NHR FAU

Friedrich-Alexander-Universität Erlangen-Nürnberg

Programming Techniques for Supercomputers Tutorial

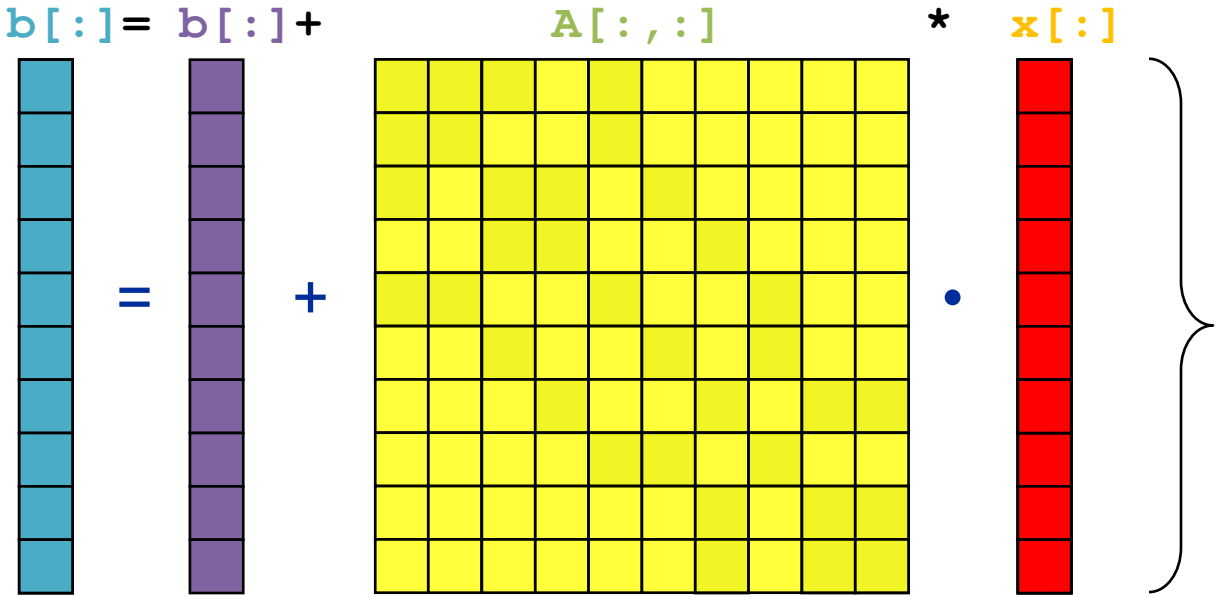
Erlangen National High Performance Computing Center

Department of Computer Science

FAU Erlangen-Nürnberg

Sommersemester 2025

Assignment 11 – Task 1a)

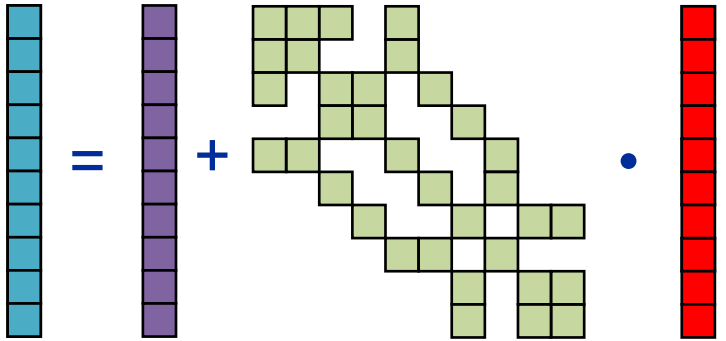


General case: some indirect addressing required!

```
for i = 0:nrows-1 //Long outer loop
    for j = row_ptr[i]:row_ptr[i+1]-1 // Short inner loop
        b[i] = b[i] + A[j] * x[col_idx[j]]
```

Assignment 11 – Task 1a)

```
for i = 0:nrows-1 //Long loop
  for j = row_ptr[i]:row_ptr[i+1]-1 //Short loop
      b[i] = b[i] + A[j] * x[col_idx[j]]
```



Min LOADs: $(4\,\mathrm{B} + 4\,\mathrm{B})\,N_{nz} + (4\,\mathrm{B} + 4\,\mathrm{B})\,N_r + 4\,\mathrm{B}\,N_c$

Min STOREs: $4\,\mathrm{B}\,N_r$

- N_{NZ}: 225×10^6
- N_r: 26.5×10^6
- data types: `float` / `uint32`

$$B_{C,min} = \frac{8\,N_{nz} + 12\,N_r + 4\,N_c}{2\,N_{nz}} \frac{B}{FLOP} = \frac{8 + 12/N_{nzr} + 4/N_{nzc}}{2} \frac{B}{FLOP} = 4.94 \frac{B}{FLOP}$$

assuming square matrix

$$P_{max} = \frac{b_s}{B_{C,min}} = \frac{1.3}{4.94} \frac{TFLOP}{s} = \mathbf{263} \frac{\boldsymbol{GFLOP}}{\boldsymbol{s}} \qquad T_{spmv} = \frac{V_{min}}{b_s} = \frac{8\,N_{nz} + 16\,N_r}{1.3 \times 10^{12}} s = \mathbf{1.71\,ms}$$

Assignment 11 – Task 1b)

Measurement: $\mathrm{P} = 190 \frac{GFLOP}{s}$ $\mathrm{V} = 2.3\ GB$

$$B_C^{act} = \frac{V}{2 \times N_{nz}\ FLOP} = 5.11\ \frac{B}{FLOP}$$

Assignment 11 – Task 1c)

```
for i = 0:nrows-1 //Long loop
  for j = row_ptr[i]:row_ptr[i+1]-1 //Short loop
      b[i] = b[i] + A[j] * x[col_idx[j]]
```

$$B_{C,min} = \frac{8\,N_{nz} + 12\,N_r + 4\,N_r}{2\,N_{nz}} \frac{B}{FLOP}$$

B_C is only 3% larger than optimum ➔ bad mem utilization
One option: change matrix storage format, e.g., to SELL-C-sigma

$$B_C(\alpha) = \frac{8\,N_{nz} + 12\,N_r + 4\alpha N_{nzr} N_r}{2\,N_{nz}} \frac{B}{FLOP} = \left(4 + \frac{6}{N_{nzr}} + 2\alpha\right) \frac{B}{FLOP}$$

$$B_C^{act} = 5.11\ \frac{B}{FLOP}$$

$$b_s = b_s^{meas} \times B_C^{act}\ GB/s = \mathbf{971\ GB/s}$$

≈ 75% of peak BW

$$\alpha = \left(5.11 - 4 - \frac{6}{N_{nzr}}\right) \div 2 = \mathbf{0.202} \text{ and } \alpha N_{nzr} = 1.72$$

Assignment 11 – 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

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

Assignment 11 – Task 2

b) Slowdown factor $\mu < 1$: $s \to \frac{s}{\mu}$, $p \to \frac{p}{\mu}$, $c(N)$ unchanged

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

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

→ $S(N, \mu < 1) > S(N, \mu = 1)$ as long as $c(N) > 0$

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

Assignment 11 – Task 3a)

axpby():

```
#pragma omp parallel for
for(int i=0; i<size; i++) {
  R[i] = (a * X[i]) + (b * Y[i]);
}
```

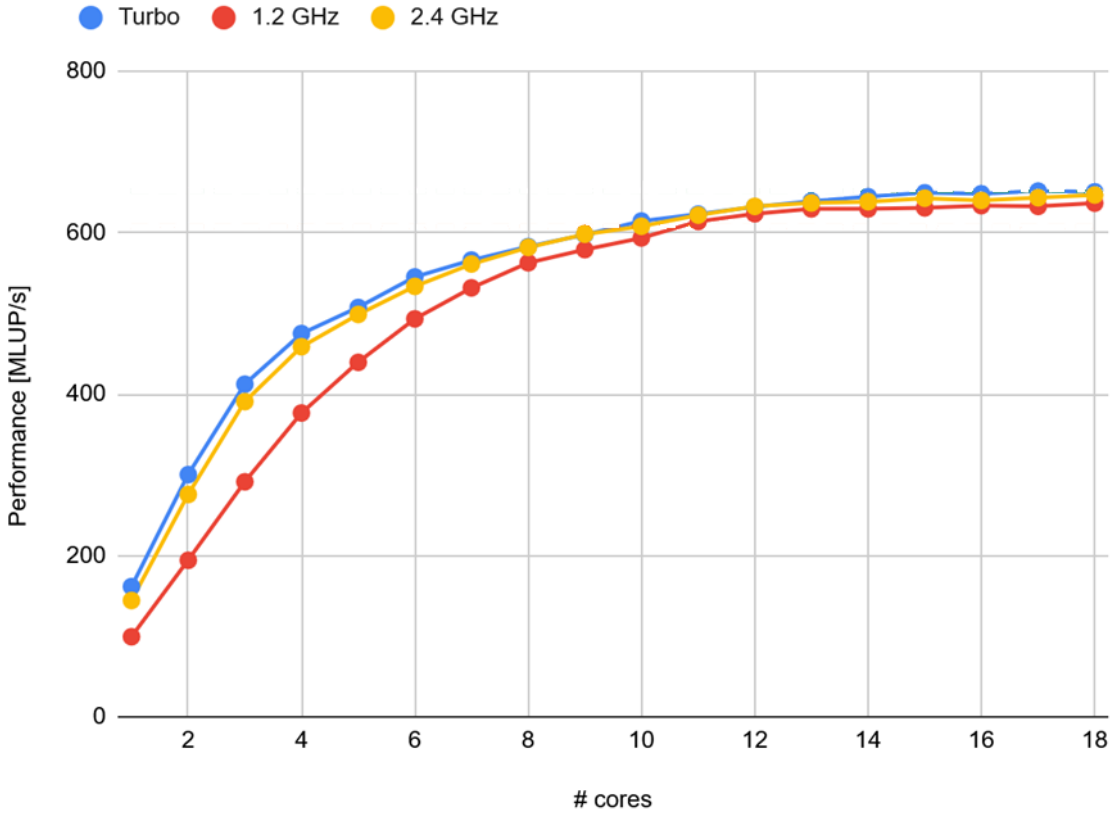
ddot():

```
#pragma omp parallel for \
                reduction(+:l2_sq)
for(int i=0; i<size; i++) {
  l2_sq += X[i] * Y[i];
}
```

applyStencil():

```
#pragma omp parallel for
for(int j=1; j<size_y-1; j++) {
  for(int i=1; i<size_x-1; i++) {
    R[j*size_x + i] = w_c * U[j*size_x + i] –
                w_y * (U[(j+1)*size_x + i] + U[(j-1)*size_x + i]) -
                w_x * (U[j*size_x + i+1] + U[j*size_x + i-1]);
        }
}
```

Assignment 11 – Task 3b)



Assignment 11 – Task 3c)

- axpby: $3 \times 24 \frac{B}{LUP}$
- ddot: $24 \frac{B}{LUP}$
- applyStencil: $16 \frac{B}{LUP}$

LC: $C > 2 \times 480\ kB$
L2 size is 1.25 MB

```
double CG(..) {
  // ...
  S = getTimeStamp();
  while( (iter < solver->niter) && (alpha_0 > tolSquared) ) {
    T(APPLY_STENCIL) applyStencil(&v, &p, solver);
    T(DOT_PRODUCT) dotProduct(&v, &p, &lambda);
    lambda = alpha_0/lambda;
    //Update x
    T(AXPBY) axpby(x, 1.0, x, lambda, &p);
    //Update r
    T(AXPBY) axpby(&r, 1.0, &r, -lambda, &v);
    T(DOT_PRODUCT) dotProduct(&r, &r, &alpha_1);
    //Update p
    T(AXPBY) axpby(&p, 1.0, &r, alpha_1/alpha_0, &p);
    alpha_0 = alpha_1;
    ++iter;
  }
  // ...
}
```

$$B_C = 3 \times 24 + 24 + 16 \frac{B}{LUP} = 112 \frac{B}{LUP} \rightarrow P_{max} = \mathbf{652\ MLUP/s} \text{ for } b_s = 73\ GB/s$$

Assignment 11 – Task 3c)