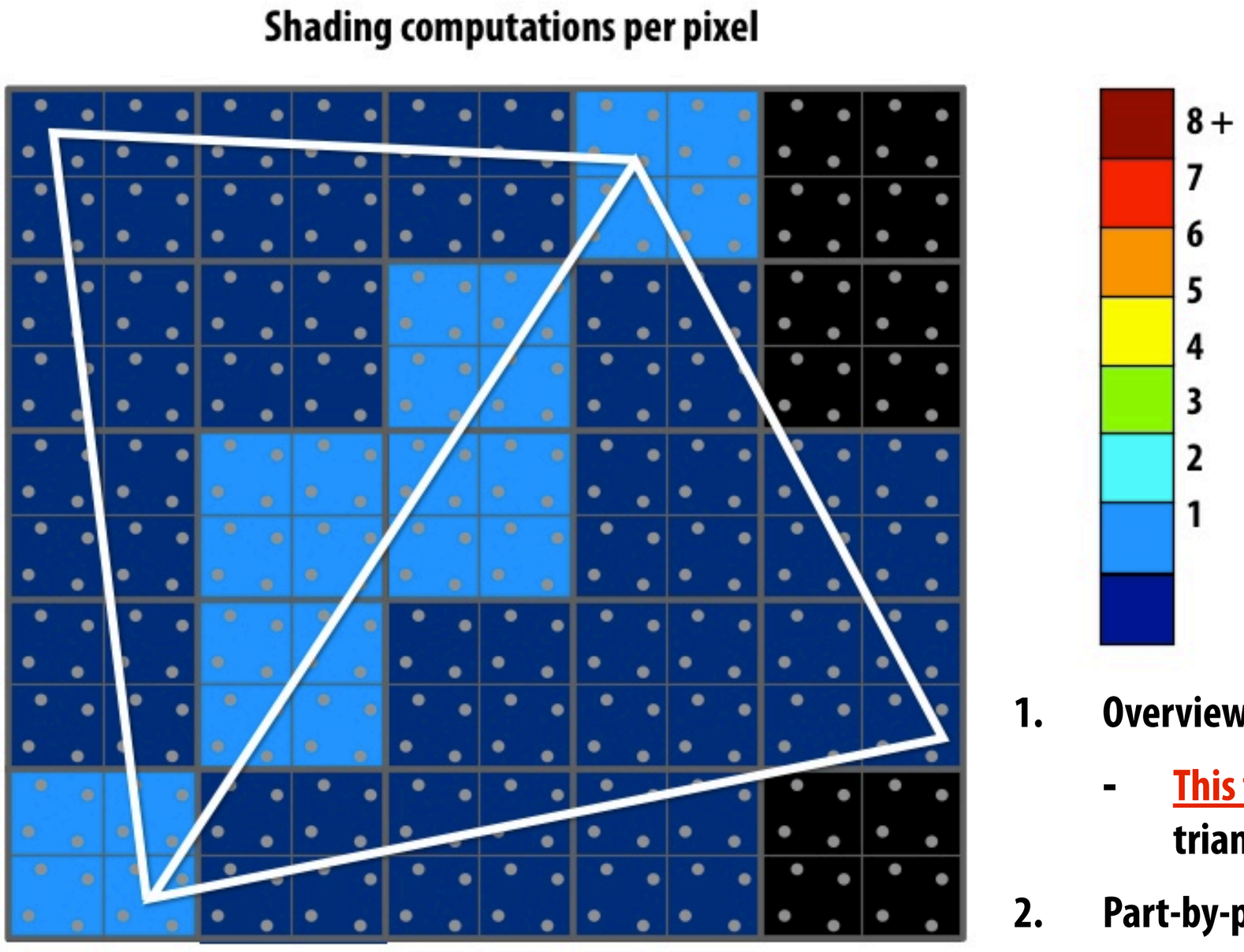
- Other projects may wish to clearly contrast parallel CPU vs. parallel GPU performance

Tip #4: how to describe a system

- Start with the nouns (the key boxes in a diagram)
  - Major components (processors, memories, interconnects, etc.)
  - Major entities (particles, neighbor lists, pixels, pixel tiles, features, etc.)
  - What is the state associated with each noun?

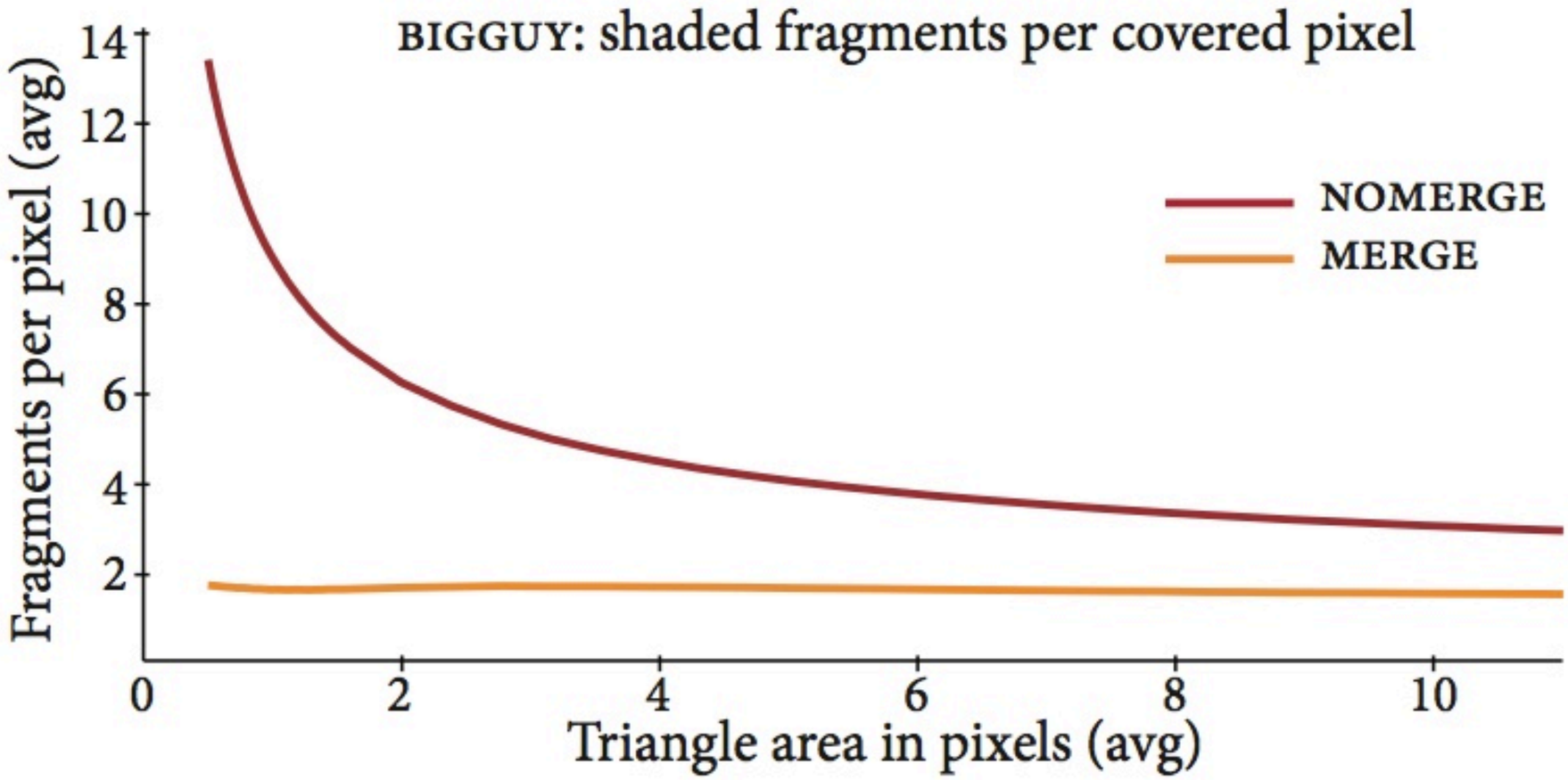
- Then describe the verbs
  - Operations that can be performed on the state (update particle positions, compute gradient of pixels, traverse graph, etc.)
  - Operations produce, consume, or transform entities

Tip #5: explain every figure or graph



1. Overview
   - This figures shows the effect of rasterizing two triangles
2. Part-by-part explanation:
   - Pixels are the boxes, they are colored according to the number of fragments generated ...
   - The sample points are given by the dots
3. Point: as you can see pixel ...

Tip #5: explain every figure or graph



1. In this graph, the X-axis is _____.
2. The Y-axis is _____.
3. If you look at the left side ...
4. So the trend that you see means ...

Common error: only explaining the result (the point) of the graph

Tip #6: provide evidence that your code is fast

- Compare against published results
  - "Our code is 10% faster than this publication"

- Determine a fraction of peak
  - "We achieve 80% of peak performance on this machine"

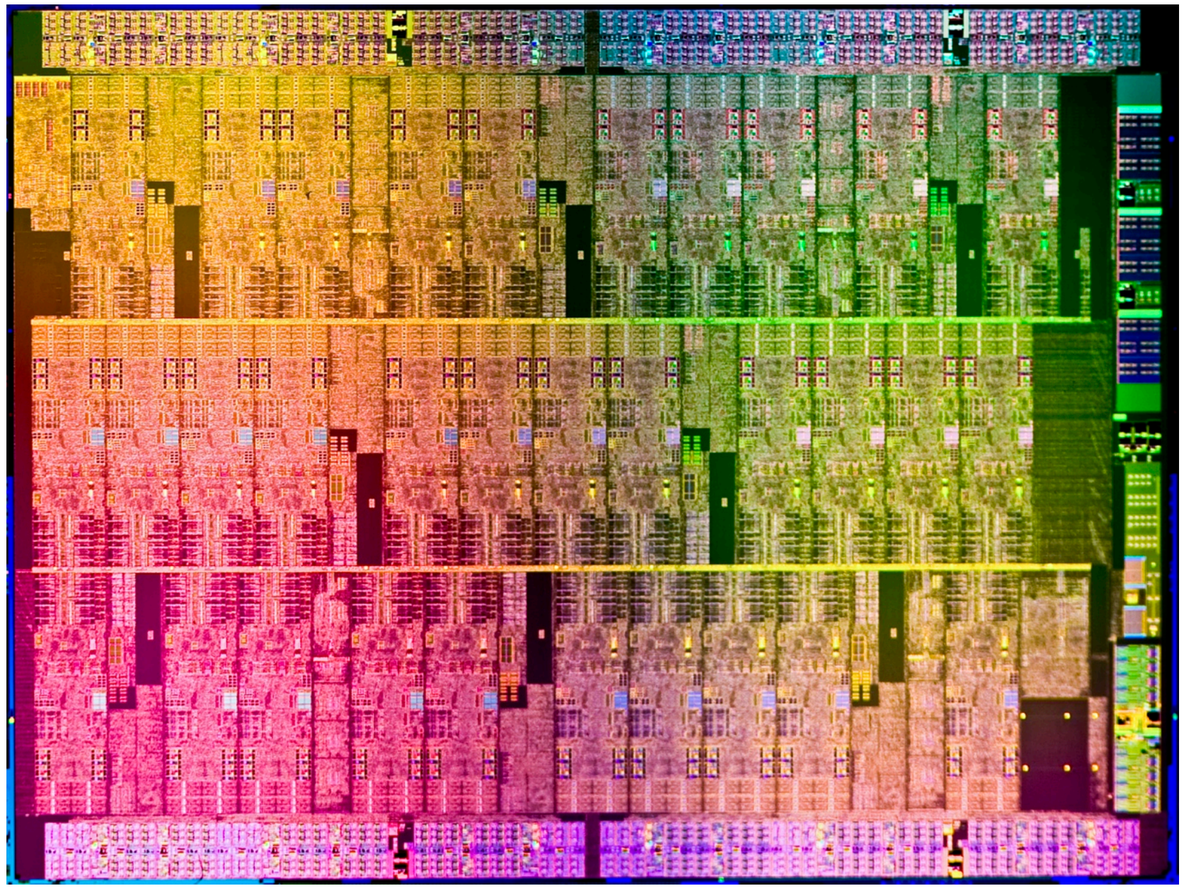
- Be truthful about comparisons between a CUDA implementation utilizing an entire GPU and single threaded, non-SIMD C program on a CPU (I'd rather not hear the conclusion: "GPU is 100x faster than the CPU".)

Tip #7: practice the presentation

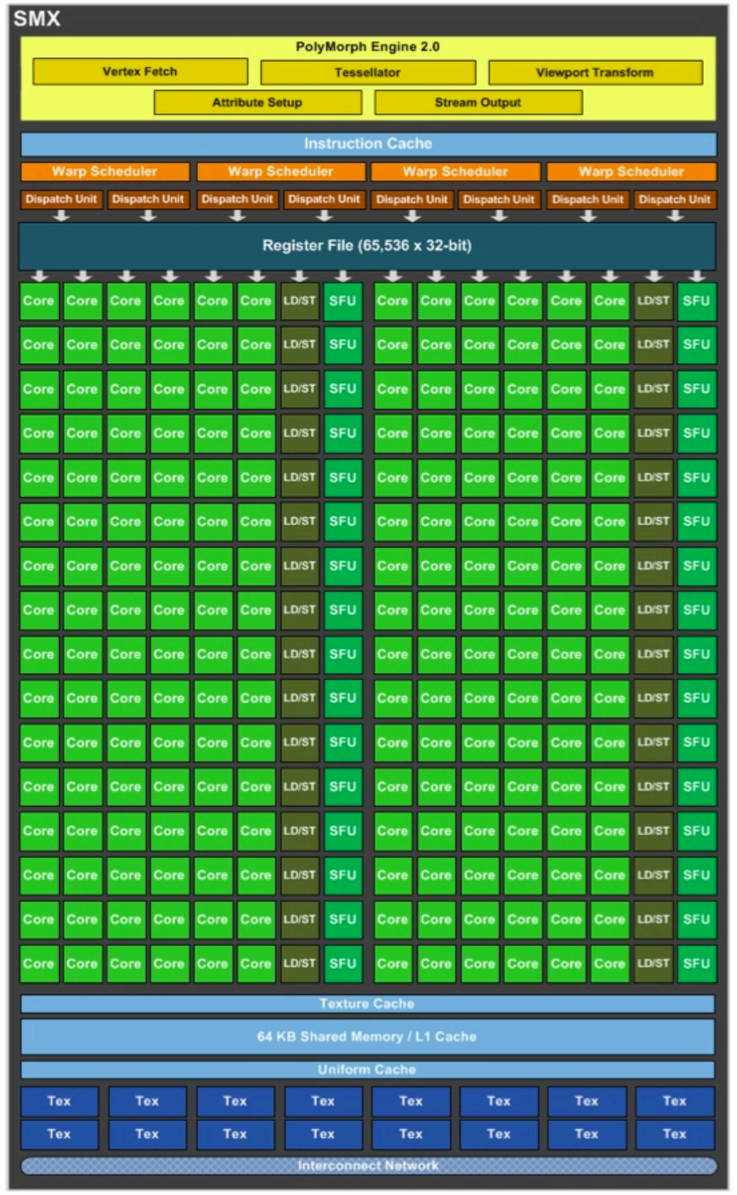
- Given the time constraints, you'll need to be smooth to say everything you want to say
- To be smooth you'll have to practice
- I hope you rehearse your demo or presentation a few times the night before (in front of a friend or two that's not in 418)

Wrap up

For the foreseeable future, the primary way to obtain higher performance computing hardware is through increased parallelism and (likely) hardware specialization.



Intel "Knight's Ferry", 32 cores, 16-wide SIMD



NVIDIA Kepler Core (SMX)
32 wide SIMD, up to 2024 CUDA/core threads

Today's software is surprisingly inefficient compared to the capability of modern machines

A lot of performance is currently left on the table (increasingly so as machines get more complex, and parallel processing capability grows)

Extracting this performance stands to provide a notable impact on many compute-intensive fields. (or enable new applications of computing!)

Given current software programming systems and tools, understanding how a parallel machine works is important to achieving high performance.

A major challenge going forward is making it simpler for programmers to extract performance on these complex machines.

Three big issues

Identifying parallelism
(or conversely, identifying dependencies)

Overcoming bandwidth limits
(or conversely, exploiting locality)

Dealing with latency

We addressed these issues at many scales and in many contexts

Single chip, multi-core CPU
Multi-core GPU
CPU+GPU connected via bus
Large scale, multi-node supercomputer

Consider the latency hiding vs. memory footprint trade-off: appeared in the context of CUDA threads, a parallel web-server, and then in parallel compilation on Exam 2, Q6.

Thanks for being a great class!

Good luck on projects. Expectations are high.

See you Thursday!