NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Programming Techniques for Supercomputers Tutorial

Erlangen National High Performance Computing Center

Department of Computer Science

FAU Erlangen-Nürnberg

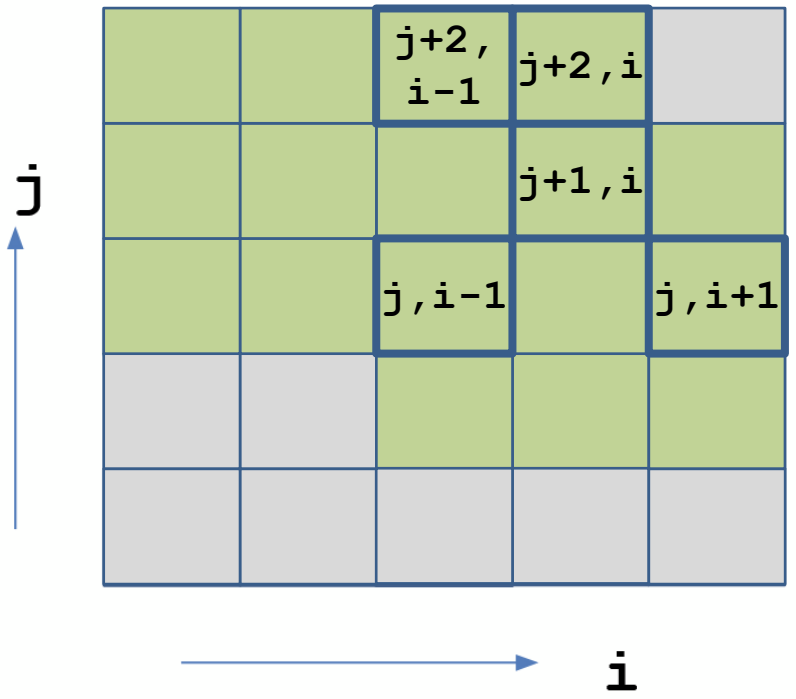
Sommersemester 2025

Assignment 9 – 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][i+1]
                 + x[j+2][i-1]
                 + x[j+1][i]
                 + x[j+2][i]) + f[j][i];
```

Stored in a different grid (Jacobi-like)



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

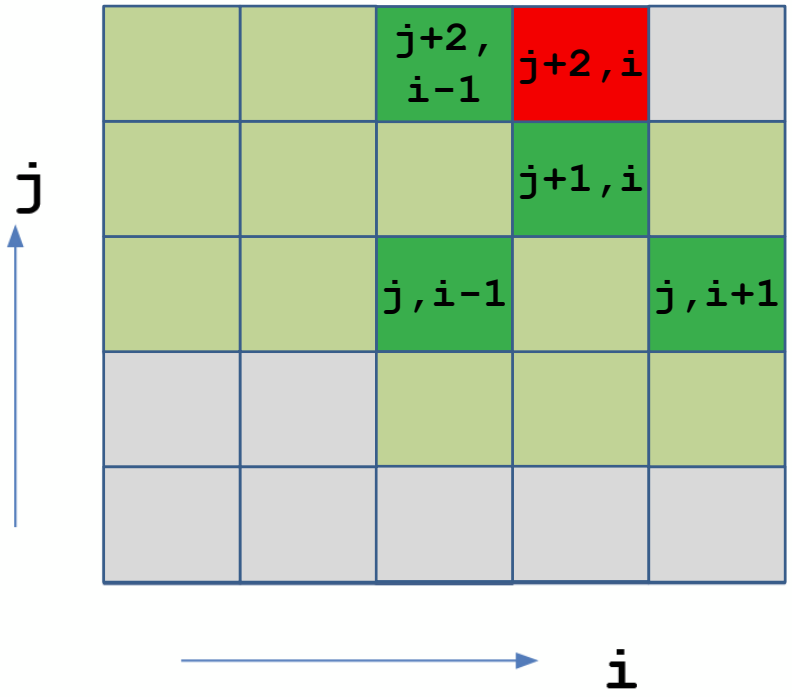
$$3 \times M \times 8\ [\mathrm{B}] < C_t/2$$

where C_t is the cache size per thread

Assignment 9 – 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][i+1]
                 + x[j+2][i-1]
                 + x[j+1][i]
                 + x[j+2][i]) + f[j][i];
```



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

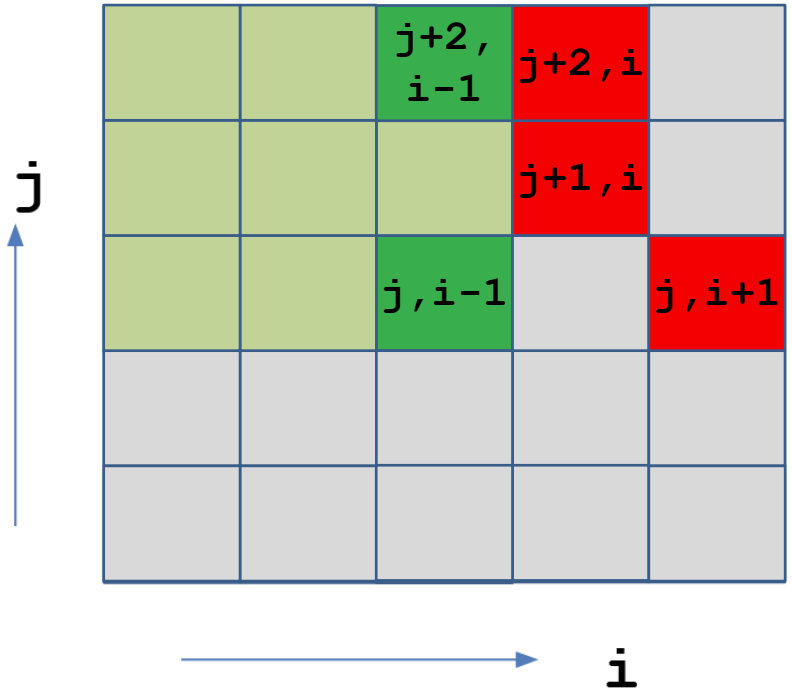
LC: $3 \times M \times 8\ [\mathrm{B}] < C_t/2$

$$\rightarrow \boldsymbol{B_C} = (8 + 8 + 16)[B/LUP] = \mathbf{32}\ [\boldsymbol{B/LUP}]$$

Assignment 9 – 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][i+1]
                 + x[j+2][i-1]
                 + x[j+1][i]
                 + x[j+2][i]) + f[j][i];
```



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

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

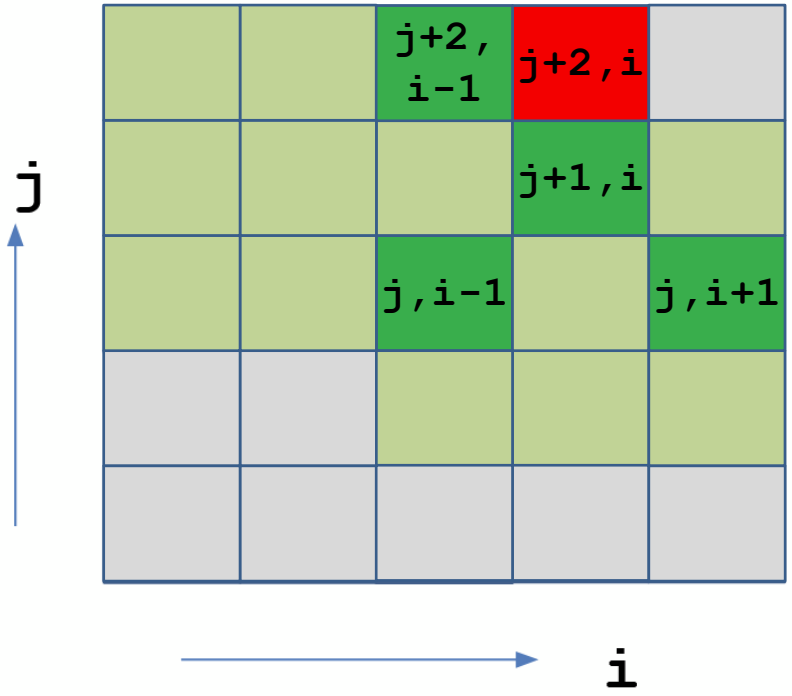
$$\rightarrow \boldsymbol{B_C} = (24 + 8 + 16)[B/LUP] = \mathbf{48}\ [\boldsymbol{B/LUP}]$$

Assignment 9 – Task 1

Stencils c) Performance Limit

- Optimistic limit when LC satisfied:

$$P = \frac{b_S}{B_C} = \frac{160\ [GB/s]}{32\ [B/LUP]} = 5\ \left[\frac{GLUP}{s}\right]$$



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

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

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

Inner grid dimension

Assignment 9 – Task 2

OpenMP Overhead

```
a[i] = a[i] + s * b[i]
```

a) Serial Performance

- With $N = 2000$, $P \approx 15.02\ [GF/s]$
- $T = \frac{4000[F]}{15.02\ [GF/s]} * 2 * 10^9[cy/s] = 533\ [cy]$

b) Parallel Performance ($nt :=$ number of threads)

- $nt = 1$: $P = 6.89\ [GF/s]$
- $nt = 2$: $P = 3.38\ [GF/s]$ with $N = 4000$
- $nt = 4$: $P = 6.52\ [GF/s]$ with $N = 8000$
- $nt = 18$: $P = 17.7\ [GF/s]$ with $N = 36000$

Increase problem size with nt

Assignment 9 – Task 2

OpenMP Overhead

```
a[i] = a[i] + s * b[i]
```

c) Overhead in cycles

- $\boldsymbol{nt = 1}$: $T = \frac{4000}{6.89} * 2\ [cy] = 1160\ [cy]$
  - **Overhead: $\boldsymbol{627}\ [\boldsymbol{cy}]$**
- $nt = 2$: $T = \frac{8000}{3.38} * 2\ [cy] = 4730\ [cy]$
  - Overhead: $4200\ [cy]$
- $nt = 4$: $T = \frac{16000}{6.52} * 2\ [cy] = 4900\ [cy]$
  - Overhead: $4370\ [cy]$
- $nt = 18$: $T = \frac{72000}{17.7} * 2\ [cy] = 8140\ [cy]$
  - Overhead: $7600\ [cy]$

Assignment 9 – Task 3

Hardware Performance Counters a) Instrumentation

```
#include <likwid-marker.h>

int main(...) {
 LIKWID_MARKER_INIT;
 #pragma omp parallel
 {
  LIKWID_MARKER_REGISTER(“triad”);
 }

 <… benchmark …>

 LIKWID_MARKER_CLOSE;
}
```

```
#pragma omp parallel
{
 LIKWID_MARKER_START("triad");
 for(int j=0; j<niter; j++){
  #pragma omp for schedule(static)
  for(i=0; i<size; ++i)
   a[i]=b[i]+c[i]*d[i];
  if(a[5]<0.0)
   cout<<a[3]<<b[5]<<c[1]<<d[9];
 }
 LIKWID_MARKER_STOP("triad");
}
```

Assignment 9 – Task 3

Hardware Performance Counters b) Data Volume

- Single core

```
srun –cpu-freq=performance likwid-perfctr -g MEM -C S0:0 -m ./a.out
```

  - Expect a read to write data volume ratio of **4 : 1**

- Single NUMA domain (18 cores)

```
srun –cpu-freq=performance likwid-perfctr -g MEM -C S0:0-17 -m ./a.out
```

  - Expect a read to write data volume ratio of **3 : 1**

- Due to Fritz ICL automatic write-allocate evasion mechanism

Assignment 9 – Task 3

Hardware Performance Counters c) Cache Traffic

- L2 Cache

```
srun –cpu-freq=performance likwid-perfctr -g L2 -C S0:0-17 -m ./a.out
```

  - Expect a read to write data volume ratio of **4 : 1**
  - Cache line claim does not help us here

- L3 Cache

```
srun –cpu-freq=performance likwid-perfctr -g L3 -C S0:0-17 -m ./a.out
```

  - Expect a read to write data volume ratio of **1 : 1**

- Due to Fritz ICL exclusive victim L3 cache
  - E.g. Every array is written back into L3, even if they aren’t modified