Project Assignment 1
—
MPI-programming

Håkan Grahn and Charlie Svahnberg
Blekinge Institute of Technology

Task 1: Blocked Matrix-Matrix Multiplication

```
1.  procedure MAT_MULT (A, B, C)
2.  begin
3.      for i := 0 to n − 1 do
4.          for j := 0 to n − 1 do
5.              begin
6.                  C[i, j] := 0;
7.                  for k := 0 to n − 1 do
8.                      C[i, j] := C[i, j] + A[i, k] × B[k, j];
9.              endfor;
10. end MAT_MULT
```

Algorithm 8.2 The conventional serial algorithm for multiplication of two $n \times n$ matrices.

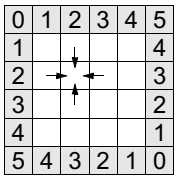
- Implement an MPI-version using a **2-dimensional data partitioning**

Task 2: LaPlace approximation

- In a matrix, calculate an average of all neighbors for each cell.

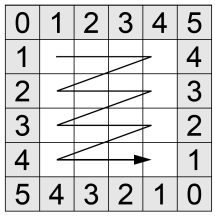
$$V[x,y] = \frac{V[x-1,y] + V[x,y-1] + V[x+1,y] + V[x,y+1]}{4}$$

- After the matrix has been traversed, new averages might need to be calculated.
- For each traversion, the largest change for any cell is compared to an acceptance value.
- When the largest change is lower than the acceptance value, a solution is found.



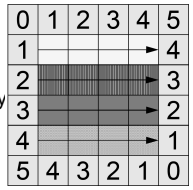
Serial (Original) solution

- Calculate new values, row-by-row
- This is also called a Gauss-Seidel iteration



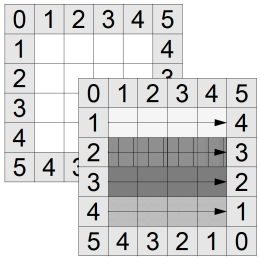
First Parallel Approach

- Calculate each row in parallel
- Why not each column?
  - Locality of reference will be very bad
  - Matrix is stored row-by-row in memory
- Keep a local maximum change (one per row)
  - Calculate global maximum change at the end of each completed iteration.
- Will not produce the same result twice!
  - Row-2 is depending on how far row-1 and row-3 has been computed. (use of old and new values are mixed).



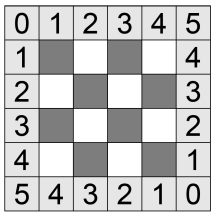
Second Parallel Approach

- Keep two matrixes;
  - one for old values
  - one for new values
- Swap role after each iteration
- Memory use has been doubled!
- A.k.a. Jacobi algorithm



Third Parallel Approach

- Make two sweeps:
  - In first sweep calculate all grey cells
  - In second sweep calculate all white cells.
- Not same algorithm any more!
- A.k.a. SOR or Red-Black solver
- Can use every second row instead in the two phases



Task: Implement an MPI-version of the SOR-solver!