Teacher instructions

See slide presentation and student instructions.

Lesson is designed as a student activity in which students are given a working heat diffusion code in 1-D, and asked to modify for 2-D. The Slide deck covers both an explanation of the relaxation method used to solve the problem as well as the syntax for 2D indexing of blocks and grids in CUDA.

Solved code is provided.

When working with the students to parallelize their code, it can be useful to discuss a few key issues.

- Asking for more resources, either in total or in certain manners (e.g. too many threads per block) will not only not produce an error, it will likely run with great speed (though erroneous results) fooling the students into thinking they have sped up their problem. Focus on the need for always checking the accuracy of results and checking for errors, especially when debugging CUDA code.
- Depending on machine, you may need specific compile options. Students may investigate setting the compute level and architecture at compile time to see if it speeds up, slows down, or breaks their code.
- Have the students investigate pairing up the rapidly changing variable in the flattened array with either the x or y dimension in thread space for the 2D problem, to see if there is a performance difference.

Common Pitfalls for Students and Instructors

Debugging CUDA can be difficult. Memory errors in CUDA may result in an abnormal program termination with no error, or it might just fail to write to memory but otherwise run with no error. Printing from within kernels may require compiling with a recent architecture type, as early versions of CUDA did not allow printing from within kernels. It is vitally important to have some level of check on CUDA programs to determine whether the results are correct, particularly during program design.

Students should be sure to remember that any memory passing back and forth from host to device must have copies on both, be allocated on both, and be freed on both.

Students should remember that the kernel replaces the loop with an element-based view, it doesn’t provide a new location in which to write a loop over elements.