Lab Course
Efficient Programming of Multicore Processors and Supercomputers

Michael Gerndt, Josef Weidendorfer

Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechnerarchitektur

Schedule Today

- Introduction
- Heat equation code
- SuperMUC at LRZ
- Groups/Accounts

Goals

- Overview of parallel architectures
- Understand sequential tuning
- Learn programming in MPI and OpenMP
- Know parallelization strategies
  - Domain decomposition
  - Task decomposition

Program

- Presentation
- Discussion of tasks and of your solutions
- Group meetings

Tasks

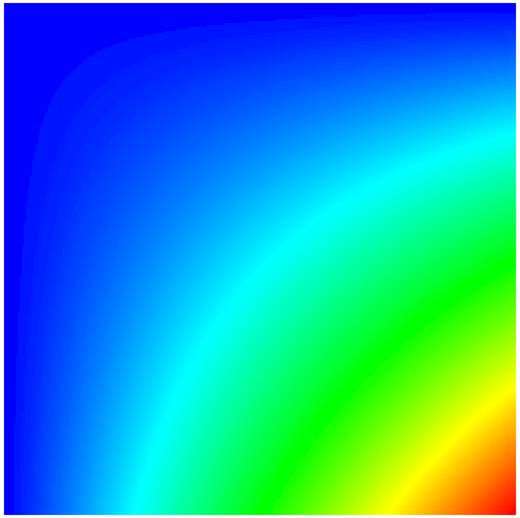
- Tasks
  - Sequential optimization of solver for heat distribution
  - OpenMP parallelization
  - MPI parallelization
  - Parallelization of Abalone

- Groups of three people

Grading

- Criteria
  - Stick to deadlines
  - Design and implementation
  - Readability of codes
  - Reports
  - Additional points for additional features
  - Presentation of your solution
- To do
  - Implement the solution
  - Write a small report.
  - Submit Report on Tuesday before the next meeting.
  - We will look at the report and select presenters for certain parts. You will be informed on Wednesday to prepare a presentation. Presentation could take us through the part of the report if you cannot do slides.

Heat Distribution in a Solid Body



Model

- The heat distribution in a 2d solid body can be modeled by the following partial differential equation (Poisson equation)

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = k * \frac{\partial u}{\partial t}$$

where u(x,y,t) is the temperature at point (x,y) at time t.

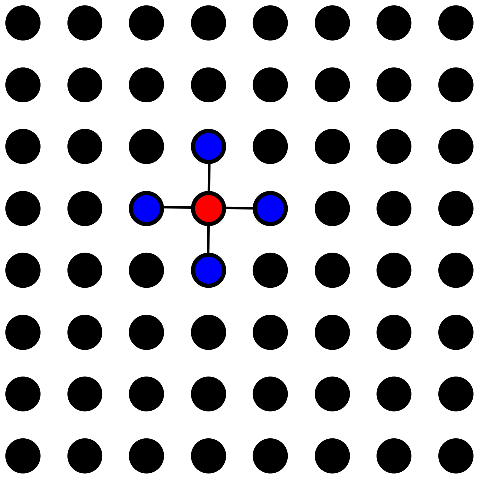
- In a stable state the change is 0 over time.

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0$$

- We solve the equation on the discretized unit square (0,1) x (0,1) with Dirichlet boundary conditions: fixed temperature on the boundary.

Solver Kernel

- Solver kernel for a simple partial differential equation.
- Grid $(n+2) \times (n+2)$
- Fixed boundaries
- Interior points are recomputed
- Simple iterative solver
  - Mean value of four neighbor points ("4 point stencil") until allover residuum is below threshold
  - Jacobi method
    – values from previous iteration
  - Gauss-Seidel method
    – new values of upper and left point
    – old values of lower and right point



Iterative Solution

Jacobi-Algorithm:

```
for (int j=1; j<=n; j++){
    for (int i=1; i<=n; i++){
        uhelp[i][j] = 0.25*( u[i-1][j] + u[i+1][j] +
                             u[i][j+1] + u[i][j-1]);
    }
}
```

Gauß-Seidel-Algorithm

```
for (int j=1; j<=n; j++){
    for (int i=1; i<=n; i++){
        u[i][j] = 0.25*( u[i-1][j] + u[i+1][j] +
                         u[i][j+1] + u[i][j-1]);
    }
}
```

Application Source

- On the home page of the lab course
- Configuration via input file

```
100       # iterations (0: until residuum < threshold)
300       # initial resolution of x/y discretization
3100      # max resolution
200       # resolution step size
0         # Algorithm 0=Jacobi 1=Gauss
2                      # number of heat sources
0.0  0.0  1.0  1.0     # (x,y), size, temperature
1.0  1.0  1.0  0.5
```

- Result
  - Residuum after specified number of iterations
  - Visualization via ppm file

Application Source

- Download and extract (tar xzf heat.tgz)

- check Makefile
- Build and run
  - make
  - ./heat test.dat