Slide 1

Administrivia

- In the original version of the Java examples and Homework 2, I have everything in package `csci3366sample`. I've refactored(?) to have package `csci3366` with subpackages `sample` and `hw2`. You don't *have* to do this, but I think it makes sense?

Slide 2

Minute Essay From Last Lecture

- Some people had progress to report on this huge next homework; others reported not starting because they had other work due. Hm. I'm having second thoughts about not asking you to turn it in in installments!

Slide 3

OpenCL — Review/Recap

- OpenCL was intended to be very portable but also not to hide too much from the programmer.
- As a result, programmers must deal with a lot of low-level details.
- The good news is that a lot of those details are the same from program to program. So I wrote some library functions (see functions in `utility.h`). Okay for you to use them if you promise to read and (try to) understand!

Slide 4

OpenCL Examples, Continued

- Last time we looked in some detail at an example to add vectors.
- I also wrote something I call “semi-hello”, which uses OpenCL functions to find information about available devices and prints the result. (Review briefly.)

Slide 5

Numerical Integration in OpenCL

- What does our familiar example look like in OpenCL?
- A first thing to note is that OpenCL provides only a single-precision floating-point type(!).
- It doesn't seem quite right, then, to compare results with code that computes using double precision. So I first wrote a version of the sequential code that uses `float` rather than `double`, and . . . .
- Results were astonishingly bad! and in fact, they got worse with increasing numbers of samples! why . . .

Slide 6

Digression: `double` versus `float`

- I did some investigation in a previous year. Problem turns out to be that at some point in computing `sum`, the increments being added are so much smaller than the current value that they just disappear.
- What to do?! I found a clever algorithm which helps quite a bit. Will this be a problem in OpenCL? maybe or maybe not . . .

Slide 7

Numerical Integration in OpenCL, Continued

- Basic strategy — split iterations of the main processing loop among UEs and the combine results — is the same. UEs here are work items.
- We *could* make each loop iteration a work item (as in the vector addition example), but that might not work out too well — adding each tiny increment to a larger result seems like it would be a bottleneck. So adopt same strategy as for MPI and Java and have each work item compute several iterations.
- And then how to combine . . .

Slide 8

Numerical Integration in OpenCL, Continued

- Unlike OpenMP and MPI, OpenCL doesn't have anything built in to help with reduction. So we have to write our Something that complicates this example quite a bit is that combining results is not very easy in OpenCL — nothing built in. We can write our own (as we did in Java), but . . .
- Synchronizing among work items can be difficult: "Barrier" synchronization is available within each work group, but there's no way to apply it across work groups(!).
- So our strategy . . .

Numerical Integration in OpenCL, Continued

- So our strategy will be multi-level . . .
- First compute a partial sum in each work item (similar to what we did in MPI and Java).
- Then combine these into one partial sum for each work group (using barrier synchronization — wait for all work items to compute their partial sums and then have one work item combine them).
- Then have the host combine these per-work-group sums into the final sum.

Numerical Integration in OpenCL, Continued

- To do this we'll need to work with different levels of memory:
- Sums for work items can go in an array shared among work items but local to a work group (this is the "local memory" previously mentioned).
- Sums for work groups need to be in "global memory" (accessible to host as well).

Numerical Integration in OpenCL, Continued

- One other thing to know about before looking at code: When executing kernel, the number of work items (indices) has to evenly divide the workgroup size.
- That said, look at code . . .
- (An interesting thing about this environment is that it's not clear what "scalable" means. Things that could vary are number of loop iterations per work item and workgroup size. I've made both of changeable via command-line arguments.)

Minute Essay

- So how's Homework 2 coming along? anything interesting to report?
- (Other questions?)