Programming Techniques for Supercomputers Tutorial

Erlangen National High Performance Computing Center

Department of Computer Science

FAU Erlangen-Nürnberg

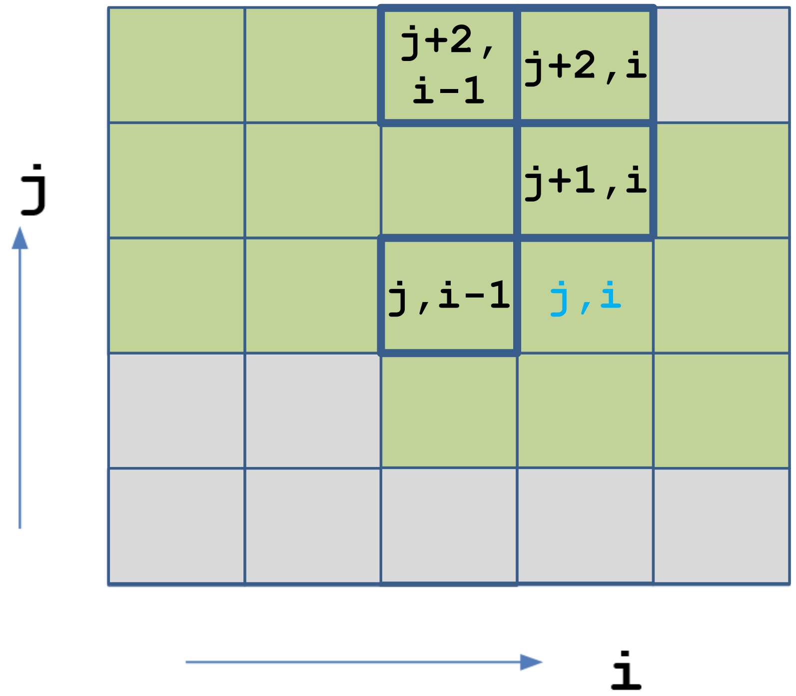
Sommersemester 2026

Assignment 10 – Task 1

Stencils a) Layer Condition

```
#pragma omp parallel for schedule(static)
for(j=0; j<N-2; ++j)
  for(i=1; i<M-1; ++i)
    y[j][i] = c * (x[j][i-1]
                 + x[j+1][i]
                 + x[j+2][i]
                 + x[j+2][i-1]);
```

Stored in a different grid (Jacobi-like)



- Layer condition: 3 rows of length M have to fit in the cache → LC:

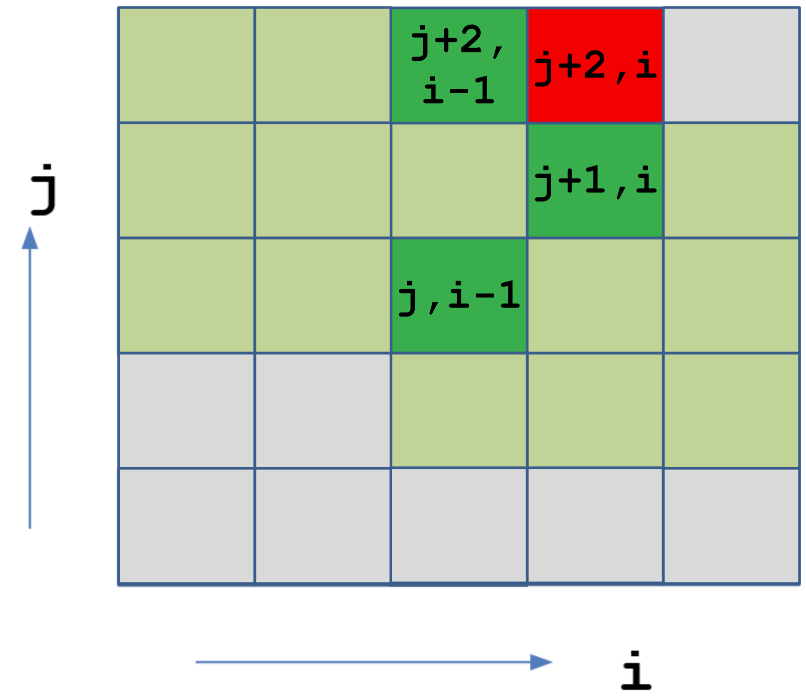
$$3 \times M \times 8\ [\mathrm{B}] < C_t/2$$

where C_t is the cache size per thread

Assignment 10 – Task 1

Stencils b) Code Balance

```
#pragma omp parallel for schedule(static)
for(j=0; j<N-2; ++j)
  for(i=1; i<M-1; ++i)
    y[j][i] = c * (x[j][i-1]
                 + x[j+2][i-1]
                 + x[j+1][i]
                 + x[j+2][i]);
```



LC: $3 \times M \times 8\ [\mathrm{B}] < C_t/2$

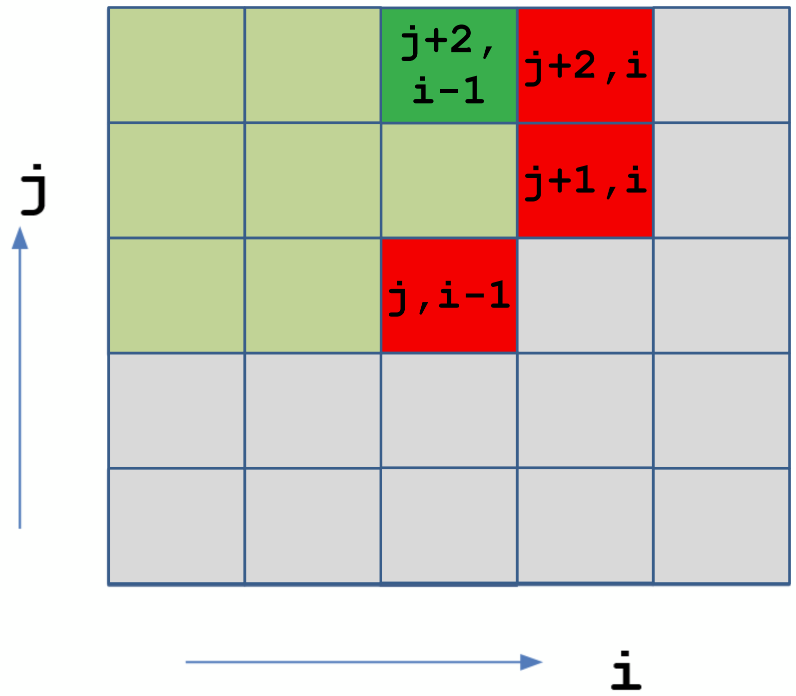
- Case 1: LC satisfied (best case)
  - 1 LD miss on **x**
  - 1 ST miss on **y**

$$\rightarrow \boldsymbol{B_C} = (8+16)[B/LUP] = \mathbf{24}\ [\boldsymbol{B/LUP}]$$

Assignment 10 – Task 1

Stencils b) Code Balance

```
#pragma omp parallel for schedule(static)
for(j=0; j<N-2; ++j)
  for(i=1; i<M-1; ++i)
    y[j][i] = c * (x[j][i-1]
                 + x[j+2][i-1]
                 + x[j+1][i]
                 + x[j+2][i]);
```



LC: $3 \times M \times 8\ [\mathrm{B}] < C_t/2$

- Case 2: LC broken (worst case)
  - 3 LD misses on **x**
  - 1 ST miss on **y**

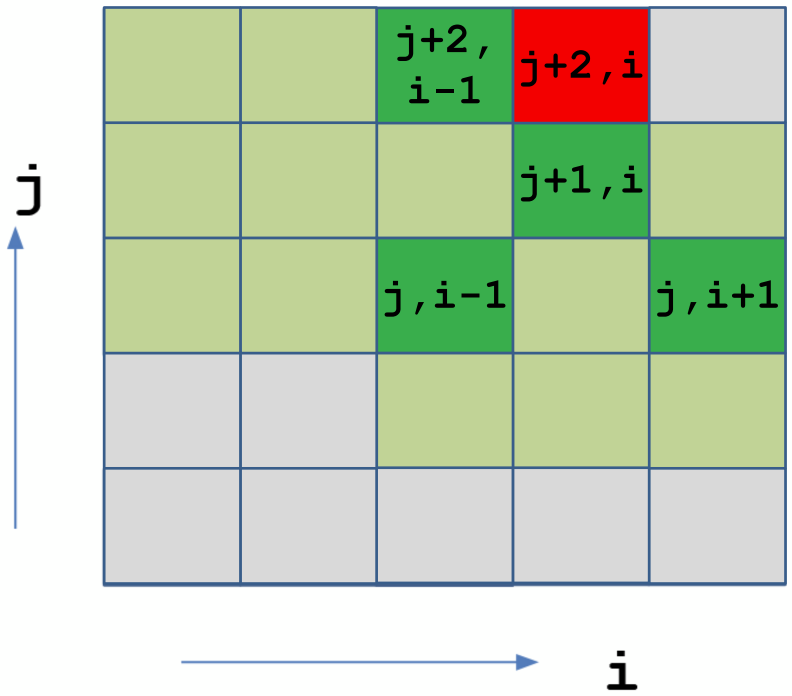
$$\rightarrow \boldsymbol{B_C} = (24 + 16)[B/LUP] = \mathbf{40}\ [\boldsymbol{B/LUP}]$$

Assignment 10 – Task 1

Stencils c) Performance Limit

- Optimistic limit when LC satisfied:

$$P = \frac{b_s}{B_C} = \frac{160\ [GB/s]}{24\ [B/LUP]} = 6.7\ \left[\frac{GLUP}{s}\right]$$



LC: $3 \times M \times 8\ [\mathrm{B}] < C_t/2$

- Single Fritz ICX core:
  - 1.5 [MiB] L3 cache
  - 1.25 [MiB] L2 cache

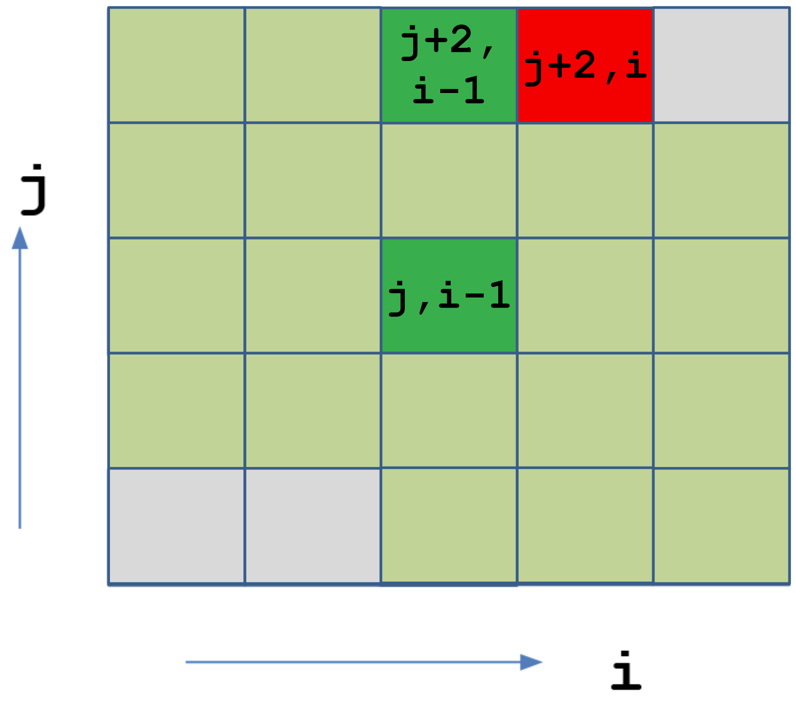
$$\rightarrow M < 2.75 \times 2^{20}[B]/48\ [B] \approx 60000$$

Inner grid dimension

Assignment 10 – Task 1

Stencils d) Modified Stencil

```
#pragma omp parallel for schedule(static)
for(j=0; j<N-2; ++j)
  for(i=1; i<M-1; ++i)
    y[j][i] = c * (x[j][i-1]
                 + x[j+2][i]
                 + x[j+2][i-1]);
```



- no “middle” row:
  each neighbor has to keep **two rows** in cache

➔ LC: $4 \times M \times 8\ [\mathrm{B}] < C_t/2$

Assignment 10 – Task 2

Claim: “Slow code scales better”

$T(1) = T_s + T_p,\;\; T(N) = T_s + \frac{T_p}{N} + T_c(N)$ (and $T_c(1) = 0$)

a) $$S(N) = \frac{T(1)}{T(N)} = \frac{T_s + T_p}{T_s + \frac{T_p}{N} + T_c(N)} = \frac{1}{s + \frac{p}{N} + c(N)}$$

with the dimensionless quantities

$$s = \frac{T_s}{T_s+T_p}, \qquad p = \frac{T_p}{T_s+T_p}, \qquad c(N) = \frac{T_c(N)}{T_s+T_p}$$

Assignment 10 – Task 2

b) Slowdown factor $\sigma < 1$: $s \rightarrow \frac{s}{\sigma}$, $p \rightarrow \frac{p}{\sigma}$, $\mathrm{c}(N)$ unchanged

$$S(N,\sigma) = \frac{1/\sigma}{s/\sigma + \frac{p}{N\sigma} + c(N)} = \frac{1}{s + \frac{p}{N} + \sigma c(N)}$$

→ communication overhead gets damped (reduced) by a factor of $\sigma < 1$

→ $S(N,\sigma < 1) > S(N,\sigma = 1)$ as long as $\mathrm{c}(N) > 0$

Prerequisite: There must be a non-negligible communication overhead for this "stunt" to work!

Assignment 10 – Task 2

c) Case $s = 0,\ c(N) = \alpha N$ unchanged

$$P(N,\sigma) = \sigma P(1) \times S(N,\sigma) = \frac{\sigma P(1)}{s + \frac{p}{N} + \sigma c(N)} = \frac{\sigma P(1)}{\frac{p}{N} + \sigma\alpha N}$$

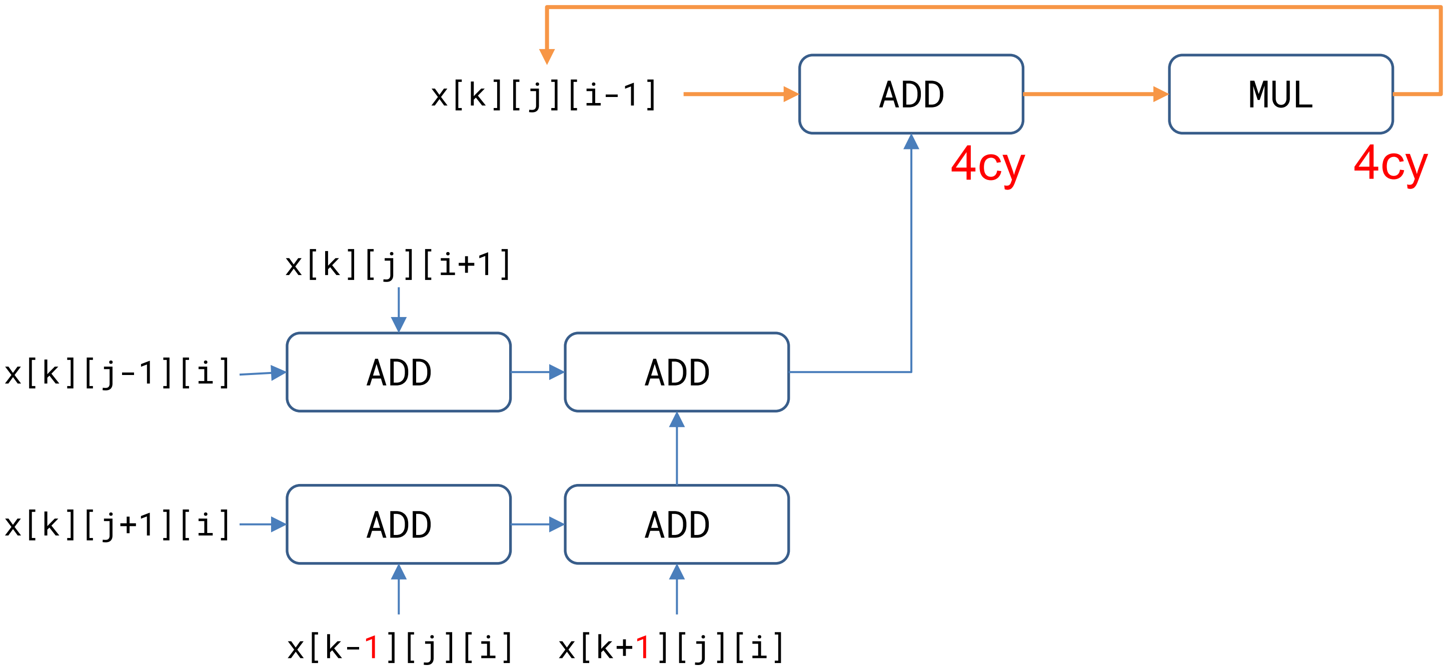
→ If $P(N \gg 1) = \frac{P(1)}{\alpha N}$

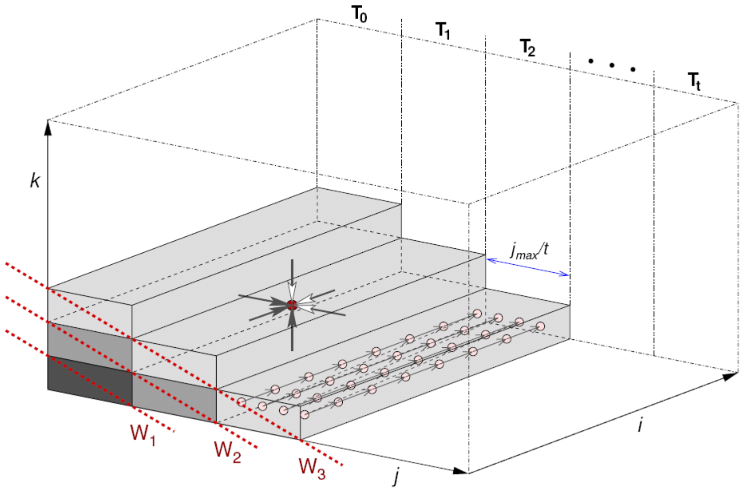
i.e., the slowdown does not matter because the sync dominates performance !

Assignment 10 – Task 3 a)

- Wavefront Gauss-Seidel

```
for(int i=1; i<imax-1; ++i) {
  x[k][j][i] = ( x[k][j][i-1] + x[k][j][i+1]
               + x[k][j-1][i] + x[k][j+1][i]
               + x[k-1][j][i] + x[k+1][j][i])
                             * one_over_six;
}
```





$$P_{max} = 1\ \mathrm{LUP}/8\ cy = 0.125\ \frac{\mathrm{LUP}}{cy}$$

Assignment 10 – Task 3 b)

- Wavefront Gauss-Seidel

```
for(int i=1; i<imax-1; ++i) {
  x[k][j][i] = ( x[k][j][i-1] + x[k][j][i+1]
               + x[k][j-1][i] + x[k][j+1][i]
               + x[k-1][j][i] + x[k+1][j][i])
                           * one_over_six;
}
```

$$P_{max} = 18 \times 0.125 \times 2.4 \times 10^9 \text{ LUP}/s \; = \; \mathbf{5.4\ GLUP}/s$$

$$b_s = \mathrm{B_C} \times P = 16 \frac{B}{LUP} \times 5.4 \text{ GLUP}/s = \mathbf{86.4}\ \boldsymbol{GB/s}$$ **> 75 GB/s**

➔ memory bandwidth will be the bottleneck!