NHR 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 8 – Task 1

Resource-driven modeling

$$\boldsymbol{T_{\text{exec}} = \frac{W}{p_{\text{E}}}} \qquad \boldsymbol{T_{\text{data}} = \frac{V \times f}{b_S}}$$

a) No overlap

$$T_{\text{nOL}} = T_{\text{exec}} + T_{\text{data}} = \frac{W}{p_{\text{E}}} + \frac{V \times f}{b_S}$$

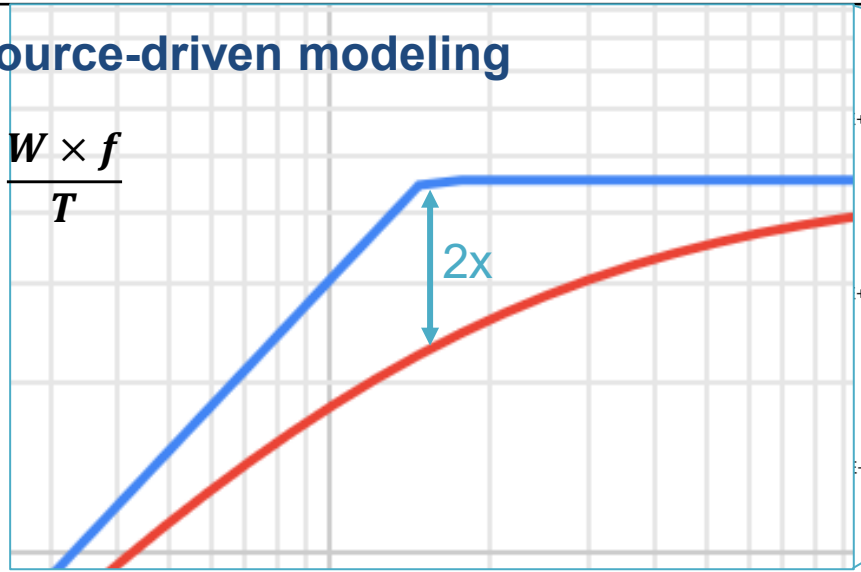
b) Full overlap

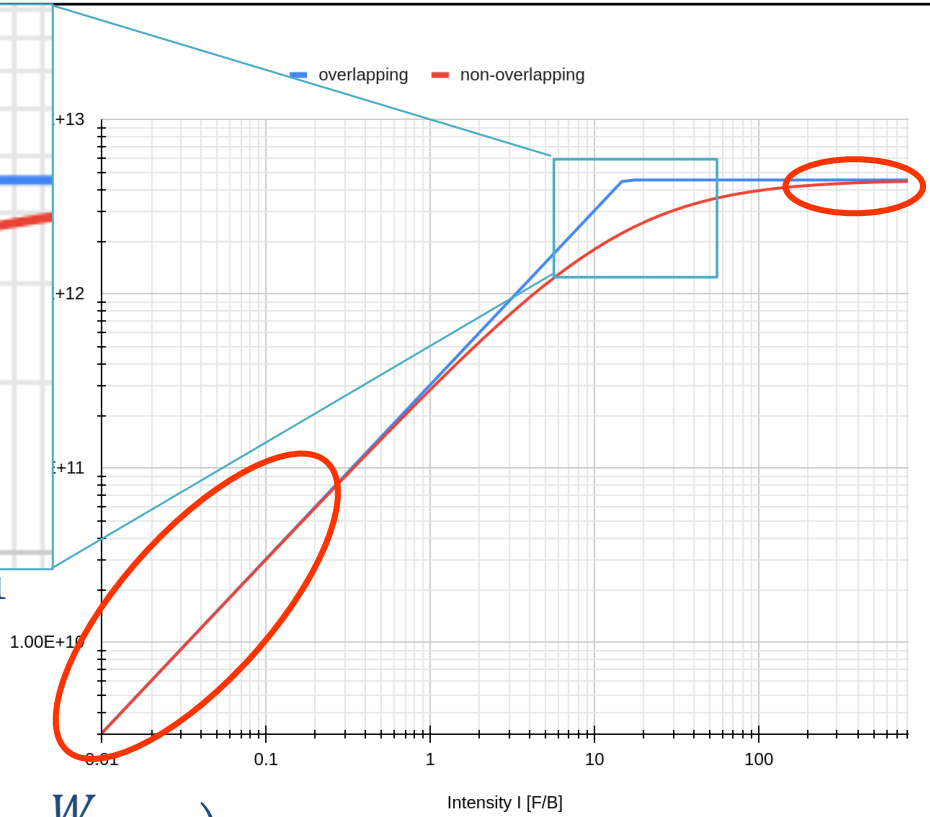
$$T_{\text{OL}} = \max(T_{\text{exec}}, T_{\text{data}}) = \max\left(\frac{W}{p_{\text{E}}}, \frac{V \times f}{b_S}\right)$$

Assignment 8 – Task 1a)

Resource-driven modeling

$$P = \frac{W \times f}{T}$$





$$\rightarrow P_{\mathrm{nOL}} = \frac{W \times f}{\frac{W}{p_{\mathrm{E}}} + \frac{V \times f}{b_S}} = \left(\frac{1}{p_{\mathrm{E}} \times f} + \frac{V}{W \times b_S}\right)^{-1}$$

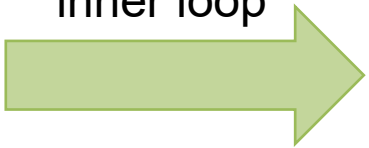
$$\rightarrow P_{\mathrm{OL}} = \frac{W \times f}{\max(T_{\mathrm{exec}} + T_{\mathrm{data}})} = \min\left(p_E \times f, \quad \frac{W}{V} \times b_S\right)$$

Assignment 8 – Task 2 a)

Optimizing loop code

```
int N = 16384;
double mat[N][N], s[N][N];
// ...
srand(1); // random seed

for(i=0; i<N; ++i) {
    val = (double)rand();
  for(j=0; j<N; ++j)
    mat[j][i] = s[j][i]/val;
  }
}
```

move DIV outside of inner loop



```
int N = 16384;
double mat[N][N], s[N][N];
// ...
srand(1); // random seed

for(i=0; i<N; ++i) {
    val = 1./(double)rand();
  for(j=0; j<N; ++j)
    mat[j][i] = s[j][i]*val;
  }
}
```

Problems:

- wrong loop order → strided access
- outer dimension
  power of 2 w/ strided access
- divide in inner loop

Assignment 8 – Task 2 a)

Optimizing loop code

Interchange loops

```
int N = 16384;
double mat[N][N], s[N][N];
double table[N];
// ...
srand(1); // random seed
double table[N];
for(i=0; i<N; ++i)
  table[i] = 1./(double)rand();


for(j=0; j<N; ++j)
  for(i=0; i<N; ++i)
    mat[j][i] = s[j][i]*table[i];
```

Compiler might move the DIV but does not apply loop interchange

Assignment 8 – Task 2 b)

Optimizing loop code

```
for(j=0; j<N; ++j)
  for(i=0; i<N; ++i)
    mat[j][i] = s[j][i]*table[i];
```

Assuming 2.0 GHz

$\boldsymbol{P_{\max}} = 1\frac{AVX512-it}{cy}$ per core ➔ $\boldsymbol{576\ GFLOP/s}$

$\boldsymbol{B_C} = 24\frac{Byte}{Flop}$ with $b_S = 160\frac{GB}{s}$

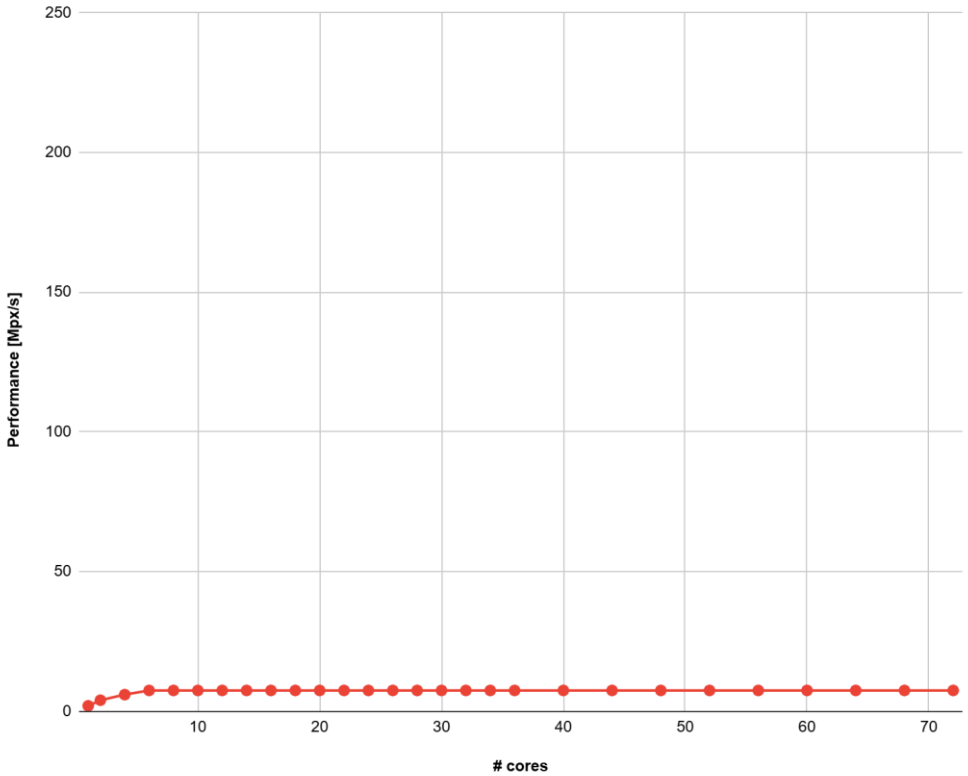
$$\boldsymbol{P} = \min\left(P_{\max}, \frac{b_S}{B_C}\right) = \min\left(576, \frac{160}{24}\right) GF/s = 6.67\ GF/s$$

memory-bound!

Assignment 7 – Task 3

```
// ...
#pragma omp parallel private(tile)
{
tile=(char*)malloc(tilesize*tilesize*sizeof(char));

//..

count = 0;
#pragma omp for private(xc,i) reduction(+:count)
for(yc=0; yc<ytiles; yc++) {
  for(xc=0; xc<xtiles; xc++) {
     /* calc one tile */
     calc_tile(size, xc*tilesize, yc*tilesize, tilesize, tile);
     /* copy to picture buffer */
     for(i=0; i<tilesize; i++) {
         int tilebase = yc*tilesize*tilesize*xtiles+xc*tilesize;
         memcpy((void*)(picture+tilebase+i*tilesize*xtiles),
                (void*)(tile+i*tilesize),
                tilesize*sizeof(char));
     }
     count++;
   }
}
}
```

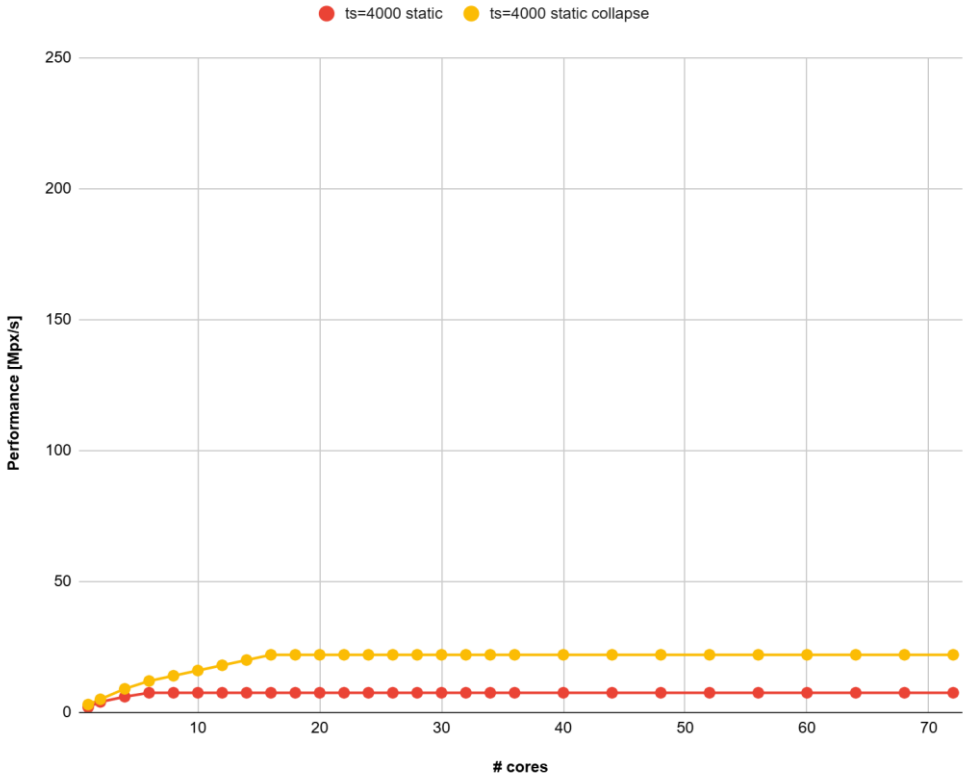
Assignment 8 – Task 3



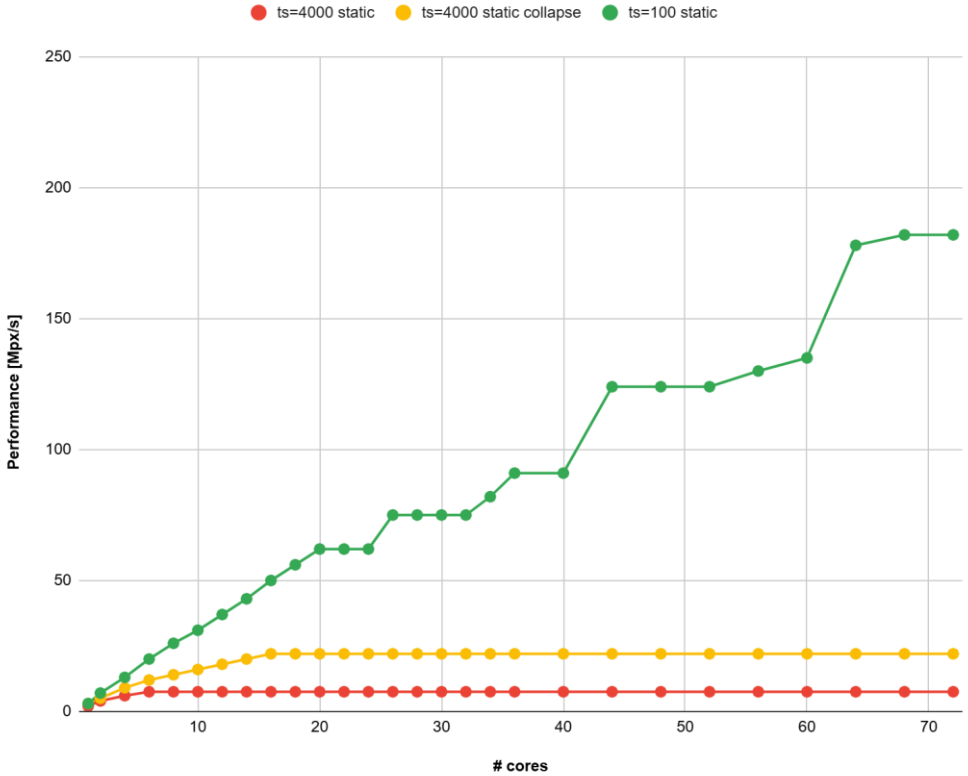
Assignment 8 – Task 3

```
// ...
#pragma omp parallel private(tile)
{
tile=(char*)malloc(tilesize*tilesize*sizeof(char));

//..

count = 0;
#pragma omp for collapse(2) private(xc,i) reduction(+:count)
for(yc=0; yc<ytiles; yc++) {
  for(xc=0; xc<xtiles; xc++) {
     /* calc one tile */
     calc_tile(size, xc*tilesize, yc*tilesize, tilesize, tile);
     /* copy to picture buffer */
     for(i=0; i<tilesize; i++) {
         int tilebase = yc*tilesize*tilesize*xtiles+xc*tilesize;
         memcpy((void*)(picture+tilebase+i*tilesize*xtiles),
                (void*)(tile+i*tilesize),
                tilesize*sizeof(char));
     }
     count++;
   }
}
}
```

Assignment 8 – Task 3



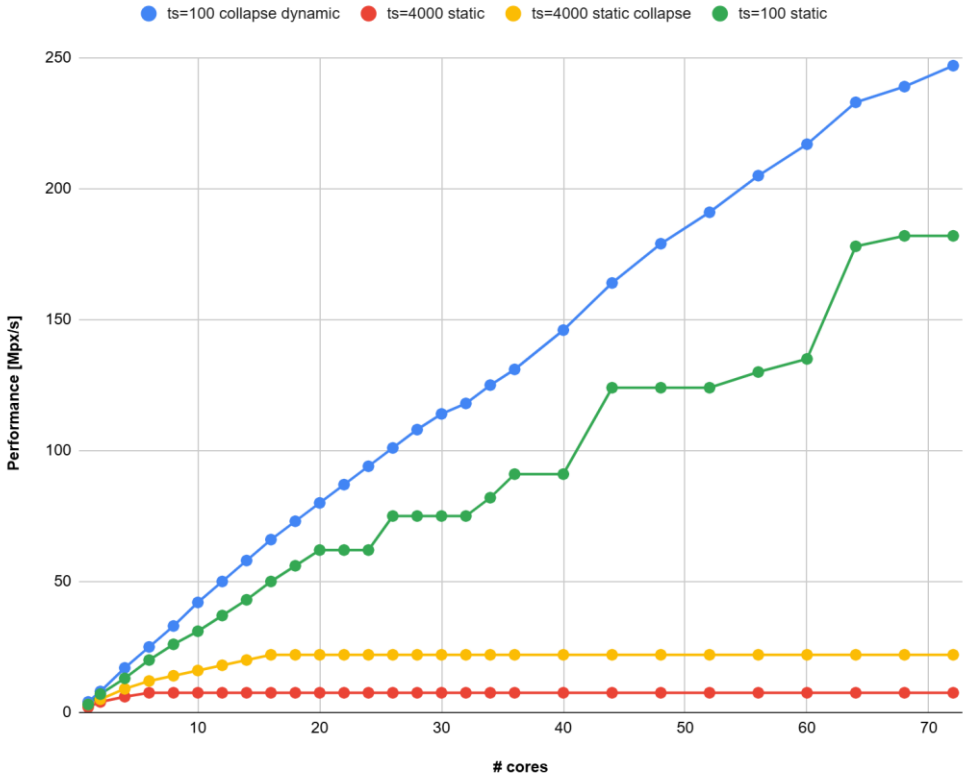
Assignment 8 – Task 3



Assignment 8 – Task 3

```
// ...
#pragma omp parallel private(tile)
{
tile=(char*)malloc(tilesize*tilesize*sizeof(char));

//..

count = 0;
#pragma omp for schedule(dynamic) collapse(2) private(xc,i) reduction(+:count)
for(yc=0; yc<ytiles; yc++) {
  for(xc=0; xc<xtiles; xc++) {
     /* calc one tile */
     calc_tile(size, xc*tilesize, yc*tilesize, tilesize, tile);
     /* copy to picture buffer */
     for(i=0; i<tilesize; i++) {
         int tilebase = yc*tilesize*tilesize*xtiles+xc*tilesize;
         memcpy((void*)(picture+tilebase+i*tilesize*xtiles),
                (void*)(tile+i*tilesize),
                tilesize*sizeof(char));
     }
     count++;
   }
}
}
```

Assignment 8 – Task 3



Assignment 8 – Task 3c)

Data transfer per pixel: 1B

With 123 Mpx/s (on 36 cores) → 123 MB/s << 160 GB/s