2 – Rechnerarchitekturen

Flynnsche Klassifikation (Flynn`sche Taxonomie)

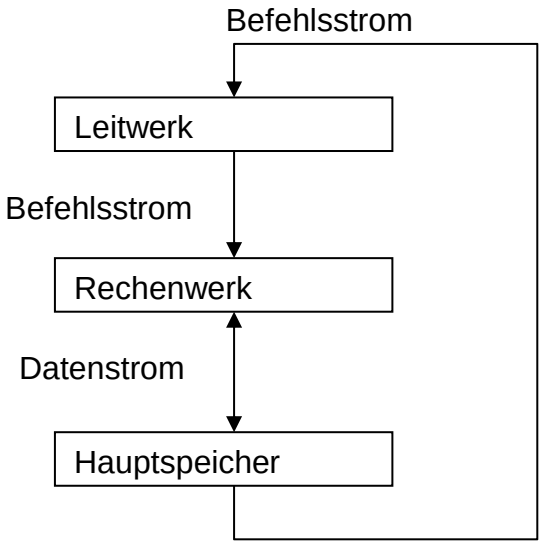
- 1966 entwickelt, einfaches Modell, bis heute genutzt
- Beschränkung der Beschreibung auf die Befehls- und Datenströme

		Instruction Streams	
		Single	**Multiple**
Data Streams	**Single**	**SISD**	**MISD**
	Multiple	**SIMD**	**MIMD**

- Befehlsstrom
  - > Folge von Maschinenbefehlen, werden vom Leitwerk als entschlüsselte Steuerinformation an das Rechenwerk weitergeleitet
- Datenstrom
  - > Daten, die vom Rechenwerk bearbeitet werden, werden zwischen dem Speicher und den Prozessorregistern übertragen

SISD – Single Instruction Stream, Single Data Stream

- Entspricht dem von-Neumann-Rechner
- Interner Aufbau unwichtig



SIMD – Single Instruction Stream, Multiple Data Streams

- Rechnersystem besitzt ein Leitwerk
- Ein Befehl kann gleichzeitig auf Operanden in mehreren Rechenwerken angewandt werden
- Erweiterung des von-Neumann-Rechners
- Beispiel: GPUs (eigentlich SIMT), Vektorprozessor

MISD – Multiple Instruction Streams, Single Data Stream

- Mehrere Leitwerke steuern die Bearbeitung eines Datenstroms
  - > Beispiel: fehlertolerante Systeme, die redundante Berechnungen ausführen
- Kaum sinnvolle Anwendung in der Praxis
  - > MIMD Systeme sind flexibler und können zur Not wie ein MISD System benutzt werden

MIMD – Multiple Instruction Streams, Multiple Data Streams

- Mehrere Befehls- und Datenströme
- Häufig mehrere Prozessoren, die zu einem Rechner zusammengefasst sind
  - > Multiprozessorsystem
- Aufbau
  - > Meist Zusammenschluss mehrerer SISD Rechner
  - > Große Rechenleistung, da problemunabhängig

2 – Rechnerarchitekturen

Parallele Rechnerarchitekturen

Wie geht es weiter?
Grundsatzfrage

Kann die Ausführungsgeschwindigkeit eines Programms durch Erhöhung von Taktrate / FLOPS einer CPU immer weiter gesteigert werden?

Es war einmal...
Expertenmeinung

Prognose für die Entwicklung von CPUs aus dem Jahre 1998

Einführungsjahr	1999	2002	2005	2008	2011	2016
Takt (GHz)	1,25	2,1	3,5	6	10	16,9
Chipgröße (mm^2)	340	430	520	620	750	900
Anzahl Pins	1867	2553	3492	4776	6532	8935
Transistoren/Chip	21M	76M	200M	520M	1,4G	3,62G

Schnellere CPUs?
Steigerung der Taktrate

Die Steigerung der Taktrate einer CPU führt zu massiver Zunahme der Abwärme

- Darstellung von 0en und 1en in der CPU durch Spannungen
- Vereinfachte Darstellung:
  - > 0: -1 Volt
  - > 1: 1 Volt
- Damit bei immer schnellerer Schaltung der Transistoren das Signal nicht so sehr verwischt, dass 0en und 1en nicht mehr unterscheidbar sind, müssen die Peaks der Spannung erhöht werden

Elektronenmigration

- Elektronen „wandern" im Leiter durch das stärkere elektrische Feld

Quelle: http://www.tomshardware.com/reviews/mother-cpu-charts-2005,1175-2.html

Schnellere CPUs?
Steigerung der FLOPS

Damit eine einzelne CPU 1 TFLOP an Leistung erzielen kann, muss im „schlimmsten" Fall ein Terabyte an Daten pro Sekunde in die CPU geladen werden.

- Die Daten müssen eine Entfernung r vom Speicher zur CPU zurücklegen
- Ein Terabyte $\triangleq 10^{12}$ Bytes
- Bestmögliche Transferrate ist Lichtgeschwindigkeit, also c
- Demnach muss $r < \frac{c}{10^{12}} = 0{,}3$ mm

Man müsste ein Terabyte Daten in ein Quadrat von 0,3 mm Seitenlänge speichern.

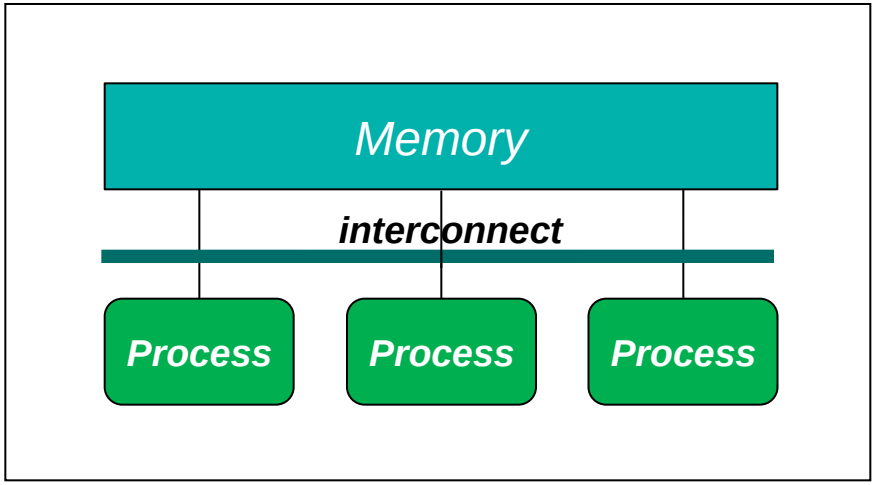
Parallele Rechnerarchitekturen
Motivation

Durch Steigerung der Taktrate / FLOPS eines einzelnen Prozessors ist keine nennenswerte Leistungssteigerung mehr zu erreichen

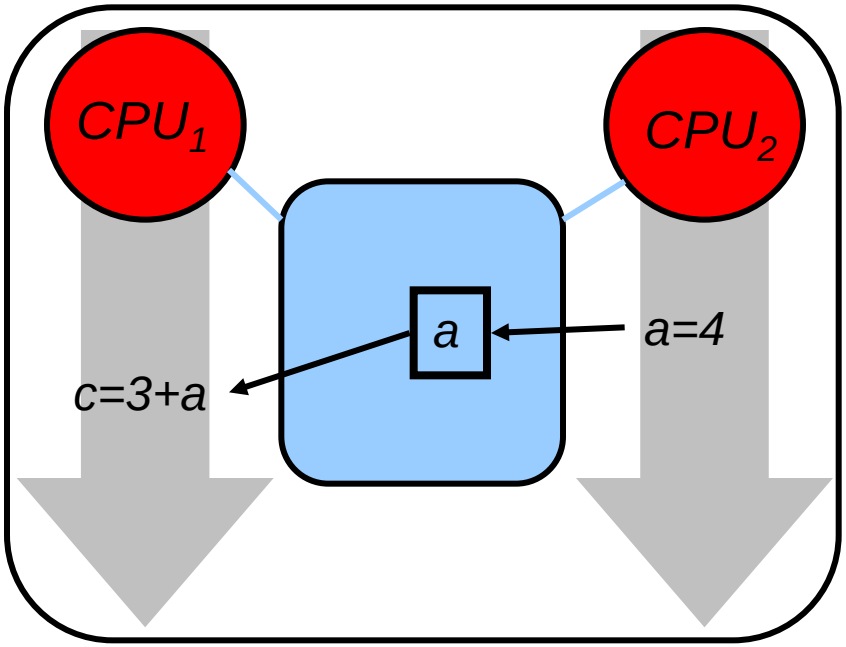
Aber:
Im Programm vorhandene, unabhängige Rechenpakete können zeitgleich verarbeitet werden

Parallele Rechnerarchitekturen
Shared-Memory Systeme

- **Implizite Datenverteilung**
- **Implizite Kommunikation**
- **Es gibt verschiedene Typen von Shared-Memory Systemen**
- **Programmierung mit ...**
  - > ... OpenMP
  - > ... Java-Threads

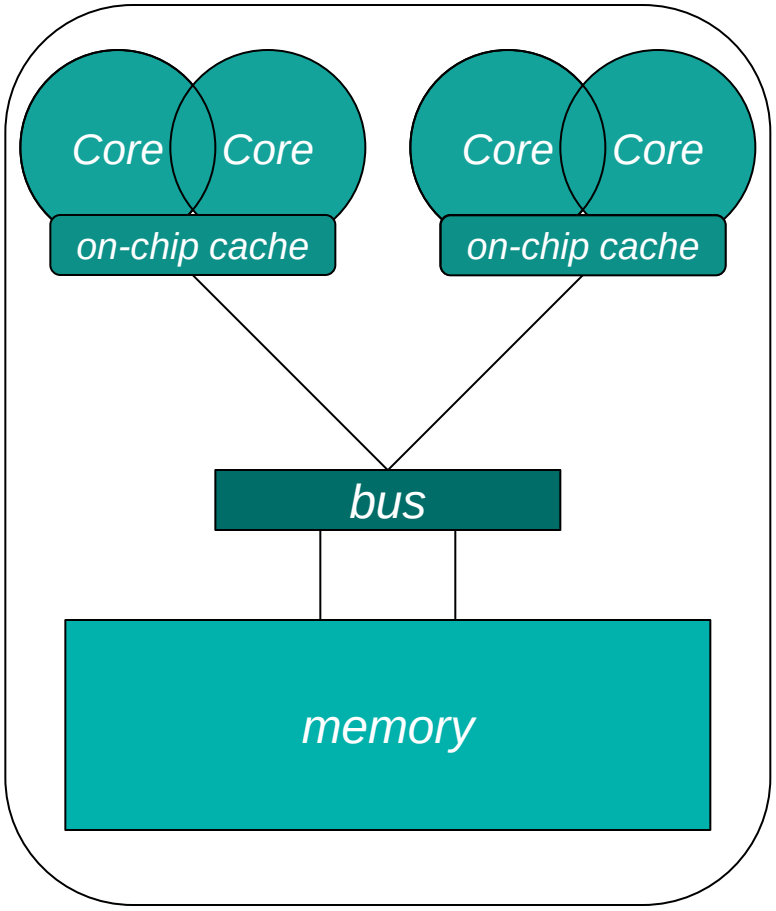


Auf den Speicher kann von allen Threads eines Programms zugegriffen werden



Kurz für *Symmetric Multi Processing*

- Zugriff auf den Hauptspeicher dauert für jeden Prozessorkern gleich lang
- Nur limitierte Skalierbarkeit
- Beispiel: *Intel Woodcrest*
  - > Zwei Kerne pro Chip, 3.0 GHz
  - > Jeder Chip hat 4 MB L2 on-chip cache, welcher von beiden Kernen genutzt wird



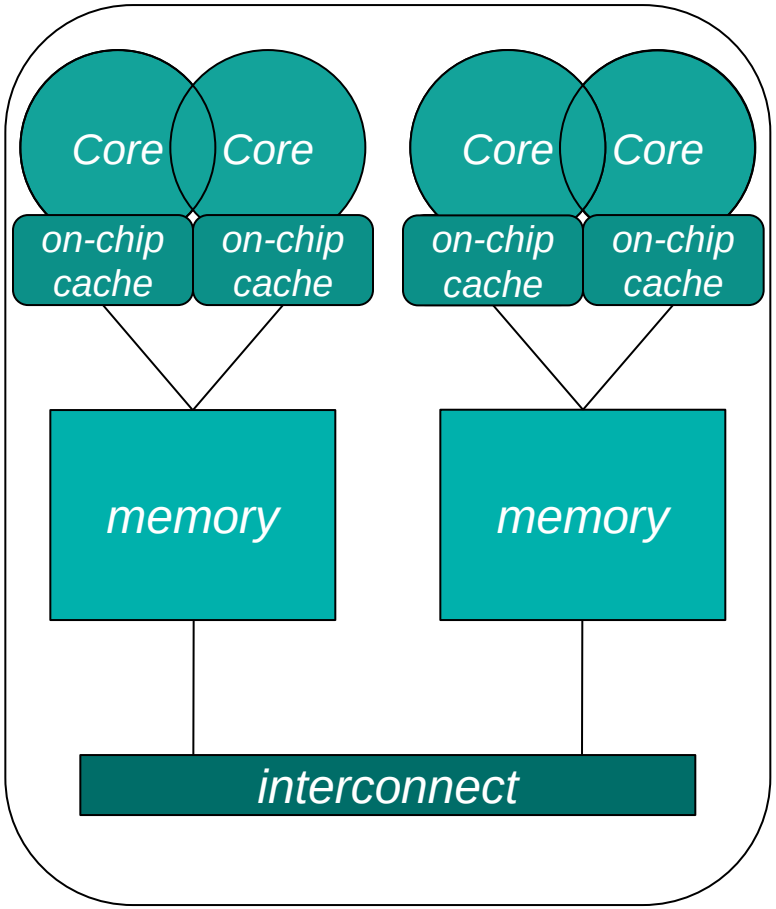
Parallele Rechnerarchitekturen
Cache-Kohärenz

- Der Einsatz multipler Caches erzeugt die Frage nach der Konsistenz der verschiedenen Speicher
- Ein Multiprozessorsystem ist Cache-Kohärent wenn:
  - Ein Wert der von einem Prozessor geschrieben wird irgendwann auch von anderen Prozessoren gesehen wird (write propagation)
  - Schreiboperationen von mehreren Prozessoren auf die gleiche Lokalität werden von allen Prozessoren in der gleichen Reihenfolge gesehen (write serialization)
- Prozessoren kommunizieren über Signale um die Cache-Kohärenz sicherzustellen

Parallele Rechnerarchitekturen
ccNUMA

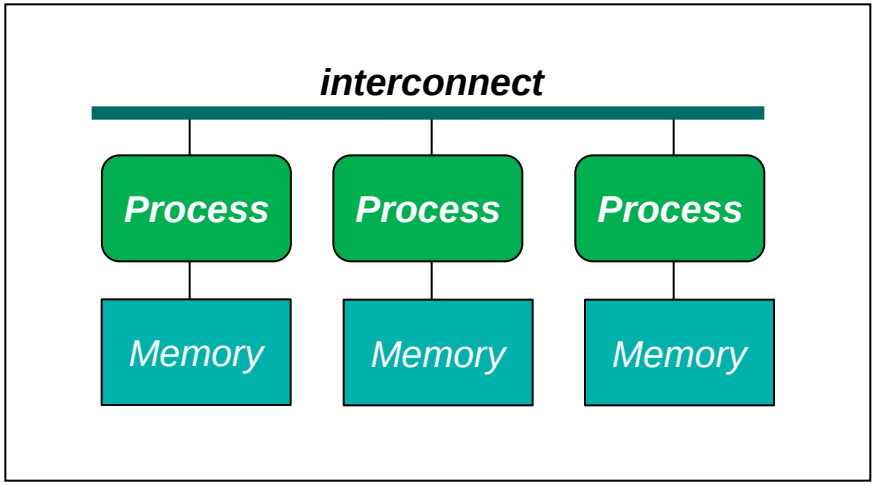
Kurz für *cache-coherent Non-Uniform Memory Architecture*

- Zugriff auf den Hauptspeicher dauert unterschiedlich lange, aber gemeinsamer Adressraum
- Gute Skalierbarkeit
- Beispiel: *AMD Opteron*
  - > Zwei Kerne pro Chip, 2.4 GHz
  - > Jeder Chip hat 1 MB eigenen L2-cache

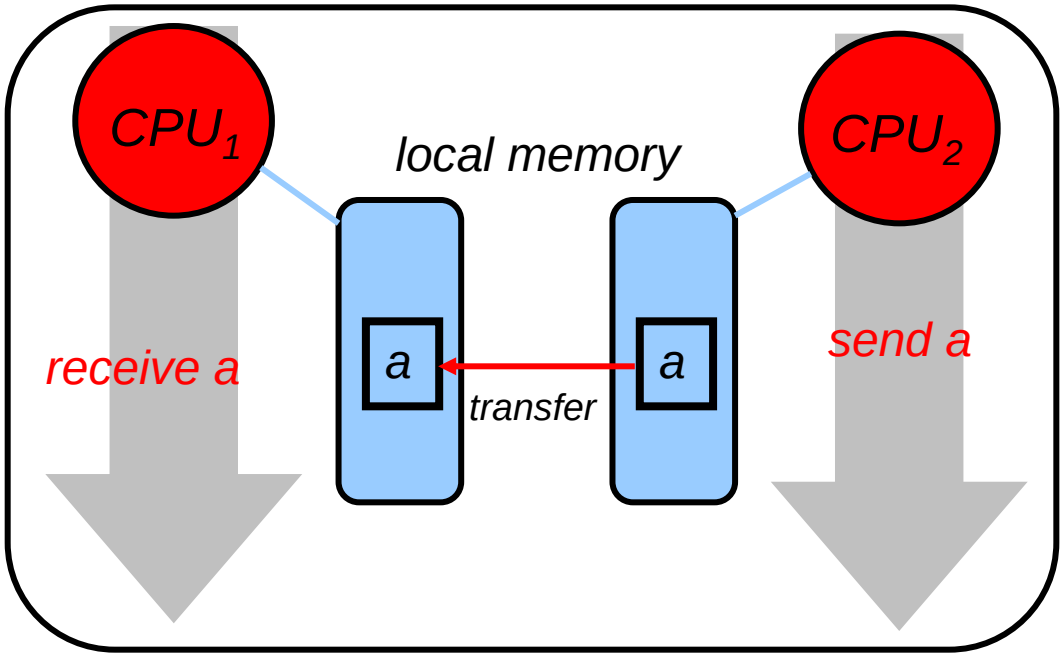


Parallele Rechnerarchitekturen
Distributed-Memory Systeme

- **Explizite Datenverteilung**
- **Explizite Kommunikation**
- **Gute Skalierbarkeit**
- **Programmierung mit MPI**



Jeder Prozess hat seinen eigenen Speicherbereich
Kommunikation per *Message Passing*



2 – Rechnerarchitekturen

Parallele Rechnerarchitekturen
Amdahl's Law

Parallele Rechnerarchitekturen
Speedup und Effizienz

- **Zeitverbrauch bei 1 CPU:** $T(1)$
- **Zeitverbrauch bei *p* CPUs:** $T(p)$
- **Speedup S:** $S(p) = \frac{T(1)}{T(p)}$
  > Gibt an, wieviel schneller die parallele Berechnung ist
- **Effizienz E:** $E(p) = \frac{S(p)}{p}$

Beispiel

- $T(1) = 6s,\ T(2) = 4s$
  - $S(2) = \frac{6}{4} = 1.5$
  - $E(2) = \frac{1.5}{2} = 0.75$

- **Idealfall**
- $S(p) = p$
- $E(p) = 1$

Parallele Rechnerarchitekturen
Amdahl's Law

Beschreibt den Einfluss des seriellen Teils eines Programms auf dessen Skalierbarkeit

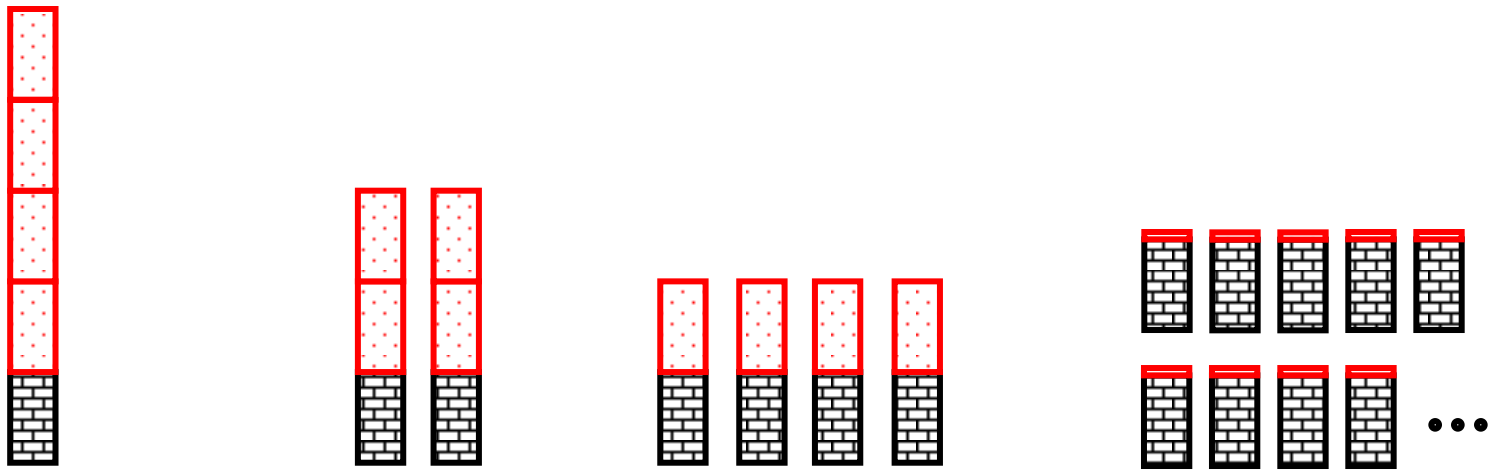
- $S(p) = \frac{T(1)}{T(p)} = \frac{T(1)}{f*T(1)+(1-f)*\frac{T(1)}{p}} = \frac{1}{f+\frac{1-f}{p}}$
  - F: serieller Teil des Programms (0 <= f<= 1)
  - T(1): Zeitverbrauch mit 1 CPU
  - T(p): Zeitverbrauch mit p CPUs
  - S(p): Speedup
  - E(p): Effizienz

- Overhead wird nicht berücksichtigt

- Skalierbarkeit jedes Programms ist durch den seriellen Programmteil limitiert

Parallele Rechnerarchitekturen
Speedup und Effizienz

Wenn 80% (gemessen an der Programmlaufzeit) der Arbeit parallelisiert werden können und "nur" 20% immer seriell ausgeführt werden müssen, dann ergibt sich der Speedup wie folgt:



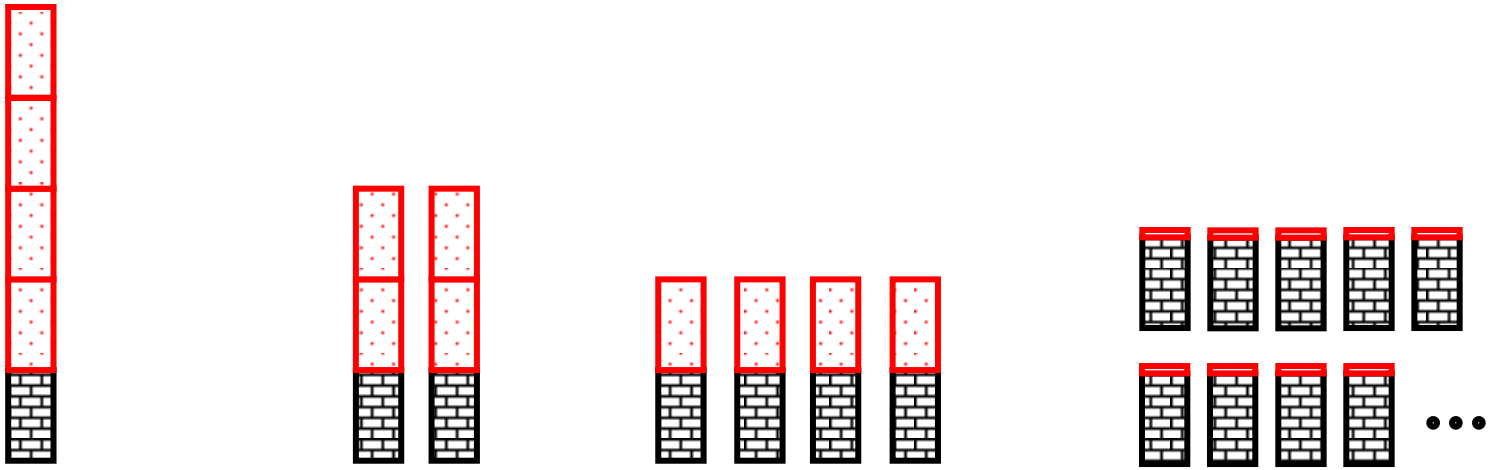
1 Prozessor:
Zeit: 100%
Speedup: 1
Effizienz: 1

2 Prozessoren:
Zeit: 60%
Speedup: 1.67
Effizienz: 0.83

4 Prozessoren:
Zeit: 40%
Speedup: 2.5
Effizienz: 0.63

∞ Prozessoren:
Zeit: 20%
Speedup: 5
Effizienz: 0

Wenn 80% (gemessen an der Programmlaufzeit) der Arbeit parallelisiert werden können und "nur" 20% immer seriell ausgeführt werden müssen, dann ergibt sich der Speedup wie folgt:



1 Prozessor:
Zeit: 100%
Speedup: 1
Effizienz: 1

2 Prozessoren:
Zeit: 60%
Speedup: 1.67
Effizienz: 0.83

4 Prozessoren:
Zeit: 40%
Speedup: 2.5
Effizienz: 0.63

∞ Prozessoren:
Zeit: 20%
Speedup: 5
Effizienz: 0

Nach der ersten Parallelisierung eines Programms wird man typischerweise eine Kurve wie folgt erhalten: