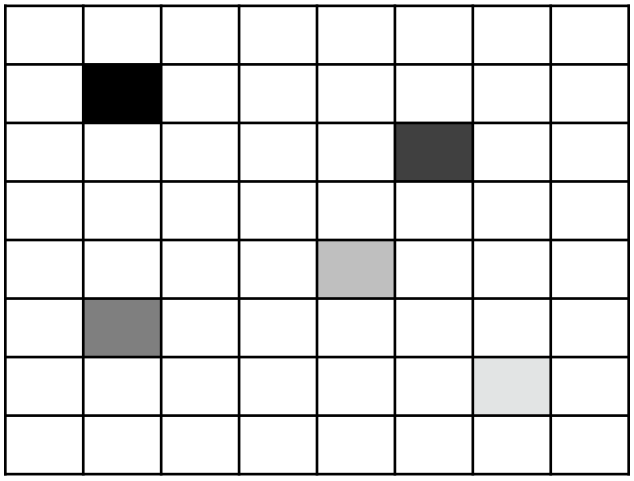
CSC 431: High Performance Computing

Simulating Heat Transfer (taken from CUDA by Example by Sanders and Kandrot)

Physical simulations can be among the most computationally challenging problems to solve. Fundamentally, there is often a trade-off between accuracy and computational complexity. As a result, computer simulations have become more and more important in recent years, thanks in large part to the increased accuracy possible as a consequence of the parallel computing revolution.

We will look at a physical simulation that is fairly straight forward to parallelize: two-dimensional heat transfer.

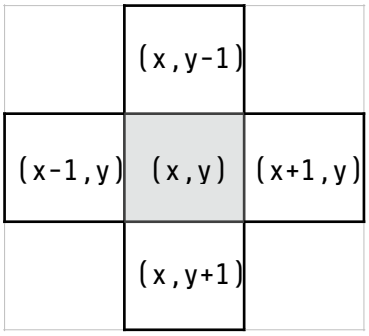
Assume we have some rectangular room that we divide into a grid. Inside the grid, we will randomly scatter a handful of "heaters" with various fixed temperatures.



Given a rectangular grid and configuration of heaters, we are looking to simulate what happens to the temperature in every grid cell as time progresses. For simplicity, cells with heaters in them always remain a constant temperature. At every step in time, we will assume that heat "flows" between a cell and its neighbors. If a cell's neighbor is warmer than it is, the warmer neighbor will tend to warm it up. Conversely, if a cell has a neighbor cooler than it is, it will cool off.

In our heat transfer model, we will compute the new temperature in a grid cell as a sum of the differences between its temperature and the temperature of its neighbors:

$$T_{new} = T_{old} + \sum_{neighbors} k \times (T_{neighbor} - T_{old})$$



(x,y) == T_{old}, T_{new}

(x-1,y) == T_{left}

(x+1,y) == T_{right}

(x,y-1) == T_{top}

(x,y+1) == T_{bottom}

In the equation for updating a cell's temperature, the constant k simply represents the rate at which heat flows through the simulation. Since we will consider only the four neighbors (top, bottom, left, right) and k and T_{old} remain constant, the equation can be expressed as follows:

$$T_{new} = T_{old} + k \times (T_{top} + T_{bottom} + T_{left} + T_{right} - 4 \times T_{old})$$

Your assignment is to create a sequential and pThreads version of the heat transfer simulation.

Grading:

Sequential (heat_sequential.c)

15% - Sequential Heat Transfer Simulation (mostly in class)

10% - Compute the number of steps to reach a steady state
➟ command line argument (delta threshold)
./heat_sequential 0.001

Parallel (heat_pthreads.c)

40% - pThreaded Heat Transfer Simulation
➟ graphical output (like sequential)
➟ make sure all threads have finished before graphically displaying a step
➟ In header comment, explain how you parallelized: by rows? by columns? square blocks? and why?

10% - Compute the number of steps to reach a steady state
➟ command line argument (delta threshold)
./heat_pthreads 0.001

Better than Average

5% - Header comment, author, correct tabbing, comments

5% - Appropriate use of constants (#define)

8% - A useful, colorful scheme for representing temps

7% - Comparison of sequential vs. parallel
Included in the header comment in heat_pthreads.c:
# of ms to find steady state (sequential)
# of ms to find stead state (parallel, t = 2)
# of ms to find stead state (parallel, t = 4)
# of ms to find stead state (parallel, t = 8)
room size = (800x800), ambient = 0.5,
heater_size = 200, single heater center on (400, 400)

Pseudocode (Sequential):

- Create two grids (2D arrays): old and new
- Initialize the old grid to the **ambient** temperature of the room
- Set the edges of the old and new grid to the **cold** temperature
- Randomly create **heater_count** number of heaters in the room
    - Each heater is a square of size **heater_size**
    - Heater temps are randomly between **heater_min** and **heater_max**
    - Heaters should be fully inside the room
    - Store the locations and temperatures of the heaters
- Loop over many **steps**
    - Show the old grid to the user (instructor supplied graphics)
    - Using as the old grid as input, compute the new grid temps
        - Only do computation on the interior cells
    - Using the stored locations, reset the heaters to constant temp
    - Copy the new grid into the old grid

* **Bold** words should be constants in your programs.