CSE 415: Introduction to Parallel Computing
Spring 2018, Homework 6
Due 8:00 AM, May 3, 2018

This homework is 20% of your ***homework grade****.*

1) Suppose that you have a vector with 1 million single precision float values, and you have implemented a GPU kernel that calculates the multiplicative inverse (1/x) of each element in this vector.
   a) Using a thread block of size 128, write down how you would setup your CUDA grid and launch the CUDA kernel?
   b) How many threads will be generated in the grid if you launch your kernel using the minimum number of thread blocks necessary to cover your entire array?
   c) If streaming multiprocessors on your GPU supports 2,048 threads and 8 thread blocks, how would you modify your kernel launch to maximize device occupancy?

2) You need to write a kernel that operates on an image. You decided to use 2D thread blocks and you would like each thread to process a 2 pixel by 2-pixel area.
   a. Write down the expression to map a thread to the top left corner of the area it needs to process in your kernel, and give the code that will ensure a threads to process its assigned pixels.
   b. You would like your thread blocks to be square. If streaming multiprocessors on your GPU can support up to 2,048 threads and 8 thread blocks, what thread block dimensions can you use in launching your kernel while ensuring full device occupancy?
   c. If your images are 256x256 pixels and your GPU has 16 streaming multiprocessors, among your options in part b, which one would you choose and why?

3) For this problem, you will write a CUDA code for 1D diffusion modeling (see below for further explanations) based on the sequential CPU implementation you are given. You are asked to only submit the source code (you do not have to execute the code and collect performance data at HPCC), but make sure to include the appropriate kernel launching commands and memory allocations in your code.

GPU-accelerated Modeling of Laplacian Diffusion

Credit: *The problem statement and sequential codes for this assignment are courtesy of Prof. Greg Wolffe of Grand Valley State University.*

Overview

In this assignment, you will write a CUDA-based massively multi-threaded application that computes and models (heat | density) diffusion.

Introduction

Diffusion is the movement of a substance across a concentration gradient; it continues until a (temperature / pressure / density) equilibrium has been reached. The diffusion equation is a partial differential equation (PDE) that describes density dynamics. If D, the diffusion coefficient (a term describing the rate of diffusion) varies depending on the density, then the equation is nonlinear. However, if the rate of diffusion is a constant, then the diffusion equation reduces to a linear equation, the well-known heat distribution equation.

The Heat Equation in 1D

Consider an insulated, conducting rod, initially at room temperature (23° C). At time t=0, one end of the rod is placed in a beaker of boiling water (100°C). Obviously, heat will be transferred from regions of higher temperature to lower temperature. The heat distribution equation models the temperature changes over time.

Assume that the density of the rod and its thermal conductivity are constant. Consider an infinitely thin slice x (width = Δx) of the rod, such that its temperature u is uniform. Since the rod is insulated, the change in temperature at location x over time Δt is equal to the "heat in" from its left boundary minus the "heat out" at its right boundary. This is given by the partial differential equation:

$$\frac{\partial u}{\partial t} = \mathrm{K}\frac{\partial^2 u}{\partial x^2},$$

where K is the diffusivity constant. It is typically used to compute the temperature u at point x at time t. With judicious choice of the time scale T and the length of the rod L, the diffusivity constant vanishes. Continuing, the Central Difference theorem can be used to derive an approximate numerical solution to the second-order derivative:

$$\frac{\partial^2 u}{\partial x^2} = \frac{dt}{dx * dx}\left[u(x+dx) - 2u(x) + u(x-dx)\right]$$

Rearranging and expressing as a discrete equation gives:

$$u_{t+1}[i] = \frac{u[i+1] + u[i-1]}{2},$$

where i is the index of point x. This finite difference equation says: The temperature at point x in the next time step is equal to the average of its two neighbors at the current time step. Computed over very small differences in time and very small distances along the rod, this method converges to provide an arbitrarily accurate approximation to the heat equation.

As with many numerical methods, the algorithm runs to convergence. This can be determined via a global notion of stable temperature, or when the temperature change over two units of time at all points is less than some small value ε. As noted, it is often used to determine the temperature at a specified point at a specified time.

This basic idea can easily be extended to two or more dimensions, but for simplicity we will stick with a 1D version. A sequential diffusion code is given to you in the diffusionSeq.c. One characteristic of the finite difference solution to the diffusion problem that can be exploited when implementing a parallel solution is the independence of the calculations. They may be performed in any order and still be correct (i.e. there are only local data dependencies). Of course, you still need to isolate old/new temperature/density values. Write a CUDA version of the sequential code you are given, together with the necessary memory allocation/deallocations, data transfers and kernel launches.