Strukturelemente von Parallelrechnern

- Parallelrechner besteht aus einer Menge von Verarbeitungselementen, die in einer koordinierten Weise, teilweise zeitgleich, zusammenarbeiten, um eine Aufgabe zu lösen

- Verarbeitungselemente können sein:
  - spezialisierte Einheiten, wie z.B. die Pipeline-Stufen eines Skalarprozessors oder die Vektor-Pipelines der Vektoreinheit eines Vektorrechners
  - gleichartige Rechenwerke, wie z.B. die Verarbeitungselemente eines Feldrechners
  - Prozessorknoten eines Multiprozessorsystems
  - vollständige Rechner, wie z.B. Workstations oder PCs eines Clusters
  - selbst wieder ganze Parallelrechner oder Cluster

Grenzbereiche von Parallelrechnern

- eingebettete Systeme als spezialisierte Parallelrechner
- Superskalar-Prozessoren, die feinkörnige Parallelität durch Befehls-Pipelining und Superskalar-Technik nutzen
- Mikroprozessoren arbeiten als Hauptprozessor teilweise gleichzeitig zu einer Vielzahl von spezialisierten Einheiten wie der Bussteuerung, DMA-,Graphikeinheit, usw.
- Ein-Chip-Multiprozessor
- mehrfädige (multithreaded) Prozessoren führen mehrere Kontrollfäden überlappt oder simultan innerhalb eines Prozessors aus
- VLIW- (Very Long Instruction Word)- Prozessor

Klassifikation von Parallelrechnern

- Klassifikation nach Flynn, d.h. Klassifikation nach der Art der Befehlsausführung
- Klassifikation nach der Speicherorganisation und dem Adressraum
- Konfigurationen des Verbindungsnetzwerks
- Varianten an speichergekoppelte Multiprozessorsysteme
- Varianten an nachrichtengekoppelte Multiprozessorsysteme

Klassifikation nach Flynn

Zweidimensionale Klassifizierung mit Kriterium Anzahl der Befehls- und Datenströme

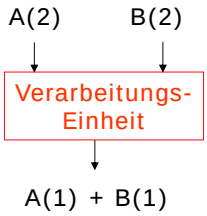
- Rechner bearbeitet zu einem Zeitpunkt einen oder mehrere Befehle
- Rechner bearbeitet zu einem Zeitpunkt einen oder mehrere Datenwerte

⇒ Damit vier Klassen von Rechnerarchitekturen

- **SISD**: Single Instruction, Single Data
  Ein Befehl verarbeitet einen Datensatz. (herkömmliche Rechnerarchitektur eines seriellen Rechners)
- **SIMD**: Single Instruction, Multiple Data
  Ein Befehl verarbeitet mehrere Datensätze, z.B. N Prozessoren führen zu einem Zeitpunkt den gleichen Befehl aber mit unterschiedlichen Daten aus.
- **MISD**: Multiple Instruction, Single Data
  Mehrere Befehle verarbeiten den gleichen Datensatz. (Diese Rechnerarchitektur ist nie realisiert worden.)
- **MIMD**: Multiple Instruction, Multiple Data
  Unterschiedliche Befehle verarbeiten unterschiedliche Datensätze.
  Dies ist das Konzept fast aller modernen Parallelrechner.

SISD Architektur

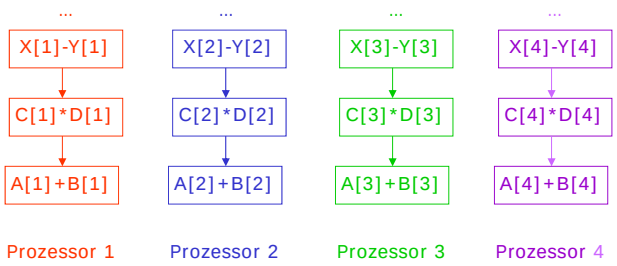
- Klassische Struktur eines seriellen Rechners:
  Nacheinander werden verschiedene Befehle ausgeführt, die z.B. einzelne Datenpaare verknüpfen



- Moderne RISC (Reduced Instruction Set Computer) Prozessoren verwenden Pipelining:
  - Mehrere Funktionseinheiten, die gleichzeitig aktiv sind.
  - Operationen sind in Teiloperationen unterteilt.
  - In jedem Takt kann eine Funktionseinheit (z.B. Addititionseinheit) eine neue Operation beginnen.
  - D.h. hohe interne Parallelität nutzbar

SIMD Architektur (Prozessorarray)

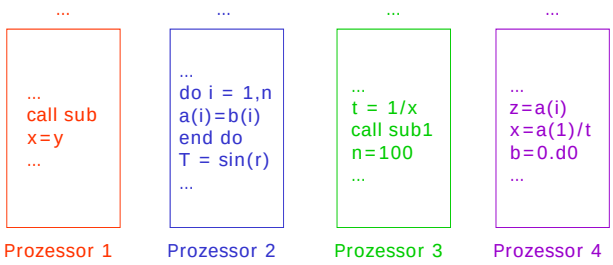
- Mehrere Prozessoren führen zu einem Zeitpunkt den gleichen Befehl aus
- Rechner für Spezialanwendungen (z.B. Bildverarbeitung, Spracherkennung)
- I.A. sehr viele Prozessorkerne (tausende Kerne in einem System)
- Beispiele: Graphikprozessoren, Numerische Coprozessoren



- Mittlerweile auch innerhalb einzelner Funktionseinheiten zu finden

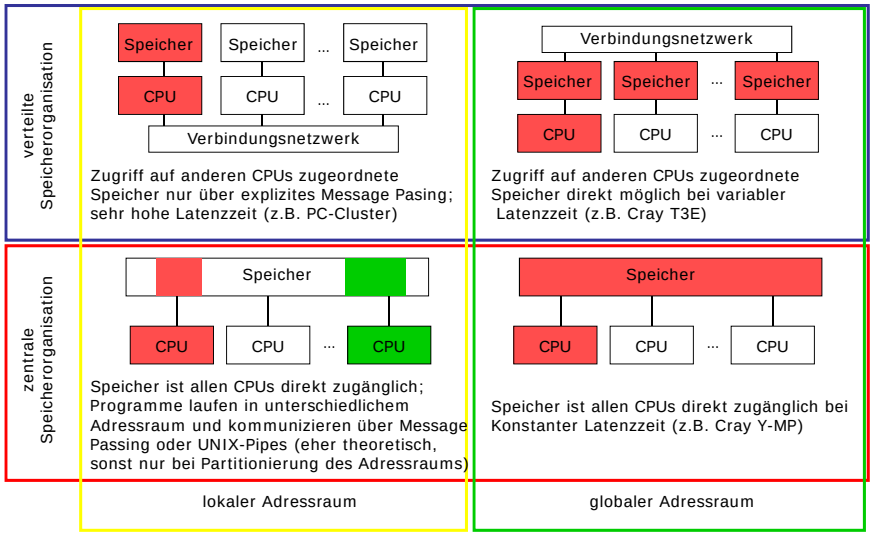
MIMD Architektur

Mehrere Prozessoren führen unabhängig voneinander unterschiedliche Instruktionen auf unterschiedlichen Daten aus:

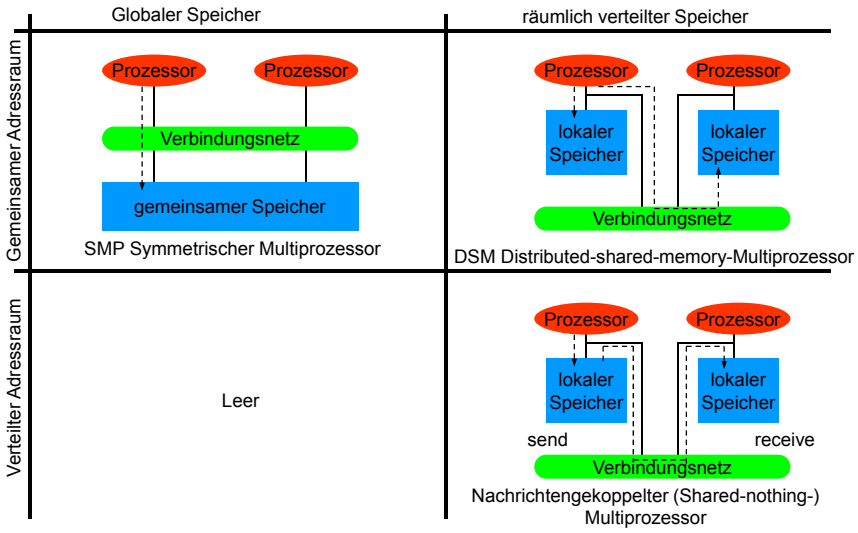


- Fast alle aktuellen Systeme entsprechen dieser Architektur.

Speicherorganisation und Adressraum



Konfiguration der Verbindungsnetzwerke



Arten von Multiprozessorsystemen

- Bei speichergekoppelten Multiprozessorsystemen besitzen alle Prozessoren einen gemeinsamen Adressraum.
  Kommunikation und Synchronisation geschehen über gemeinsame Variablen.
  - **Symmetrisches Multiprozessorsystem (SMP)**: ein globaler Speicher
  - **Distributed-Shared-Memory-System (DSM)**: gemeinsamer Adressraum trotz räumlich verteilter Speichermodule

- Beim nachrichtengekoppelten Multiprozessorsystem besitzen alle Prozessoren nur räumlich verteilte Speicher und prozessorlokale Adressräume.
  Die Kommunikation geschieht durch Austausch von Nachrichten.
  - **Massively Parallel Processors (MPP)**, eng gekoppelte Prozessoren
  - Verteiltes Rechnen in einem **Workstation-Cluster** (z.B. Linux Cluster).
  - **Grid-/Cloud-Computing**: Zusammenschluss weit entfernter Rechner

Speichergekoppelte Multiprozessorsysteme

- Alle Prozessoren besitzen **einen** gemeinsamen Adressraum; Kommunikation und Synchronisation geschieht über **gemeinsame Variablen**.
- Uniform-Memory-Access-Modell (UMA):
  - Alle Prozessoren greifen in gleichermaßen auf einen gemeinsamen Speicher zu. Insbesondere ist die *Zugriffszeit* aller Prozessoren auf den gemeinsamen Speicher *gleich*.
    Jeder Prozessor kann zusätzlich einen *lokalen Cache-Speicher* besitzen.
    Typische Beispiel: die symmetrischen Multiprozessorsysteme (SMP)
- Nonuniform-Memory-Access-Modell (NUMA):
  - Die *Zugriffszeiten* auf Speicherzellen des gemeinsamen Speichers *variieren* je nach dem Ort, an dem sich die Speicherzelle befindet.
    Die Speichermodule des gemeinsamen Speichers sind physisch auf die Prozessoren aufgeteilt.
  - Typische Beispiele: Distributed-Shared-Memory-Systeme.

Nachrichtengekoppelte Multiprozessorsysteme

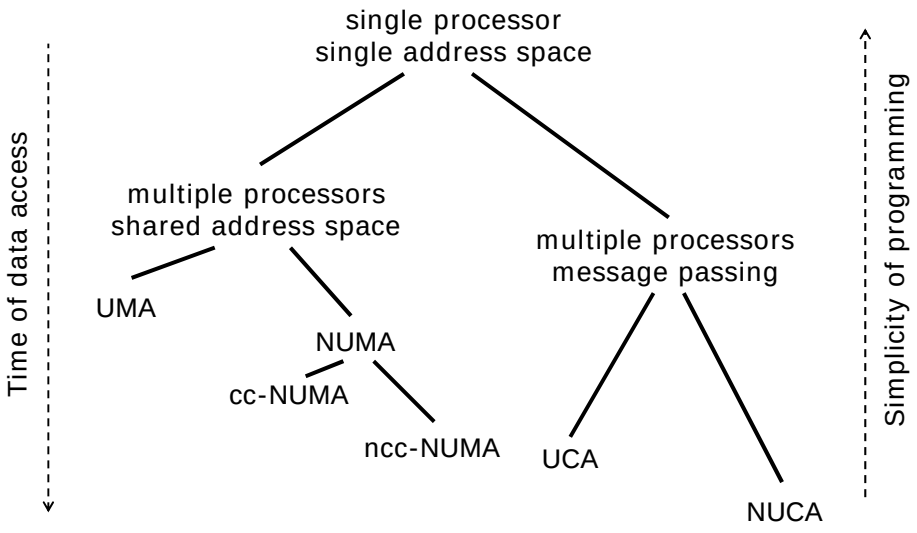
- **Uniform-Communication-Architecture-Modell** (UCA):
  Zwischen allen Prozessoren können gleich lange Nachrichten mit einheitlicher Übertragungszeit geschickt werden.

- **Non-Uniform-Communication-Architecture-Modell** (NUCA):
  Die Übertragungszeit des Nachrichtentransfers zwischen den Prozessoren ist je nach Sender- und Empfänger-Prozessor unterschiedlich lang.

Speicher- vs. Nachrichtenkopplung

- Distributed-Shared-Memory-Systeme sind NUMAs: Die Zugriffszeiten auf Speicherzellen des gemeinsamen Speichers variieren je nach Ort, an dem sich die Speicherzelle befindet.
  - **cc-NUMA** (Cache-coherent NUMA): Cache-Kohärenz wird über das gesamte System gewährleistet, z.B. *HPE Integrity MC990 X Server (ehemals SGI Altix)*
  - **ncc-NUMA** (Non-Cache-coherent NUMA): Cache-Kohärenz wird nur innerhalb eines Knotens gewährleistet, z.B. *InfiniBand Cluster mit RDMA*
  - **COMA** (Cache-only-Memory-Architecture): Der Speicher des gesamten Rechners besteht nur aus Cache-Speicher. Nur in einem kommerziellen System realisiert (ehemalige Firma *Kendall Square Research - KSR*)
- Nachrichten gekoppelte Multiprozessorsysteme sind **NORMAs** (No-remote-memory-access-Modell) oder Shared-nothing-Systeme, z.B. *Linux Cluster Systeme*

Transfer Time vs. Simplicity of Programming



Zusammenfassung: Klassifizierung

Klassifizierung nach

- Befehls- und Datenströme,
- Speicherorganisation,
- Verbindungsnetzwerk
  - weitere Details später in der Vorlesung

Quantitative Bewertung von Parallelrechnern

Merkmale: Geschwindigkeit, Auslastung

- Ausführungszeit T eines parallelen Programms
  - Zeit zwischen dem Starten der Programmausführung auf einem der Prozessoren bis zu dem Zeitpunkt, an dem der **letzte** Prozessor die Arbeit an dem Programm beendet hat
- Während der Programmausführung sind alle Prozessorkerne in einem der drei Zustände
  - rechnend
  - kommunizierend
  - untätig

Ausführungszeit T

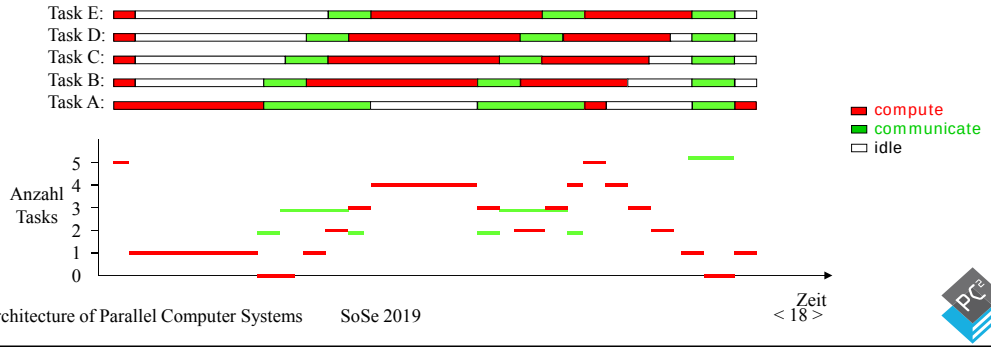
Ausführungszeit T eines parallelen Programms auf einem dediziert zugeordneten Parallelrechner setzt sich zusammen aus:

- Berechnungszeit T_{comp}
  - Zeit für die Ausführung von Rechenoperationen
- Kommunikationszeit T_{com}
  - Zeit für Sende- und Empfangsoperationen
- Untätigkeitszeit T_{idle}
  - Zeit für Warten (auf zu empfangende oder zu sendende Nachrichten)

Es gilt: $T \approx T_{comp} + T_{com} + T_{idle}$

Parallelitätsprofil

- Parallelitätsprofil zeigt die vorhandene Parallelität in einem parallelen Programm (einer konkreten Ausführung)
  - Grafische Darstellung:
    Auf der x-Achse wird die Zeit und auf der y-Achse die Anzahl paralleler Aktivitäten aufgetragen.
  - Perioden von Berechnungs- Kommunikations- und Untätigkeitszeiten sind erkennbar.



Beschleunigung und Effizienz

- Beschleunigung (Leistungssteigerung, Speedup):

$$S(n) = \frac{T(1)}{T(n)}$$

- Effizienz:

$$E(n) = \frac{S(n)}{n}$$

- T(1) Ausführungszeit auf einem Einprozessorsystem
- T(n) Ausführungszeit auf einem System mit n Prozessoren

Die „Zeit“ ist auch in Schritte oder Takte messbar.

Skalierbarkeit

Skalierbarkeit eines Parallelrechners

- Das Hinzufügen von weiteren Verarbeitungselementen führt zu einer kürzeren Gesamtausführungszeit, ohne dass das Programm geändert werden muss.
- Wichtig für die Skalierbarkeit sind jeweils angemessene Problemgrößen.
- Bei fester Problemgröße und steigender Prozessorzahl wird ab einer bestimmten Prozessorzahl eine Sättigung eintreten. Die Skalierbarkeit ist in jedem Fall beschränkt (*strong scaling*).
- Steigt mit Anzahl an Prozessoren auch die Problemgröße, muss dieser Effekt bei skalierenden Hardware- oder Software-Systemen nicht auftreten (*weak scaling*).

Gute Skalierbarkeit:
Lineare Steigerung der Beschleunigung mit einer Effizienz nahe Eins.

(Allgemeine) Gesetz von Amdahl

Amdahl´s Law:

„The performance improvements to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.“

Damit ist der Speedup ist durch die Ausführungszeit beschränkt, die durch eine Verbesserung *E* erzielt werden kann

Es gilt:

$$\text{Speedup}_E = \frac{\text{ExecTime}_{\text{without E}}}{\text{ExecTime}_{\text{with E}}} = \frac{\text{Performance}_{\text{with E}}}{\text{Performance}_{\text{without E}}}$$

Daraus folgt: (siehe nächste Folie)

(Allgemeine) Gesetz von Amdahl

Oftmals lässt sich nur ein Teil des Programms beschleunigen.

$$\text{ExecTime}_{\text{new}} = \text{ExecTime}_{\text{old}} \cdot \left[(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right]$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExecTime}_{\text{old}}}{\text{ExecTime}_{\text{new}}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}} + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right)}$$

$$\text{Fraction}_{\text{enhanced}} \to 0 \quad \Rightarrow \quad \text{Speedup}_{\text{overall}} \to 1$$

$$\text{Fraction}_{\text{enhanced}} \to 1 \quad \Rightarrow \quad \text{Speedup}_{\text{overall}} \to \text{Speedup}_{\text{enhanced}}$$

Wobei

$\text{Fraction}_{\text{enhanced}}$ = Anteil der Ausführungszeit, der von der Beschleunigung *E* profitiert.

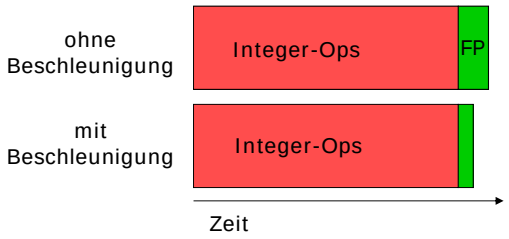
$\text{Speedup}_{\text{enhanced}}$ = Ausführungszeit des Originalprogrammteils dividiert durch Ausführungszeit des verbesserten Programmteils

Gesetz von Amdahl: Beispiel

Fließkomma-Operationen können um den Faktor zwei beschleunigt werden, aber nur 10% aller Instruktionen sind FP-Ops.

$$\text{ExecTime}_{new} = \text{ExecTime}_{old} \cdot \left(0.9 + \frac{0.1}{2}\right) = 0.95 \cdot \text{ExecTime}_{old}$$

$$\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053$$



Gesetz von Amdahl angewendet auf Parallelität

$$T(n) = T(1) \cdot a + T(1) \cdot \frac{1 - a}{n}$$

wobei :

a = der Anteil der Ausführungszeit des nur sequentiell ausführbaren Programmteils

$$\Rightarrow \quad S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1) \cdot (1 - a)/n + T(1) \cdot a}$$

$$= \frac{1}{(1 - a)/n + a}$$

$$\Rightarrow \quad S(n) \leq \frac{1}{a}$$

Grenzen der Skalierbarkeit

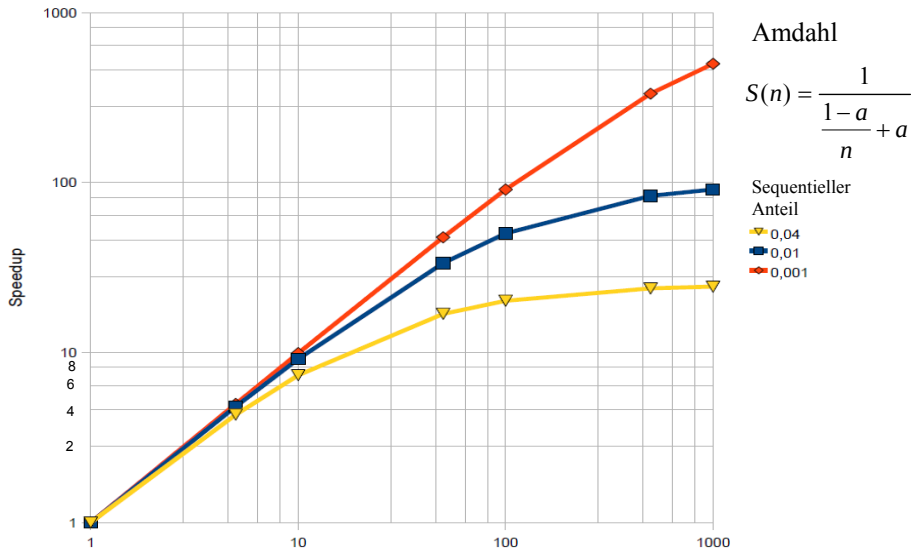
Beispiel: 100 Sekunden Ausführungszeit auf einem Prozessor

Zeit in Sekunden.

Anzahl Proz.	sequentieller Anteil							
	100%	50%	25%	12.5%	6.25%	3.125%	1.5625%	0.7812%
1	100	100	100	100	100	100	100	100
2	100	75	62,5	56,25	53,125	51,5625	50,78125	50,39062
4	100	62,5	43,75	34,375	29,6875	27,34375	26,17187	25,58593
8	100	56,25	34,375	23,4375	17,96875	15,23437	13,86718	13,18359
16	100	53,125	29,6875	17,96875	12,10937	9,179687	7,714843	6,982421
32	100	51,5625	27,34375	15,23437	9,179687	6,152343	4,638671	3,881835
64	100	50,78125	26,17187	13,86718	7,714843	4,638671	3,100585	2,331542
128	100	50,39062	25,58593	13,18359	6,982421	3,881835	2,331542	1,556396
256	100	50,19531	25,29296	12,84179	6,616210	3,503417	1,947021	1,168823
512	100	50,09765	25,14648	12,67089	6,433105	3,314208	1,754760	0,975036
1024	100	50,04882	25,07324	12,58549	6,341557	3,219604	1,658630	0,878143

Speed-Up = 114

Speedup Diagramm

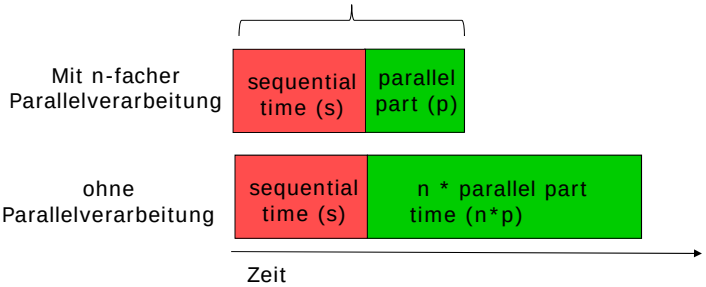


Weitergehende Skalierung

- Gesetz von Amdahl berücksichtigt nicht, dass oftmals das zu lösende Problem mit Anzahl an Prozessoren wächst
- Der sequentielle Anteil kann sich mit der Problemgröße bzw. Anzahl an Prozessoren relativ verkleinern (ggf. ist der seq. Anteil konstant)
- „Gesetz von Gustafson“

$S(n)=a+n\cdot(1-a)$

mit $a = s/(s+p)$ Anteil der Laufzeit des sequentiellen Teilstückes des parallel ausgeführten Programms



$a \to 0 \Rightarrow S(n) \to n$

$a \to 1 \Rightarrow S(n) \to 1$

Gesetz von Gustafson - Herleitung

$T(1)=s+n\cdot p$ und $T(n)=s+p$

sei $a = \frac{s}{s+p} \Rightarrow 1-a = \frac{p}{s+p}$

$$S(n) = \frac{T(1)}{T(n)} = \frac{s+n\cdot p}{s+p}$$

$$\Rightarrow S(n) = \frac{s}{s+p} + \frac{n\cdot p}{s+p}$$

$$\Rightarrow S(n) = a+n\cdot(1-a)$$

Gesetz von Gustafson

Wie groß darf der seq. Anteil bei Ausführung mit einem Prozessor sein?

Sei

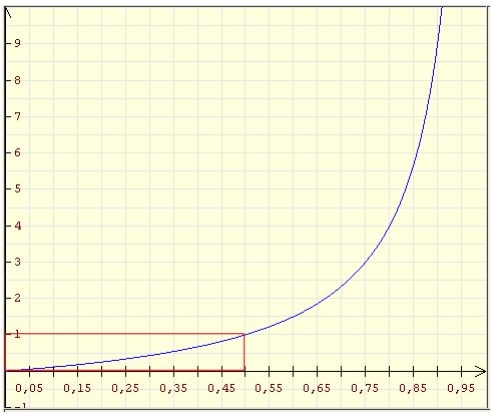
$$s \le p \quad \Rightarrow \quad a = \frac{s}{s+p} \le \frac{1}{2}$$

Dann gilt:

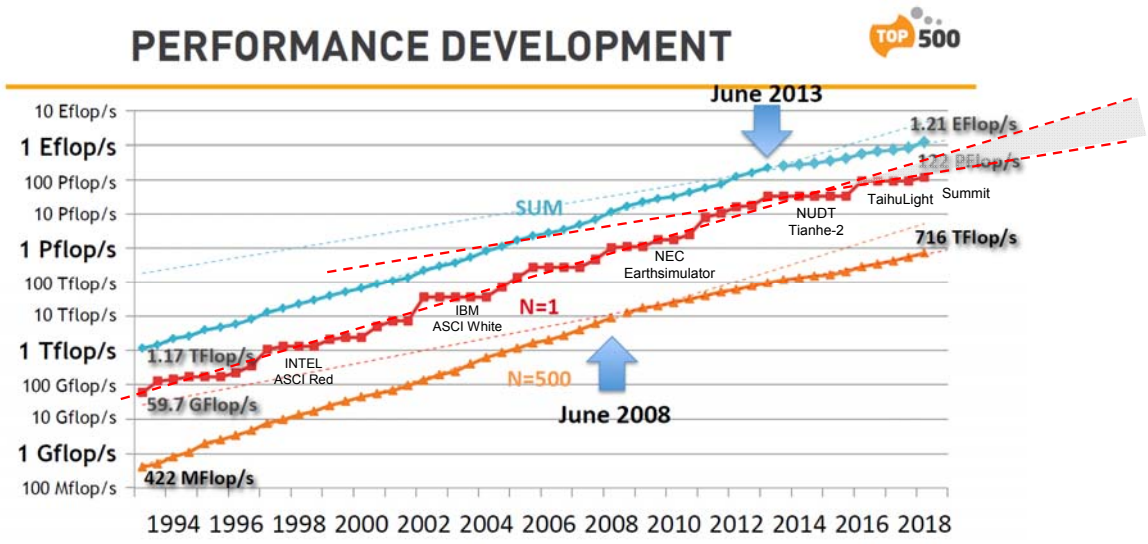
$$\frac{s}{s+n\cdot p} \le \frac{s}{n\cdot p} = \frac{s}{n\cdot(s/a-s)}$$

$$= \frac{1}{n}\cdot\frac{s}{(s-s\cdot a)/a} = \frac{1}{n}\cdot\frac{s}{s\cdot(1-a)/a}$$

$$= \frac{1}{n}\cdot\frac{a}{1-a} \le \frac{1}{n}$$



f(a)=a/(1-a)



Worldwide fastest HPC System

Source: Top500.org

Performance is measured with perfect scaling Linpack Benchmark

Moore´s Law: Rechenleistung

Gordan Moore, Mitbegründer von Intel Corp.:

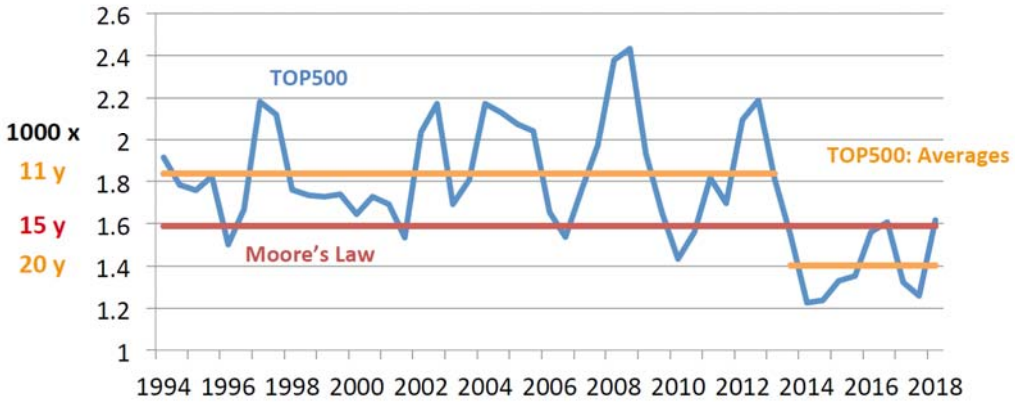
„Die Anzahl der auf einem Silizium-Chip passenden Elemente verdoppelt sich ungefähr jedes Jahr.“

Gordan Moore, Electonics Magazin (1965)

„Rechenleistung, die man zu einem konstanten Betrag X kaufen kann, verdoppelt sich in etwa alle 18 Monate.“

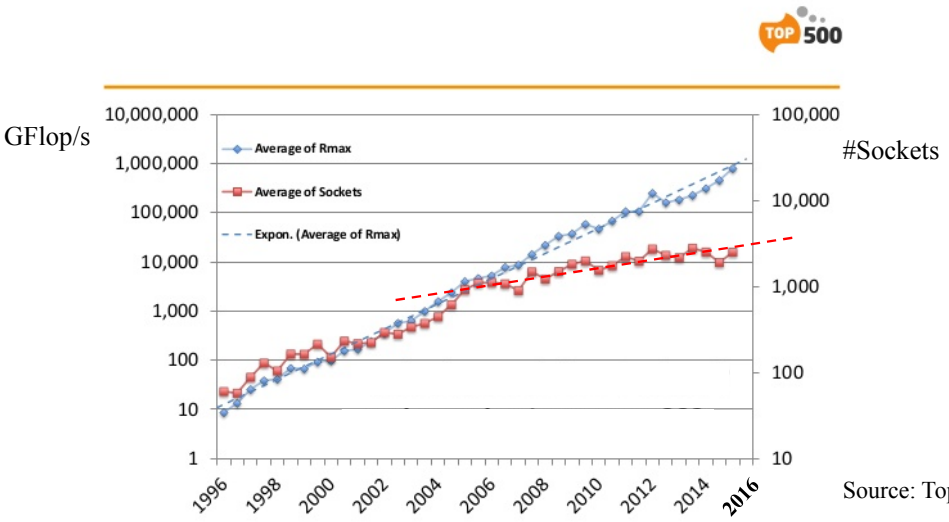
angepasste Interpretation (21th century)

ANNUAL PERFORMANCE INCREASE OF THE TOP500



Source: Top500.org

Trend: Performance vs. #Processor Sockets



Source: Top500.org

Trend – Processor Cores per Socket

Data source: Top500.org