Universität Heidelberg
Institut für Technische Informatik

Holger Fröning
Benjamin Baumann

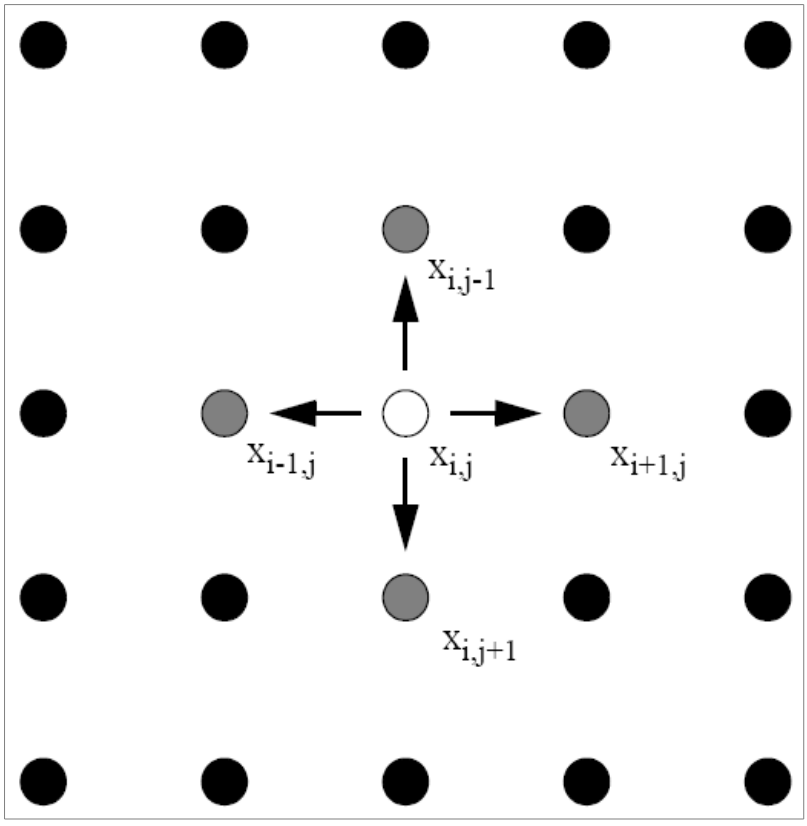
Introduction to High Performance Computing
Term 2014/2015 (Winter)

Exercise 5

- **Return electronically until Tuesday, 24.11.2014 23:55 Uhr**
- **Include name and your account (introhpc[xx]) on the top sheet.**
- **A maximum of two students is allowed to work jointly on the exercises.**

5.1 Heat Relaxation – Sequential Implementation

Relaxation is a mathematical technique used for modeling or the simulation of dynamic processes (heat distribution, material yielding, etc.). In this technique, the solution of a multi-dimensional function is mapped to a discrete grid. The value of a given grid point is dependent on the values of the previous time step, usually of the point itself and its surrounding neighbors. Implement a program, which calculates the new value of grid points as average of the point itself and its four direct neighbors.



The value of a grid point for the next time step t+1 calculates as follows, note that i and j are the coordinates of this grid point:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \Phi \cdot ((-4) \cdot x_{i,j}^{t} + x_{i+1,j}^{t} + x_{i-1,j}^{t} + x_{i,j+1}^{t} + x_{i,j-1}^{t}) \qquad \Phi = \frac{24}{100}$$

- Dynamically allocate a grid of NxN double precision floating point values. The allowed value range is [0,127].

- Inject heat in the topmost grid points (j=0), with i in between [N/4, 3N/4]. These grid points are set statically to 127. The value of all other grid points at the borders is statically set to 0.
- The size of the grid N shall be configurable using a command line parameter.
- Write a suitable sequential program that performs the described calculation. Test your program extensively.
- Optionally: try to visualize the output to check correctness.

(25 points)

5.2 Heat Relaxation - Experiments

- Measure the average time for grid sizes of **128 x 128, 512 x 512, 1024 x 1024, 2k x 2k, 4k x 4k**. Report the average time of one iteration by performing for instance 100 iterations, measuring the time with a suitable function (e.g., *gettimeofday()* in Linux) and dividing by the number of iterations. Do not include time for initialization or output. Use compiler-specific optimizations to minimize the runtime.
- Determine the number of FLOPs achieved for each of the grid sizes from above.
- Is this program computationally bound or memory-bound? Provide a detailed explanation!
- Interpret results!

Grid size	Time/iteration	Flops total	GFLOP/s
128 x 128			
512 x 512			
1024 x 1024			
2k x 2k			
4k x 4k			

(15 points)

5.2 Heat Relaxation – Pre-considerations for parallelization

- In the next exercise we will parallelize this application using message passing (obviously). For preparation, develop now a suitable partitioning and task model. Propose a suitable model, and try to answer the following questions:
  - How would you decompose this problem into sub-tasks? Which dependencies between tasks do exist?
  - Is 1D or 2D partitioning more suitable?
  - Are there opportunities to leverage overlap between computation and communication?
  - How do you synchronize after each iteration?
- Explain in detail; include a drawing to visualize your approach!

(15 points)

Total: 55 points