Administrivia

• HW2 out, due Nov. 2.
• HW3 will be out by Nov. 2.
• A comment on comments…
• Get OpenCL set up on your machine, or on Odyssey.
• Final Project Proposals due Nov. 6.
Comments in HW

Are Awesome.

They tell us what you are thinking (partial credit).

They make grading easier (extra credit).

Your future self will thank you.
# Dear Future Self,
#
# You're looking at this file because
# the parse function finally broke.
# It's not fixable. You have to rewrite it.
# Sincerely, Past Self

Dear Past Self, it's kinda creepy how you do that.

Also, it's probably at least 2013. Did you ever take that trip to Iceland?

Stop judging me!
OpenCL Setup

- https://github.com/harvard-cs205/OpenCL-test-setup
- On your machine: python test.py
- On Odyssey: sbatch run_test.sh
  - I've added in the missing lines to run_test.sh
Final Projects

• Read over cs205.org Projects page, look at projects from previous years.

• Proposals will be due Nov. 6, we’ll provide reviews back by the 9th.
Project Proposal

• ~1 page
• Team Members
• Background and Motivation
• Objectives - Functionality and Performance
• Design Overview - Technologies and use of Parallelism
• Verification
• Schedule, Milestones, and Division of Work
Team Members

- 2-3 people.

- 1 or 4 people with permission from CS205 staff.
Background & Motivation

• Motivation and reasons for proposing this project.
• Why are you interested in it?
• Background research you've done.
  • It's fine to repeat work!
Objectives

- Functionality
  - What will your project do?
  - What features are must-have, and which are optional?

- Performance
  - How fast will it be?
  - Realtime / Interactive / Coffee-break / Lunch / Days?
Design Overview

• What technologies from class will you use?

• It's fine to include extensions of Spark, or Cython, or OpenCL, but…

• You do need to explain where you will apply what you've learned from this class.

• Where is the parallelism in the algorithm?
Verification

• How will you make sure your code is working correctly?

• Do you have test data, serial implementations, known results, … ?
Schedule & Milestones

• Have at least three Milestones.

• You have one month (Nov. 9 - December 7), so one milestone per week.

• You'll have an assigned TF, you'll need to send them reports, and meet at least once in the middle.
Who does What?

• Set this out beforehand.

• Make sure everyone agrees.

• You're responsible for keeping each other on schedule, helping out, etc.

  • Groups are graded together.

• If there are problems, let us know earlier rather than later.
OpenCL
platforms = cl.get_platforms()
platform.name
platform.vendor
platform.version
platform.get_devices():

device.name
device.type
    # cl.device_type.to_string(device.type)
device.max_clock_frequency
device.max_mem_alloc_size  # (bytes)
device.max_work_group_size
device.max_work_item_dimensions
device.max_work_item_sizes
Apple Apple version: OpenCL 1.2 (May 10 2015 19:38:45)
The devices detected on platform Apple are:
-----------------------------
Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz [Type: CPU ]
Maximum clock Frequency: 1700 MHz
Maximum allocable memory size: 2147 MB
Maximum work group size 1024
Maximum work item dimensions 3
Maximum work item size [1024, 1, 1]
-----------------------------
HD Graphics 5000 [Type: GPU ]
Maximum clock Frequency: 1100 MHz
Maximum allocable memory size: 402 MB
Maximum work group size 512
Maximum work item dimensions 3
Maximum work item size [512, 512, 512]
Image Processing in OpenCL:

Dealing with Halos
https://github.com/harvard-cs205/OpenCL-examples

load_halo.cl
Have some local computation over a window of pixels.

(The blue pixel needs information from all of the pixels in the red box.)
Don't want to read global memory multiple times.

So we copy it to a buffer in local memory.
Our work group is the green square.

One thread ("work-item") per output pixel.
How large does the local buffer need to be?
(work group size) + (Input Size - 1)
These pixels are the work group's Halo.
How should we get this from global to local memory?

Remember, Coalesced Reads (adjacent threads read adjacent memory) are good.
Loop over rows, each thread reads/writes one element.
Loop over rows, each thread reads/writes one element.
Loop over rows, each thread reads/writes one element.
Loop over rows, each thread reads/writes one element.
https://github.com/harvard-cs205/OpenCL-examples

load_halo.cl
Next Time

• GPU memory - bank conflicts

• Profiling Kernels