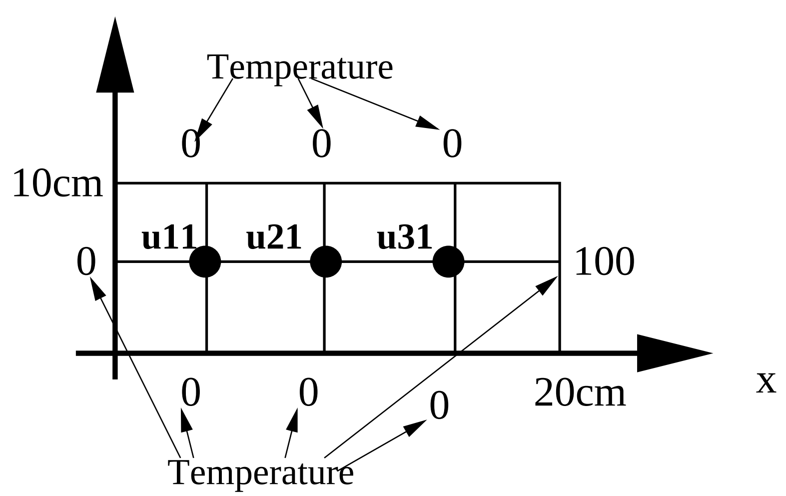
Use of parallel matrix algorithms

for Laplace partial differential equations

A steady-state heat-flow problem on a rectangular $10cm \times 20cm$ metal sheet.

One edge maintains temperature of 100 degree, other three edges maintain 0 degree. What are the steady-state temperatures at interior points?



The mathematical model

Laplace equation:

$$\frac{\partial^2 U(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0$$

with the boundary condition:

$$u(x,0) = 0, \quad u(x,10) = 0.$$

$$u(0,y) = 0, \quad u(20,y) = 100.$$

Finite difference method to solve this PDE:

- Discretize the region: Divide the function domain into a grid with gap h at each axis.
- At each point (ih, jh), let $u(ih, jh) = u_{i,j}$. Setup a linear equation using an approximated formula for *numerical differentiation*.
- Solve the linear equations to find values of all points $u_{i,j}$.

Approximating numerical differentiation

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \quad \text{or} \quad f'(x) \approx \frac{f(x)-f(x-h)}{h}$$

$$f''(x) \approx \frac{f'(x+h)-f'(x)}{h} \approx \frac{\frac{f(x+h)-f(x)}{h} + \frac{f(x)-f(x-h)}{h}}{h}$$

Thus

$$f''(x) \approx \frac{f(x+h)+f(x-h)-2f(x)}{h^2}$$

Then

$$\frac{\partial^2 u(x_i, y_i)}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

$$\frac{\partial^2 u(x_i, y_i)}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}$$

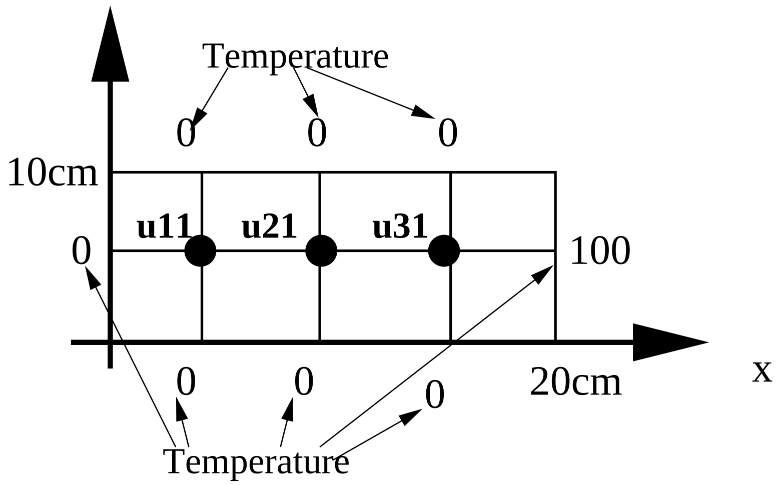
Adding the above two equations

$$u_{i+1,j} - 2u_{ij} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1} = 0$$

Then

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0$$

Example of Derived Linear Heat Equations



For this case: Let $u_{11} = x_1, u_{21} = x_2, u_{31} = x_3$.

$$\text{At } u_{11}, \quad 4x_1 - 0 - 0 - x_2 = 0$$

$$\text{At } u_{21}, \quad 4x_2 - x_1 - 0 - x_3 - 0 = 0$$

$$\text{At } u_{31}, \quad 4x_3 - x_2 - 0 - 100 - 0 = 0$$

$$\begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix}$$

Solutions:

$$x_1 = 1.786, \quad x_2 = 7.143, \quad x_3 = 26.786$$

Linear heat equations for a general 2D grid

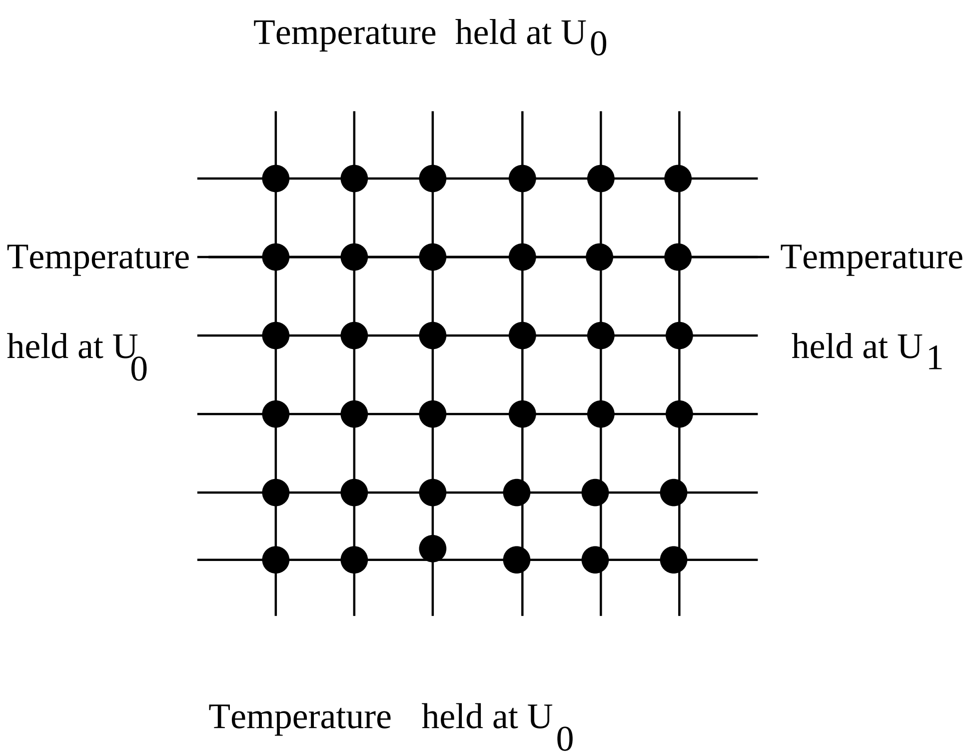
Given a general $(n+2) \times (n+2)$ grid, we have n^2 equations:

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0$$

for $1 \leq i, j \leq n$. Or express them as:

$$u_{i,j} = (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})/4$$

Example, $r = 2, n = 6$.



We order the unknowns as

$(u_{11}, u_{12}, \cdots, u_{1n}, u_{21}, u_{22}, \cdots, u_{2n}, \cdots, u_{n1}, \cdots, u_{nn})$

For $n = 2$, the ordering is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} u_{11} \\ u_{12} \\ u_{21} \\ u_{22} \end{bmatrix}$$

The matrix is:

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} u_{01} + u_{10} \\ u_{20} + u_{31} \\ u_{02} + u_{13} \\ u_{32} + u_{23} \end{bmatrix}$$

In general, the left side matrix is:

$$\begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & -I & T & -I & \\ & & \ddots & \ddots & \ddots \\ & & & -I & T \end{bmatrix}_{n^2 \times n^2}$$

$$T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 4 \end{bmatrix}_{n \times n}$$

$$I = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}_{n \times n}$$

The matrix is too sparse, direct methods for solving this system takes too much time.

The Jacobi Iterative Method

Given

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0$$

for $1 \leq i, j \leq n$.

The Jacobi program:

Repeat
    For i=1 to n
        For j=1 to n
            $u_{i,j}^{new} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})$.
        EndFor
    EndFor
Until $\| u_{ij}^{new} - u_{ij} \| < \epsilon$

Called **5-point stencil computation** as $u_{i,j}$ depends on 4 neighbors.

The Gauss-Seidel Method

Repeat

$u^{old} = u.$

For i=1 to n

For j=1 to n

$u_{i,j} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}).$

EndFor

EndFor

Until $\| u_{ij} - u_{ij}^{old} \| < \epsilon$

Parallel Jacobi Method

Assume we have a mesh of $n \times n$ processors.

Assign $u_{i,j}$ to processor $p_{i,j}$.

The SPMD Jacobi program at processor $p_{i,j}$:

Repeat

Collect data from four neighbors: $u_{i+1,j}$, $u_{i-1,j}$, $u_{i,j+1}$, $u_{i,j-1}$ from $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j+1}$, $p_{i,j-1}$.

$$u_{i,j}^{new} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}).$$

$$diff_{i,j} = |u_{ij}^{new} - u_{ij}|$$

Do a global reduction to get the maximum of $diff_{i,j}$ as M.

Until $M < \epsilon$

Performance evaluation

- Each computation step takes $\omega = 5$ operations.
- There are 4 communication messages to be received. Assume sequential receiving. Communication costs $4(\alpha + \beta)$.
- Assume that the global reduction takes $(\alpha + \beta) \log n$.
- The sequential time $Seq = K\omega n^2$ where K is the number of steps.
- Assume $\omega = 0.5, \beta = 0.1, \alpha = 100, n = 500, p^2 = 2500.$
- The parallel time $PT = K(\omega + (4 + \log n)(\alpha + \beta))$

$$Speedup = \frac{\omega * n^2}{\omega + (4 + \log n)(\alpha + \beta)} \approx 192$$

$$Efficiency = \frac{Speedup}{n^2} = 7.7\%.$$

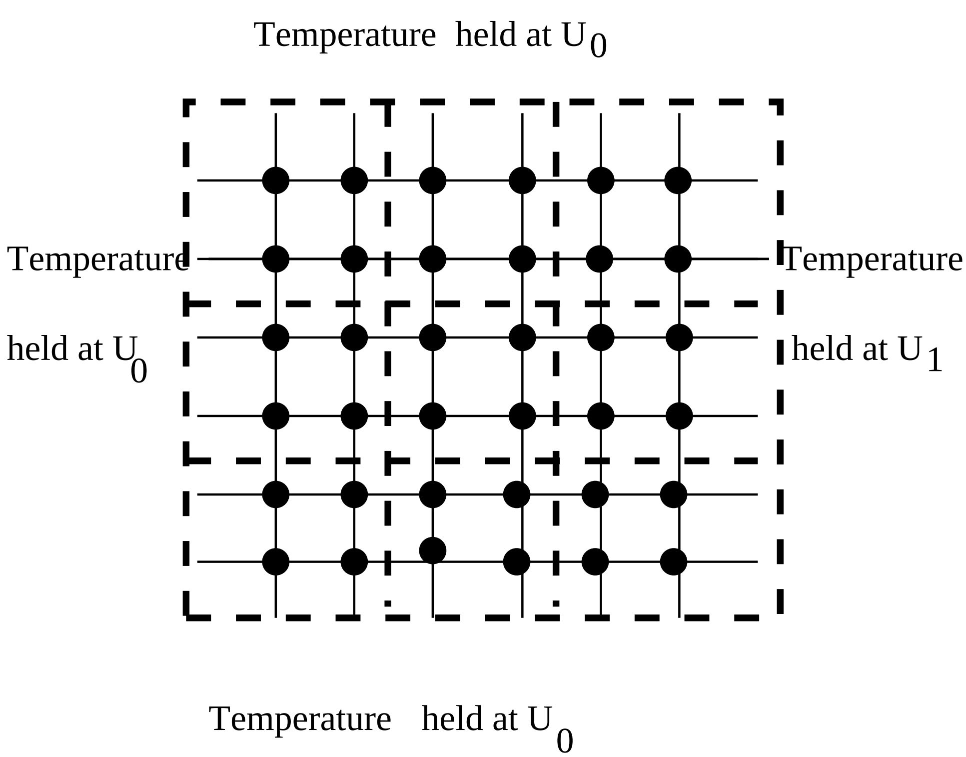
Grid partitioning

• Reduce the number of processors. Increase the granularity of computations.

• Map the $n \times n$ grid to processors using 2D block method.

Assume a $p \times p$ mesh,$\gamma = \frac{n}{p}$.

Example, $r = 2, n = 6$.



Code partitioning

Re-write the kernel part of the sequential code as:

For $bi = 1$ to p
  For $bj = 1$ to p
    For $i = (b_i - 1)\gamma + 1$ to $b_i\gamma$
      For $j = (b_j - 1)\gamma + 1$ to $b_j\gamma$
        $u_{i,j}^{new} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})$.
      EndFor
    EndFor
  EndFor
EndFor

Parallel SPMD code

On processor p_{b_i,b_j}:

Repeat

Collect the data from its four neighbors.

For $i = (b_i - 1)\gamma + 1$ to $b_i\gamma$

For $j = (b_j - 1)\gamma + 1$ to $b_j\gamma$

$u_{i,j}^{new} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})$.

EndFor

EndFor

Compute the local maximum $diff_{b_i,b_j}$ for the difference between old values and new values.

Do a global reduction to get the maximum $diff_{b_i,b_j}$ as M.

Until $M < \epsilon$

Performance evaluation

- At each processor, each computation step takes ωr^2 operations.
- The communication cost is $4(\alpha + r\beta)$.
- Assume that the global reduction takes $(\alpha + \beta)\log p$.
- The number of steps is K.
- Assume $\omega = 0.5, \beta = 0.1,\ \alpha = 100,\ n = 500, r = 100, p^2 = 25$.

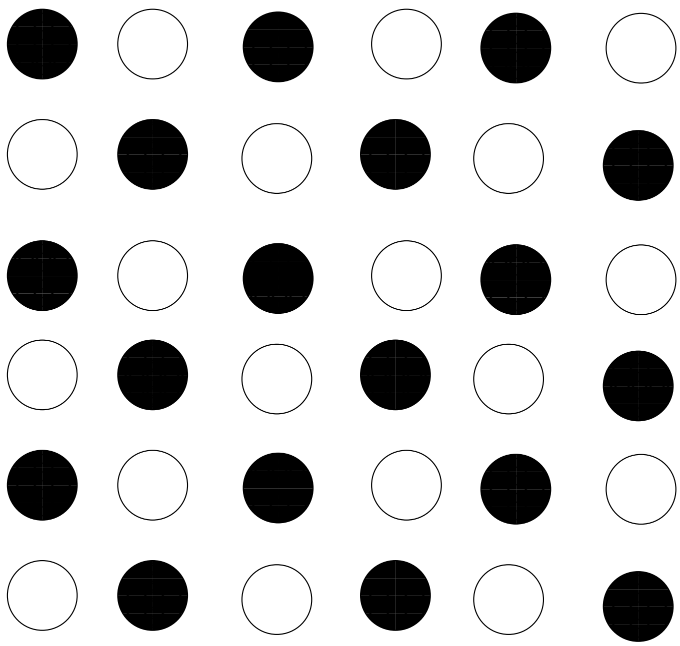
$$PT = K(r^2\omega + (4 + \log p)(\alpha + r\beta))$$

$$Speedup = \frac{\omega r^2 p^2}{r^2\omega + (4 + \log p)(\alpha + r\beta)} \approx 21.2.$$

$$Efficiency = 84\%.$$

Red-Black Ordering

Reordering variables to eliminate most of data dependence in the Gauss Seidel algorithm.



- Points are divided into "red" points (white) and black points.
- First, black points are computed (using the old red point values).
- Second, red points are computed (using the new black point values).

Parallel code for red-black ordering

- Point (i,j) is black if i+j is even.
- Point (i,j) is red if i+j is odd.
- Computation on black points (stage 1) can be done in parallel.
- Computation on red points (stage 2) can be done in parallel.

Parallel Code (Kernel)

- For all points (i,j) with (i+j) mod 2=0, do in parallel

$$u_{i,j} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}).$$

- For all points (i,j) with (i+j) mod 2=1, do in parallel

$$u_{i,j} = 0.25(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}).$$