Zur Erinnerung:

$$Ax = b$$

Störung nur der rechten Seite: $b \to b + \delta b$

$$\Rightarrow x \to x + \delta x \text{ mit } \delta x = A^{-1}\delta b$$

$$\|\delta x\| = \|A^{-1}\delta b\| \leq \underbrace{\|A^{-1}\|}_{\kappa_{abs}} \cdot \|\delta b\|$$

$$\frac{\|\delta x\|}{\|x\|} \leq \|A^{-1}\| \cdot \frac{\|\delta b\|}{\|x\|} = \|A^{-1}\| \cdot \frac{\|\delta b\|}{\|b\|} \cdot \frac{\|b\|}{\|x\|}$$

$$\Rightarrow \kappa_{rel} = \|A^{-1}\| \cdot \frac{\|b\|}{\|x\|} = \|A^{-1}\| \cdot \frac{\|Ax\|}{\|x\|}$$

$$\leq \|A^{-1}\| \cdot \frac{\|A\| \cancel{\|x\|}}{\cancel{\|x\|}}$$

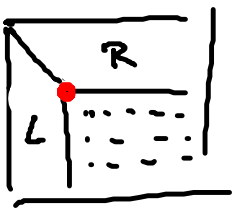
$$= \|A^{-1}\| \cdot \|A\| = \kappa(A)$$

- Die Abschätzung ist scharf
- aber nur für ganz spezielle Wahl von $b$ und $\delta b$

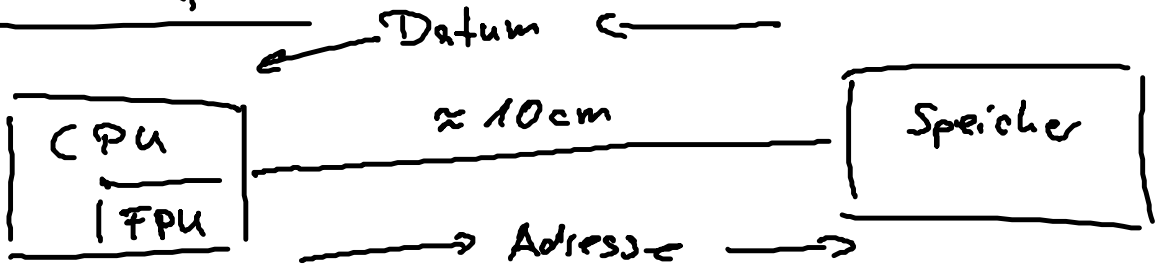
## Performance von LR-Zerlegungen

Aufwandsmaß für Algorithmen: FLOPS
(floating point operations)

Performance-Maß daher: FLOPS (floating point operations per second)



## Rechneraufbau




3 GHz

1 FLOP / Takt

⇒ 0.3 ns / FLOP

Signallaufzeit

$$= \frac{\text{Weg}}{\text{Geschwindigkeit}}$$

$$= \frac{0.1\,m}{\frac{1}{3}c}$$

$$= 10^{-9}\,s = 1\,ns$$

Zeit für round-trip: 2 ns

→ eine Zahl aus dem Speicher zu holen dauert 6x so lange, wie sie zu bearbeiten!

Weitere Effekte:
- Signalverarbeitung in CPU und Speicher
- Kapazitäten / Induktivitäten der Leitungen verringern die effektive Signalgeschw.

⇒ Die meiste Zeit wartet die CPU auf Daten

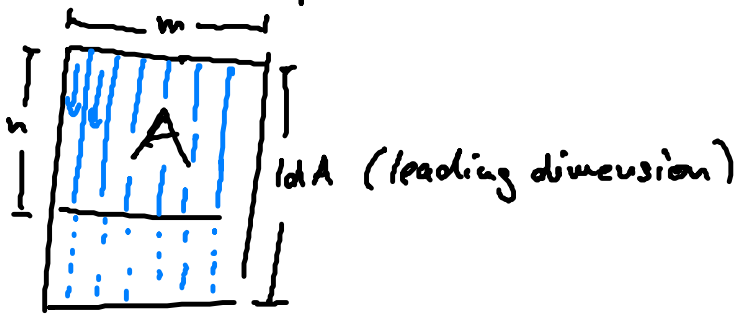
## Technische Maßnahmen

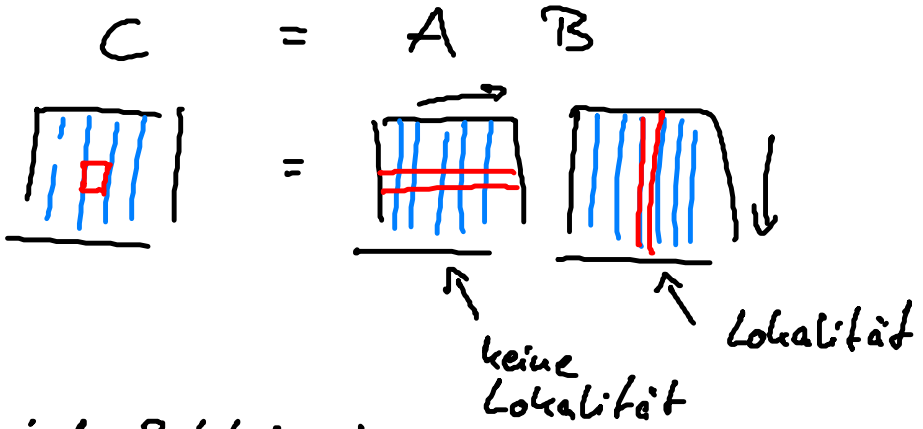
- Verstecken der Latenz: Speicher sendet ganze Blöcke von Zahlen
- Vermeiden der Latenz: Caches (schnelle kleine Zwischenspeicher auf der CPU)

Beides ist nur effektiv bei Lokalität der Datenzugriffe!

**Bsp**: Matrix - Matrix - Multiplikation

Fortran-Speicherschema (column-major)




jede Zahl in A wird n-mal (mangels Lokalität) aus dem Speicher geholt.

**Blockalgorithmen**: Umsortierung der Operationen zur Erhöhung der Lokalität

$$\begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix}$$

$$C_{00} = \underbrace{A_{00}B_{00}}_{n^3} + \underbrace{A_{01}B_{10}}_{n^2 \quad n^3}$$

Multiplikation $A_{00}B_{00}$ schnell, wenn $A_{00}$, $B_{00}$ in den Cache passen

Strategie:
- Zerlege $A\,B\,C$ in Blöcke $A_{ik}\,B_{kj}\,C_{ij}$
- kopiere $A_{ik}\,B_{kj}$ in zusammenhängende Speicherbereiche Aufwand $n^2$
- Berechne $C_{ij} = C_{ij} + A_{ik} B_{kj}$ Aufwand $n^3$

Effizienzgewinn, falls $2n^2$ Speicher + $n^3$ Cache kleiner als $n^3$ Speicher

$\longrightarrow$ $n$ möglichst groß, aber so, daß alles in den Cache paßt.

typische Blockgrößen $n = 16, \ldots, 64$

## LR-Zerlegung

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} = \begin{bmatrix} L_{00} & 0 \\ L_{10} & L_{11} \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} \\ 0 & R_{11} \end{bmatrix}$$

$$\Rightarrow \quad \begin{aligned} A_{00} &= L_{00} R_{00} \\ A_{01} &= L_{00} R_{01} \\ A_{10} &= L_{10} R_{00} \\ A_{11} &= L_{10} R_{01} + L_{11} R_{11} \end{aligned}$$

(i) LR-Zerlegung von $A_{00}$ → $L_{00}, R_{00}$ (rekursiv)

(ii) $R_{01} = L_{00}^{-1} A_{01}$ (Lösung eines Dreieckssyst. mit mehreren rechten Seiten)

(iii) $L_{10} = A_{10} R_{00}^{-1}$ "

Matrix-Matrix-Operationen

(iv) $\tilde{A}_{11} = A_{11} - L_{10} R_{01}$ ← effizientes Matrix-Matrix-Produkt verwenden

(v) LR-Zerlegung von $\tilde{A}_{11}$ → $L_{11}, R_{11}$ (rekursiv)

## Lösung von Dreieckssystemen

$LX = B$

$$\begin{bmatrix} L_{00} & \\ L_{10} & L_{11} \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} B_0 \\ B_1 \end{bmatrix}$$

$$\Rightarrow \quad L_{00} X_0 = B_0$$
$$L_{10} X_0 + L_{11} X_1 = B_1$$

(i) Löse $L_{00} X_0 = B_0$ (rekursiv)

(ii) $\tilde{B}_1 = B_1 - L_{10} X_0$

(iii) Löse $L_{11} X_1 = \tilde{B}_1$