Paralleles Rechnen in der Physik

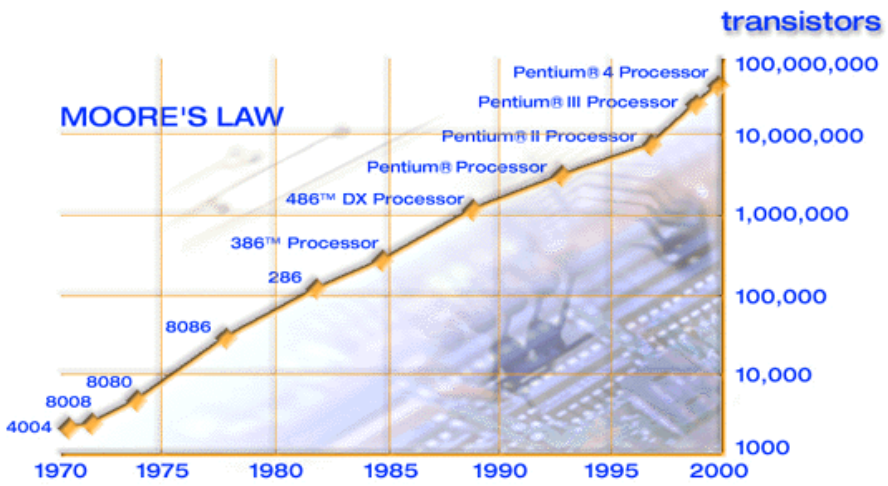
Georg Hager
Regionales Rechenzentrum Erlangen
HPC Services
Universität Erlangen-Nürnberg
07.05. 2002

Überblick

- Motivation
- Parallelität verstehen
- Paralleles Rechnen in Beispielen
- Aktuelle Hardware im Supercomputing
- Führung durch die Rechnerräume des RRZE

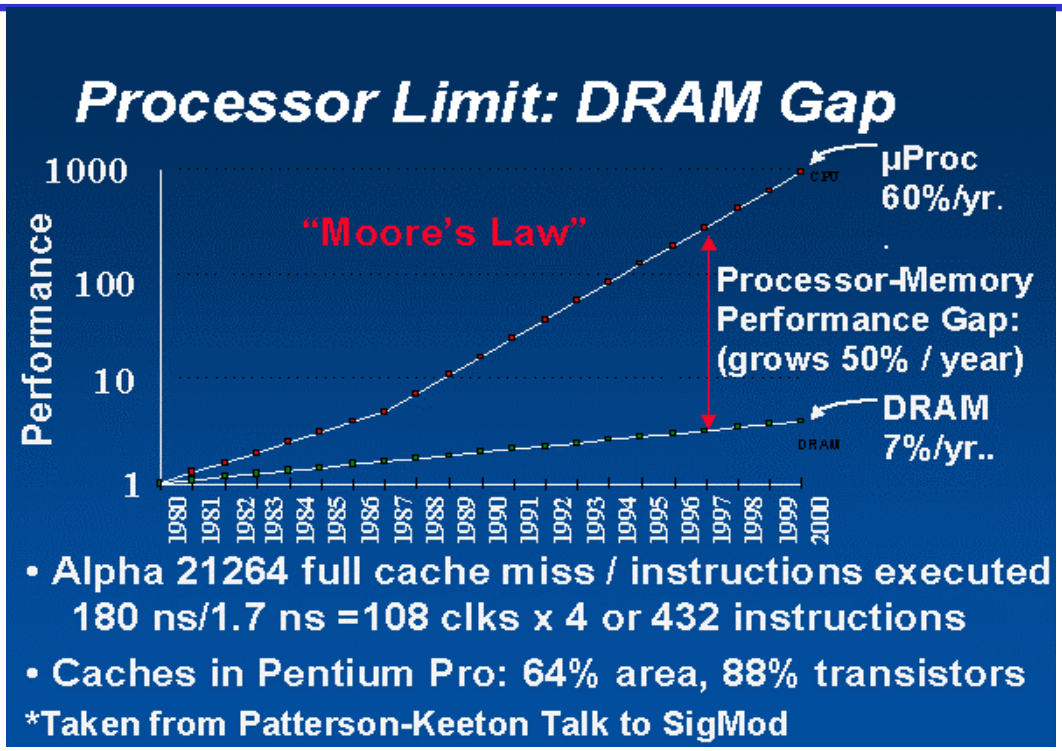
Motivation:
Warum reicht der Pentium nicht aus?

- Chips haben in den vergangenen Jahrzehnten einen immensen Leistungsschub verzeichnet
  - Moore's Law: Verdopplung der Chipkomplexität bzw. -Leistung alle 18 Monate



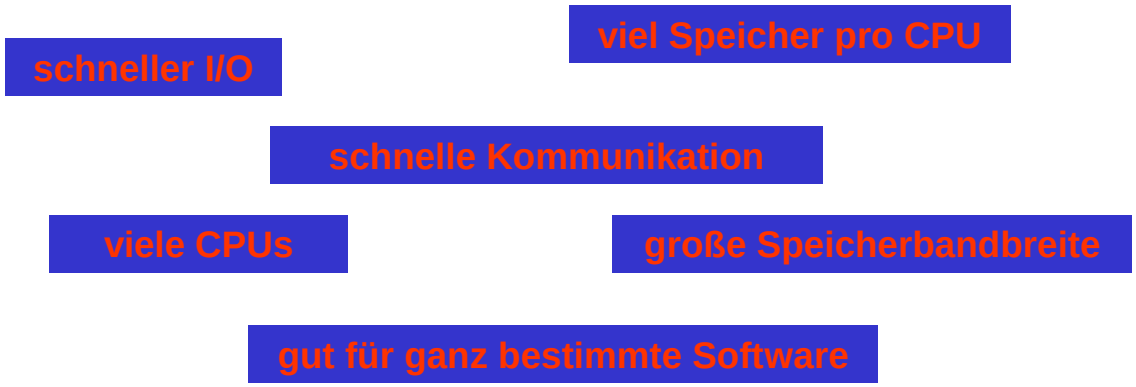
Quelle: Intel Corp.

Motivation:
Schere zwischen Prozessor und Speicher



"Geschwindigkeit messen"

- Was ist ein gutes Maß für die Geschwindigkeit eines Computers?
  - Er soll mein Problem schnell lösen!
  - das bedeutet: die Anforderungen, die mein Problem stellt, sollen bestmöglich erfüllt sein
- Was sind also die Anforderungen der "Large Scale User"?



"Geschwindigkeit messen"

- Alle Aspekte der Benutzeranforderungen lassen sich nicht zusammen in eine Zahl pressen
- Dennoch: Bekannteste Bewertungsziffer ist der LINPACK-Benchmark
  - Lösung eines großen, linearen Gleichungssystems Ax=b
  - Messung der Performance in Flops/s

1 Flop = 1 Fließkomma-Operation (+ - * /)

- Analog: MFlops/s, GFlops/s, TFlops/s etc.
- Rangliste der 500 "schnellsten" Rechner der Welt:

`http://www.top500.org/`

- Zweimal im Jahr (Juni/November) aktualisiert
- Zum Vergleich: aktueller PC hat R_{max} = 1-2 GFlops/s

TOP500 supercomputer SITES — HPC @RRZE

Rank	Man.	Computer	R_{max} (GFlops)	Installation Site	Country	Proc.	R_{peak} (GFlops)
1	IBM	ASCI White	7226	Lawrence Livermore National Laboratory	USA	8192	12288
2	Compaq	AlphaServer SC	4059	Pittsburgh SCC	USA	3024	6048
3	IBM	SP Power3	3052	NERSC/LBNL	USA	3328	4992
4	Intel	ASCI Red	2379	Sandia	USA	9632	3207
5	IBM	ASCI Blue-Pacific	2144	Lawrence Livermore	USA	5808	3868
6	Compaq	AlphaServer SC	2096	Los Alamos National Laboratory	USA	1536	3072
7	Hitachi	SR8000/MPP	1709	University of Tokyo	Japan	1152	2074
8	SGI	ASCI Blue Mountain	1608	Los Alamos	USA	6144	3072
9	IBM	SP Power3	1417	NAVOCEANO	USA	1336	2004
10	IBM	SP Power3	1272	Deutscher Wetterdienst	Germany	1280	1920

Stand: November 2001

"Geschwindigkeit messen": Die Rolle der Parallelität

HPC @RRZE

- **Enormes Anwachsen der Performance des Spitzenreiters über die Jahre:**
  - **Moore's Law: Verdopplung alle 18 Monate**
  - **1993: 59,7 GFLOPs LINPACK**
  - **1999: 2380 GFLOPs LINPACK**
  - **2000: 4039 GFLOPs LINPACK**

  Performance der Nr. 1 steigt schneller als von Moore's Law vorausgesagt (Faktor 67 statt 26 in 7 Jahren)!

  - **⇒ Massive Parallelität gibt einen "extra Schub"**

Warum reicht der Pentium nicht aus?
Ganz einfach: Er ist zu langsam und zu klein!

- Die Definition eines "schnellen" Rechners ändert sich mit der Zeit:

> "A supercomputer is a computer that is just one generation behind the requirements of the large scale users"
>
> Neil Lincoln (CDC Cyber 205 [1981])

- Was tut ein "Large Scale User"?

Die zwei "Limetes" des Supercomputing:

Durchsatz	Capability
"Aldi-Limes"	"Rockefeller-Limes"
Viele kleine Probleme	wenige große Probleme
Wenig Ressourcen pro Problem	enorme Ressourcen
"Trivial parallel"	Parallelisierung notwendig

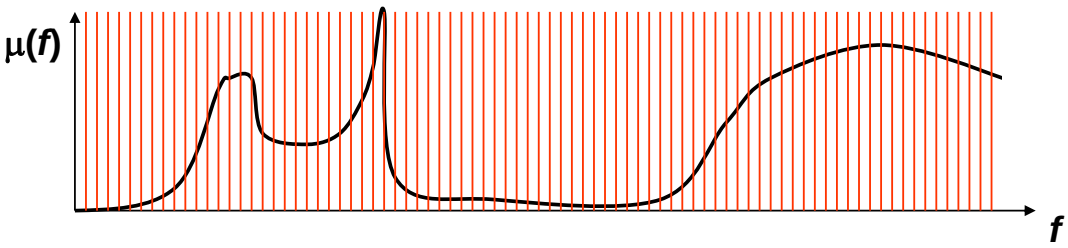
Warum reicht der Pentium nicht aus?
Das Durchsatzproblem

- Sogar wenn ein Einzelproblem schnell abgearbeitet ist, kann die schiere Zahl der Probleme nicht nacheinander bewältigt werden
- Typisches Beispiel: Parameterstudien
  - "Abgrasen" eines großen, eventuell multidimensionalen Parameterraumes (Temperatur, Dichte, Anfangsbedingungen ...)
  - Einzelne Läufe müssen keine Daten miteinander kommunizieren
  - Einzelprobleme laufen unter Kontrolle eines "Master-Programmes" oder des Benutzers selbst
- Parallele Abarbeitung vieler Einzelprobleme lastet den Rechner meist gut aus (feine Granularität)
- Kommunikationshardware kann billig gehalten werden
  - Einfaches Ethernet genügt meist
- Konzentration auf Spitzenleistung des Einzelrechners
  - Ausgewogenheit des Gesamtsystems steht hintenan

Beispiel für ein Durchsatzproblem: Absorptionsspektrum von Molekülen

- Modellierung der elektronischen Struktur eines Moleküls
  - Numerische Lösung der Schrödinger-Gleichung
  - Anregung von außen durch oszillierendes elektrisches Feld
- Spektrum: Durchstimmen der Anregungsfrequenz über einen großen Bereich
  - Berechnungen für jede Frequenz sind i.W. gleich aufwendig
  - Viele Frequenzen notwendig, um komplettes Spektrum zu errechnen

⇒ **Ideales Durchsatzproblem!**



Warum reicht der Pentium nicht aus? Das Durchsatzproblem

- Lösung des Durchsatzproblems: massivster Einsatz von vernetzter, preiswerter "Aldi-Hardware"

Warum reicht der Pentium nicht aus? Das Capability-Problem

- Speicherverbrauch und Laufzeit sprengen die Möglichkeiten eines Einzel-PC
  - Technologische Grenzen für Moore's Law in unter 20 Jahren erreicht!
- Parallelisierung nicht trivial, da Zerlegung in unabhängige Einzelprobleme nicht möglich
  - Verschiedene Strategien zur Parallelisierung bieten sich an (s. später)
  - Kommunikation ist oft nicht zu vernachlässigen
  - Gute Auslastung des Gesamtsystems ist nicht einfach zu erreichen (s. später)
- Kommunikationshardware ist extrem teuer
  - dazu zählt auch die "Kommunikation" zum Speicher der CPUs!
  - keine Lösungen "von der Stange"
- Ausgewogenes System (Spitzenleistung/Bandbreite) ist anzustreben

Beispiel für ein Capability-Problem: Strömungsmechanik

- Strömungsmechanik (CFD):
  - Grid: 1000 x 1000 x 1000 Gitterpunkte $O(n^3)$
  - Variablen pro Gitterpunkt (u,v,w,p,T,...): 20
  - Operationen zur Berechnung jeder Variable: 100
  - Zeitschritte: 10000 O(n)
  - Gesamtzahl Fließkomma-Operationen:

1000 x 1000 x 1000 x 20 x 100 x 10000 Flop=

20 000 000 000 000 000 Flop=

2×10^{16} Flop = **20 000 Tflop**

- Speicheranforderung: ca. 160 GByte
- Langzeitspeicherung von Ergebnissen: 160 x 10 = 1.6 Tbyte

Beispiel für ein Capability-Problem: Strömungsmechanik

- **Pentium IV, 2GHz (ohne Berücksichtigung des Speicherproblems):**
  - Max. Leistung: 2 GFLOP/s = 2.0 x 10^9, Rechenzeit:

$$\frac{2 \times 10^{16}\ \text{Flop}}{2 \times 10^{9}\ \text{Flop/s}} = 10\,000\,000\ \text{s} = 115\ \text{Tage}$$

- **Moderner Supercomputer (Hitachi SR8000):**
  - Max. Leistung: 2.0 TFLOP/s = 2.0 x 10^{12}
  - Rechenzeit:

Realität: Lücke noch viel größer!

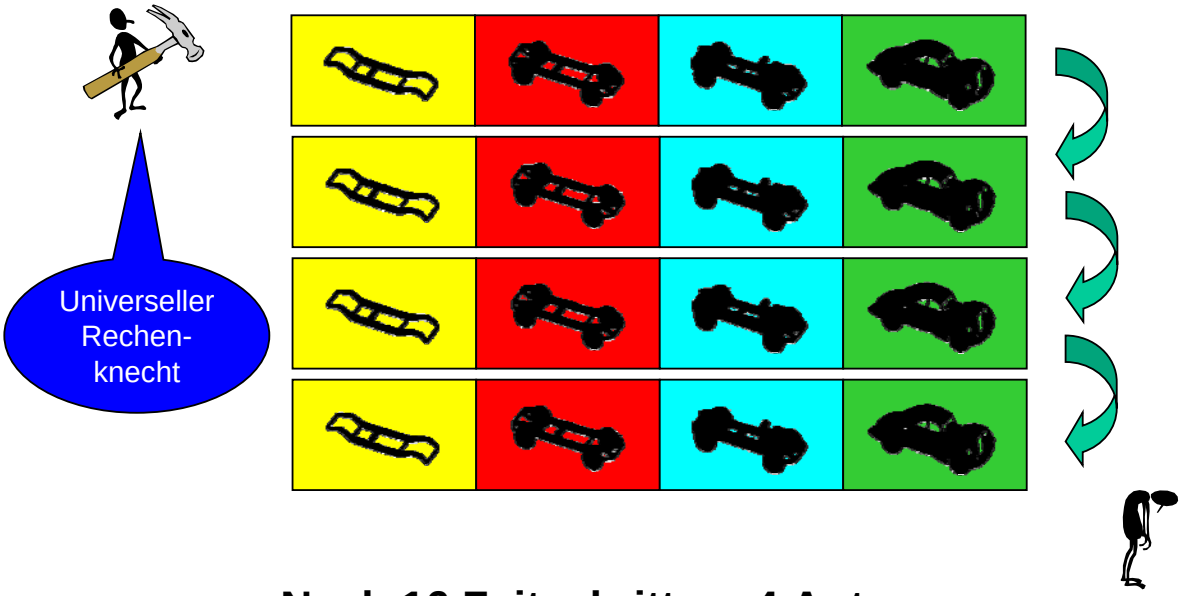
$$\frac{2 \times 10^{16}\ \text{Flop}}{2 \times 10^{12}\ \text{Flop/s}} = 10\,000\ \text{s} = 2.8\ \text{h}$$

Warum reicht der Pentium nicht aus? Das Capability-Problem

- **Lösung des Capability-Problems: speziell angefertigte Superrechner mit außergewöhnlicher Bandbreite**

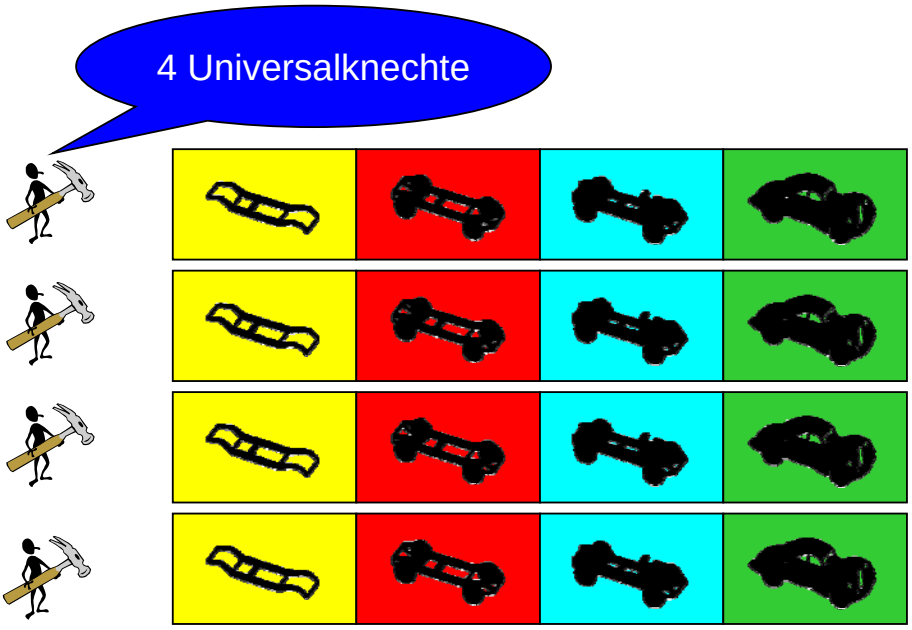
Quelle: LRZ München

Parallelität verstehen: Sequenzielle Arbeit



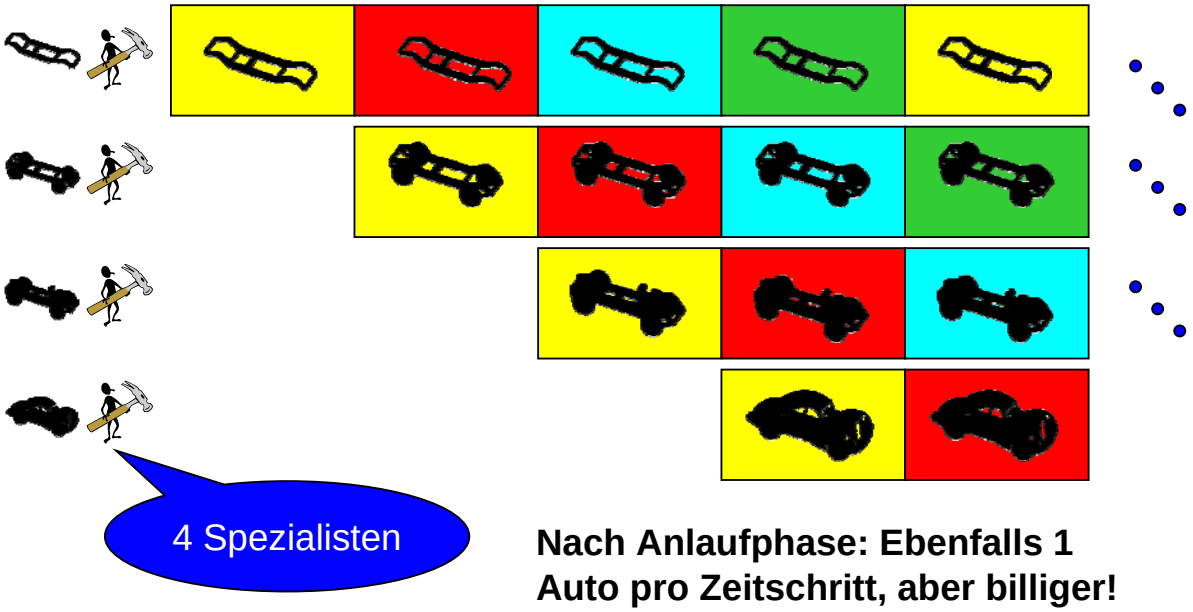
Nach 16 Zeitschritten: 4 Autos

Parallelität verstehen: Ideal parallele Arbeit



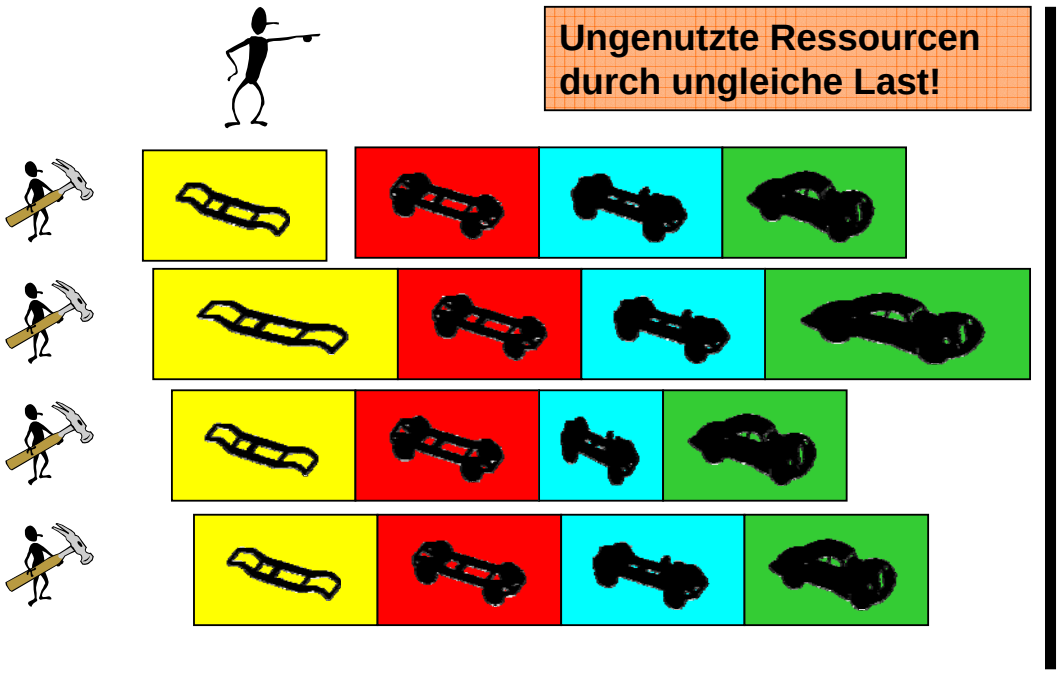
Faktor 4 schneller, aber: Universalgenies sind teuer und langsam!

Parallelität verstehen: Spezialisierung und Fließbandarbeit



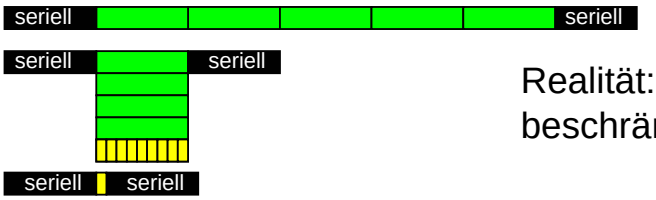
Nach Anlaufphase: Ebenfalls 1 Auto pro Zeitschritt, aber billiger!

Parallelität verstehen: Grenzen der Parallelität

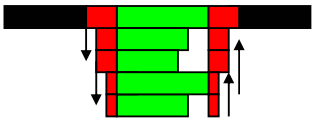


Grenzen der Parallelität: Amdahls Gesetz

Idealsituation: Arbeit vollständig und gleichmäßig verteilbar

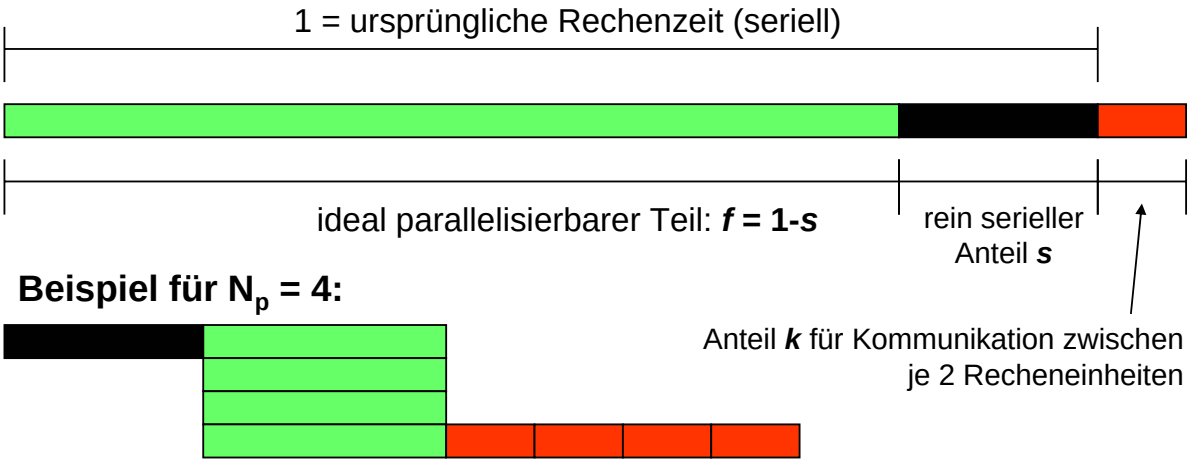


Realität: rein serielle Anteile beschränken maximalen "Speedup"



Noch schlimmer: Kommunikation verschlechtert Speedup bei großen "Arbeiterzahlen"

Grenzen der Parallelität: Amdahls Gesetz

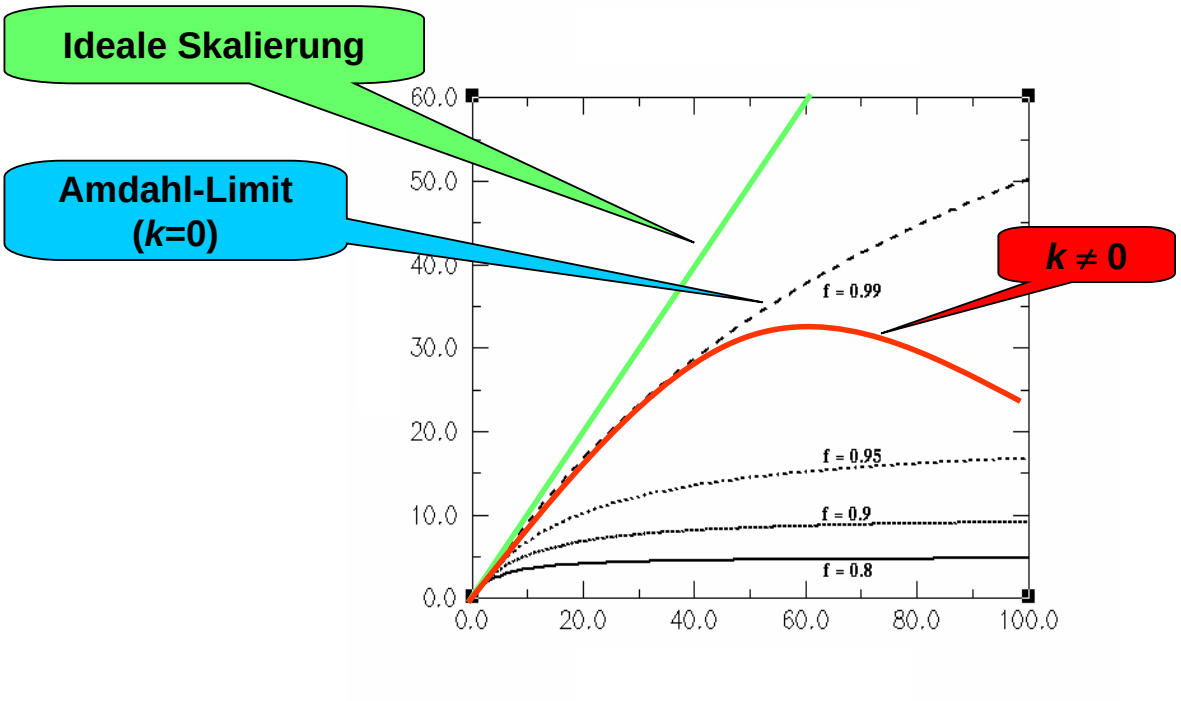


Beispiel für $N_p = 4$:

Allgemeine Formel für den Speedup (worst case):

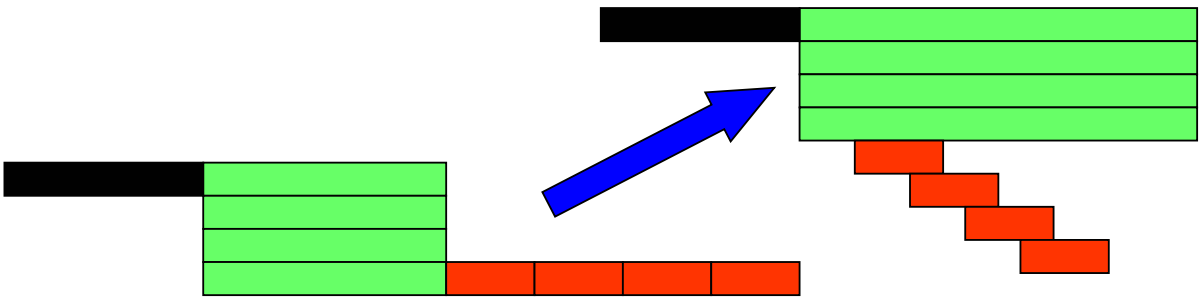
$$P = \frac{1}{s + \frac{1-s}{N_p} + N_p k}$$

Grenzen der Parallelität: Amdahls Gesetz



Grenzen der Parallelität: Wie man Amdahls Gesetz austrickst

- Serieller Anteil ist oft unabhängig von der Problemgröße
  - Vergrößerung des Problems schafft bessere Parallelität
- Kommunikation ist nicht notwendigerweise vollkommen seriell
  - Blockierungsfreie Kreuzschienen-Netzwerke können viele Nachrichten gleichzeitig zwischen Rechenknoten übermitteln (technische Maßnahme)
  - Kommunikation kann u.U. mit sinnvoller Rechenarbeit überlappt werden (Implementierung, Algorithmik)



Parallelität verstehen: Granularität

- Numerische Probleme haben oft unterschiedliche Ebenen von Granularität
  Beispiel: Wettersimulation

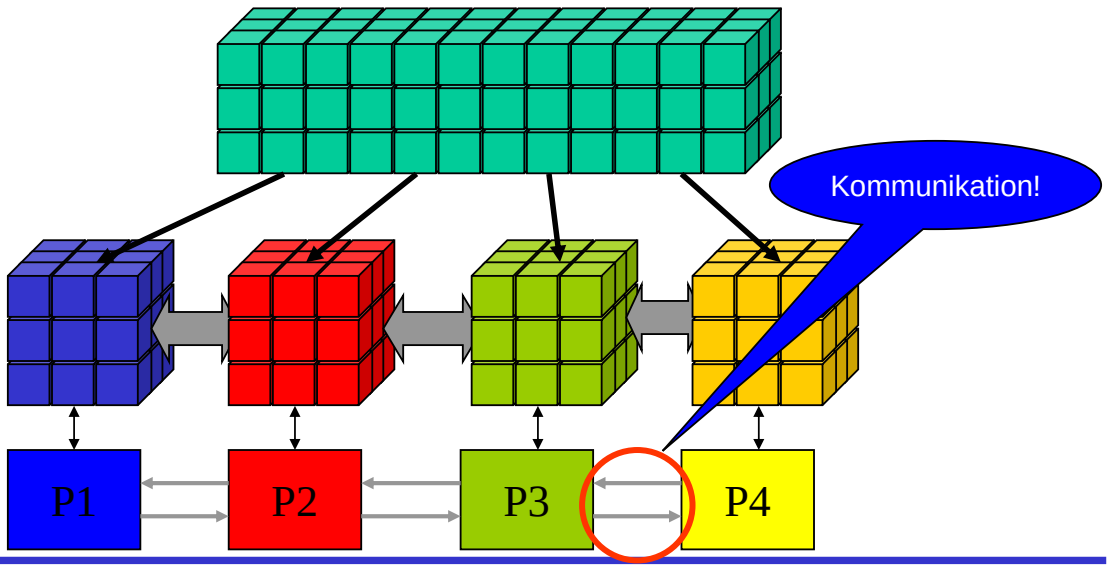
grob ↓ fein

- Oberste Ebene: Simulation verschiedener Teilaspekte (Wind, Temperatur, Niederschlag...) — Physik
- Zerlegung des Simulationsgebietes in Blöcke
- Jeder Block besteht aus N Gitterplätzen, an denen Observablen gemessen werden — Mathematik
- Numerische Verfahren zur Lösung der Gleichungen arbeiten mit Matrizen und Vektoren
- Unterste Ebene: Instruktions- und Datenströme in der CPU — Silizium

- Aufgabe des Programmierers: Abbildung der Granularitätsstruktur des Problems auf die Rechnerstruktur (Parallelisierung)!

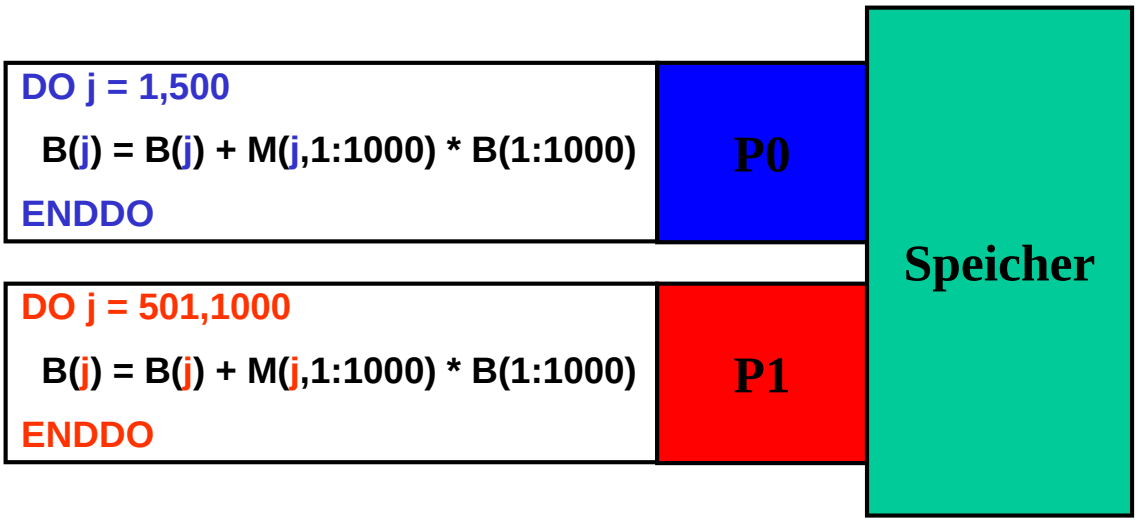
Grobkörnige Parallelität: Gebietszerlegung

- Aufteilung des Rechengebietes in Blöcke, die in unterschiedlichen Teilen des Parallelrechners bearbeitet werden
  - Px kann auch aus mehreren CPUs bestehen



Mittelkörnige Parallelität: Schleifen

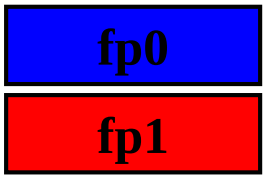
- Blockweise Aufteilung der Arbeit in Schleifen auf verschiedene CPUs, die auf gemeinsamem Speicher arbeiten
  - Beispiel: Matrix-Vektor-Multiplikation



Feinkörnige Parallelität: Multiple Funktionseinheiten auf dem Chip

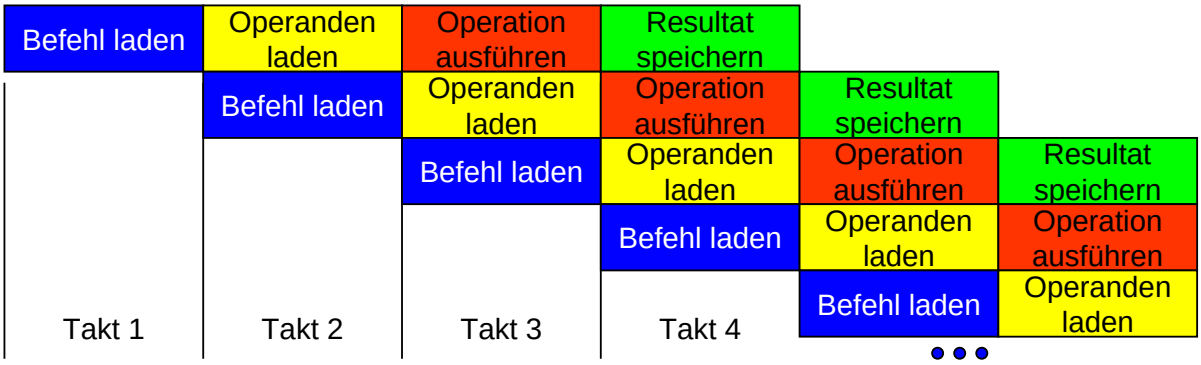
- Beispiel: Prozessor mit zwei Funktionseinheiten, die je eine Multiplikation pro Takt (z.B. 1.0 ns bei 1 GHz) ausführen können
  - Alternierende Iterationen laufen jeweils in unterschiedlicher Funktionseinheit ab
  - Effektive Verdopplung der Fließkomma-Leistung bei solchen Schleifen

```
DO i = 1 , 1000 ,2
  C(i) = A(i) * B(i)
  C(i+1) = A(i+1) * B(i+1)
ENDDO
```



Feinkörnige Parallelität: Fließbandarbeit (Pipelining)

- Auf unterster Ebene einer CPU sind alle Operationen aus elementaren Bausteinen zusammengesetzt:
  - Operand(en) laden
  - Arithmetische Operation ausführen
  - Ergebnis speichern
  - Schleifenzähler justieren/prüfen auf Schleifenende
  - Verzweigung/Sprung
- Einfaches Beispiel: Instruktionspipeline



Beispiele zur Parallelisierung: Matrix-Vektor-Multiplikation

- MVM auf Parallelrechner mit verteiltem Speicher
  - häufige Operation bei Eigenwertproblemen
- Mathematische Formulierung:

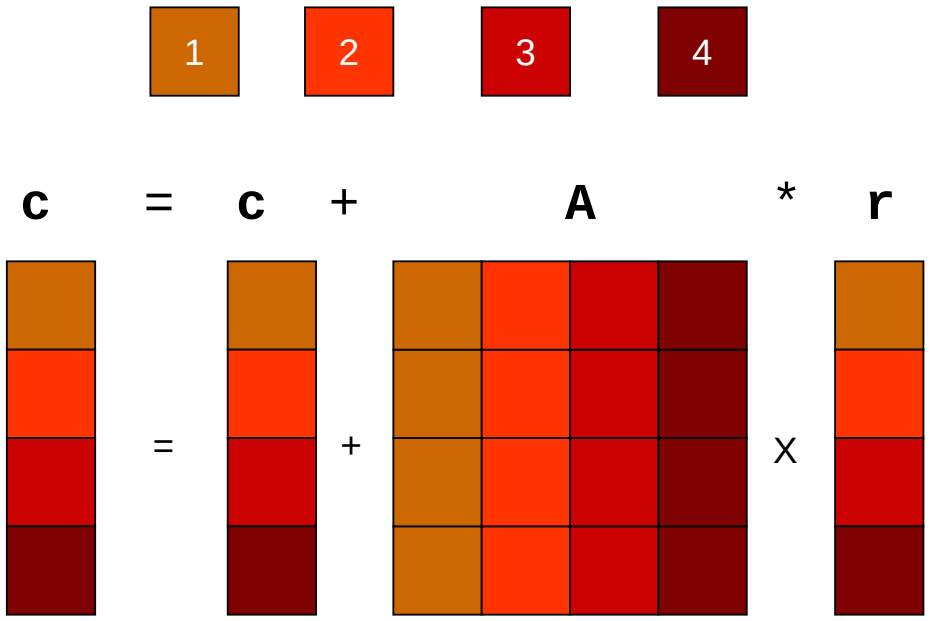
$$c_i = c_i + \sum_j A_{ij} r_j \quad (i,j=1,...,n\_dim)$$

- Serieller Code:

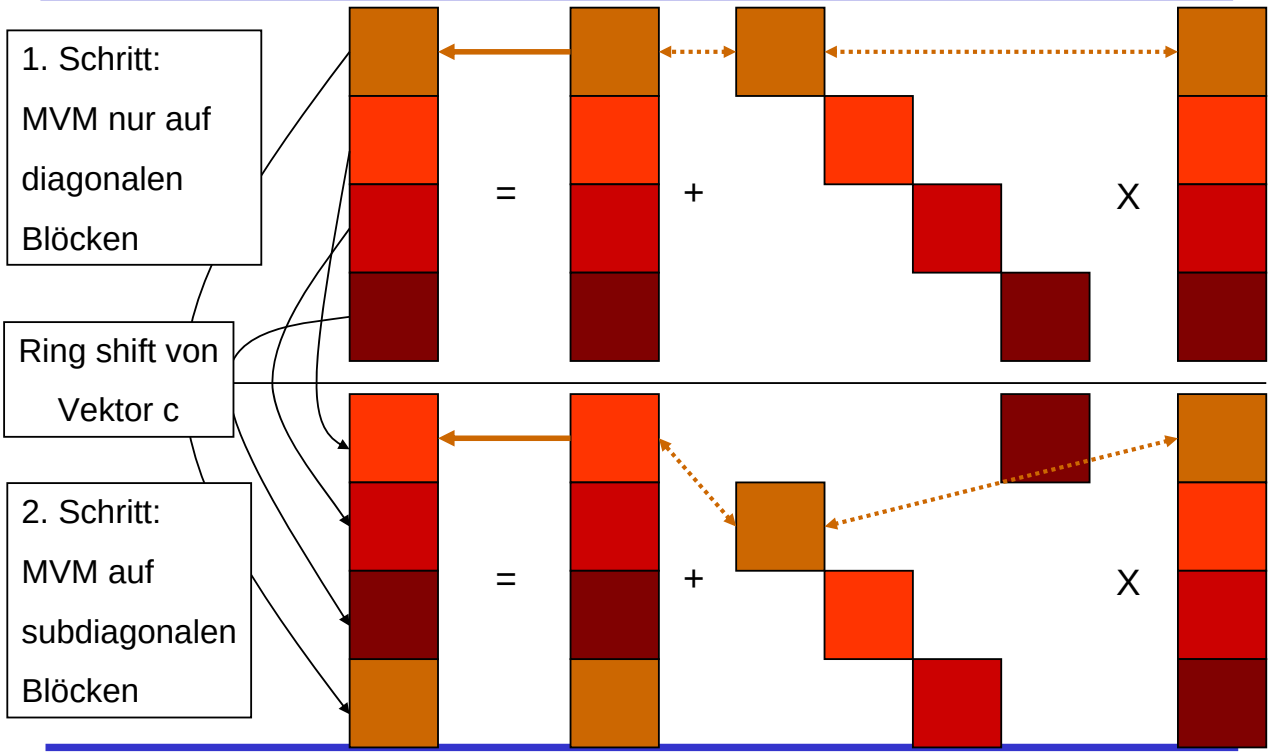
```
do i = 1 , n_dim
   do j = 1 , n_dim
      c( i ) = c( i ) + A( i , j ) * r( j )
   enddo
enddo
```

Beispiele zur Parallelisierung: MVM

- Verteilung der Matrix und des Vektors auf 4 Rechner



Beispiele zur Parallelisierung: MVM

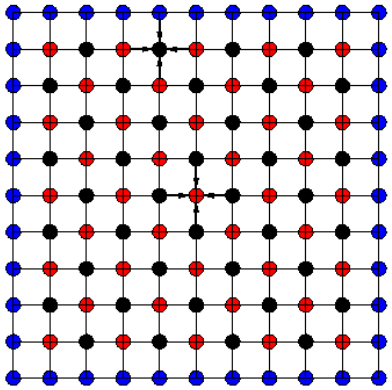


Beispiele zur Parallelisierung: Randwertproblem

- **Problem: Temperaturverteilung auf quadratischer Platte bei fester Vorgabe am Rand**
  - **Lösung der Laplace-Gleichung unter Randbedingungen:**

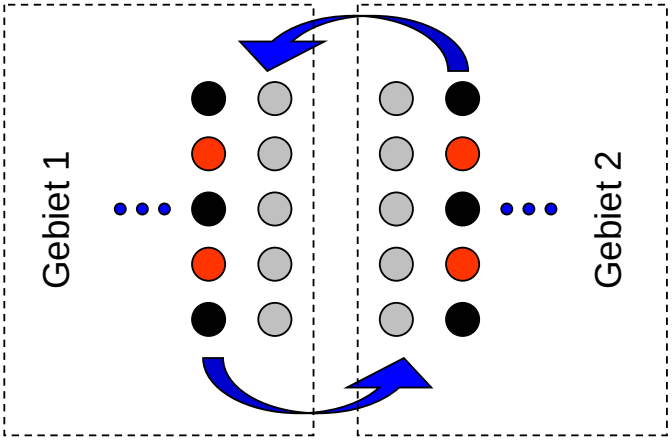
$$\Delta T = 0$$

  - **Numerisch: Abbildung auf NxN-Gitter**
  - **Verfahren: "Relaxation"**
    T an einem Punkt ist der Mittelwert üver die 4 nächsten Nachbarn
  - **Iteration: Bei genügend durchgängen durch das Gitter konvergiert T gegen die Lösung**
- **Parallelisierung mittels Gebietszerlegung**
  - **Was passiert mit Randzellen eines Gebietes?**



Beispiele zur Parallelisierung: Randwertproblem

- **Das Problem der Gebietsränder wird durch "Geisterzellen" gelöst**
  - **Jedes Gebiet ist an jeder Kontaktfläche eine Zelle breiter als notwendig**
  - **Nach jeder Iteration werden die Inhalte der Geisterzellen mit dem Inhalt der Nachbar-Randzellen gefüllt**



Supercomputing: Hochleistungsrechner in der Realität

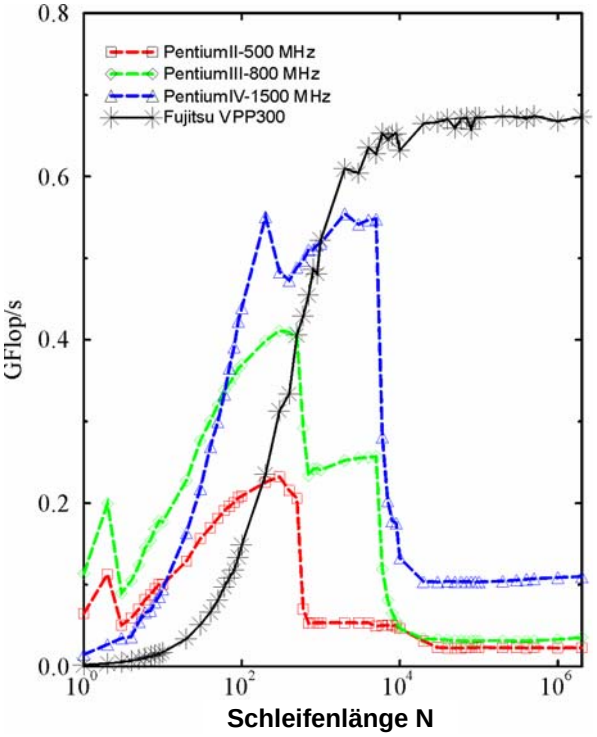
- **Supercomputer unterscheiden sich in den Ausprägungen der Parallelität**
  - **Massiv parallele Systeme**
  - **Vektorsysteme bzw. Vektor- Parallelrechner**
  - **Hybride Systeme**

- **Bestimmende Größe ist nicht die Spitzen- oder LINPACK-Performance, sondern die Speicherbandbreite!**
  - **Wie viele Daten können die CPUs pro Sekunde laden/speichern?**
  - **Speicher- und Kommunikationsbandbreite sind die kostenbestimmenden Faktoren**
  - **Es gibt immer Applikationen, die auf große Bandbreite angewiesen sind**

Speicherbandbreite als bestimmender Faktor: Vektortriade

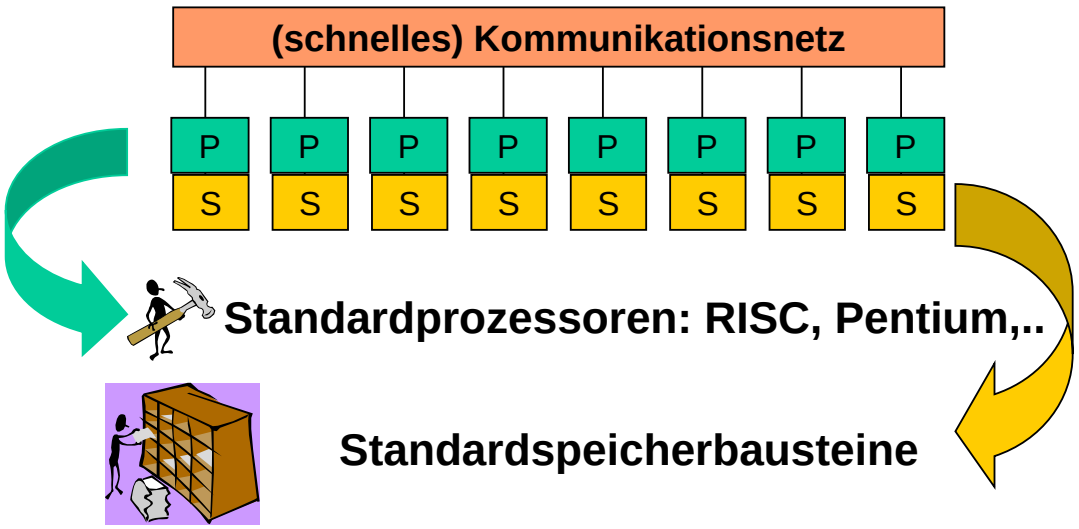
- Standardtest: Vektor-Triade
- Extrem sensitiv auf Speicherbandbreite

```
DO J=1,NTIMES
  DO I=1,N
    A(I) = B(I) + C(I)*D(I)
  ENDDO
ENDDO
```



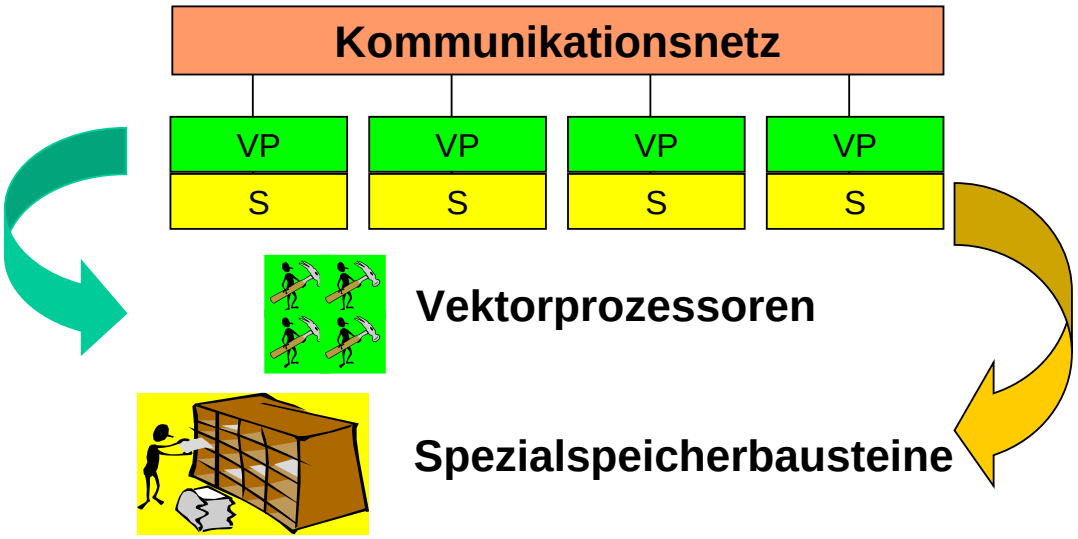
Supercomputing: Massiv parallele Systeme

- Typische CPU-Zahlen: 100 - 10000



Supercomputing: Vektorsysteme

- Extremes Pipelining auf Datenebene (SIMD: Single Instruction Multiple Data)
- Extrem hohe Speicherbandbreite
- Typische CPU-Zahlen: 10-100



Supercomputing heute: Aktuelle Hardware

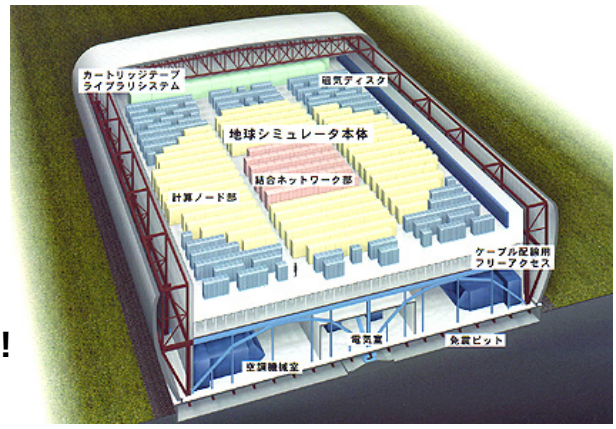
❑ IBM ASCI White (Lawrence Livermore National Laboratory, USA)

- 8192 CPUs (512 Knoten mit je 16 CPUs)
- Spitzenleistung: 12.3 TFlop/s
- 8 TByte Hauptspeicher
- 145 TByte Festplattenplatz
- Zugriff ist beschränkt (classified)
- LINPACK: 7.2 TFlop/s
- Aktuelle Nr. 1 in Top500-Liste, ab Juni 2002 Nr. 2

Supercomputing heute: Aktuelle Hardware

❑ Earth Simulator

- 5120 Vektor-CPUs (640 Knoten mit je 8 CPUs)
- Spitzenleistung: 41 TFlop/s
- LINPACK: 35 TFlop/s
  - Schneller als die ersten 20 Rechner der Top500 zusammen, neue Nummer 1!
- Vorgesehen für Simulation von Klima- und Umweltmodellen



Supercomputing heute: Aktuelle Hardware

❑ Hitachi SR8000-F1/168 am Leibniz-Rechenzentrum in München

- 1344 CPUs (168 Rechenknoten mit je 8 CPUs)
- 1376 GByte Hauptspeicher (8 GByte pro Knoten, 4 Knoten mit 16 GByte)
- 10 TByte Festplattenplatz
- Spitzenperformance: 168 x 12 GFlop/s = 2016 GFlop/s
- LINPACK: 1645 GFlop/s
  - ca. Platz 9 in nächster Top500

Supercomputing heute: Aktuelle Hardware

❑ Kepler-Cluster der Universität Tübingen

- 196 CPUs (98 Rechenknoten mit je 2 CPUs)
- 100 GByte Hauptspeicher
- Standard-PC-Hardware (Pentium III @ 650 MHz)
- Vernetzung durch Myrinet interconnect
- Spitzenleistung: 127 GFlops
- LINPACK: 96.2 GFlops
  - effizientester Cluster in Europa
  - Top500 Rang 495

Links und Literatur

- **HPC@RRZE:** `http://www.rrze.uni-erlangen.de/dienste/hpc/`
- **LRZ München:** `http://www.lrz.de/services/compute/hlr/`
- **HLRS Stuttgart:** `http://www.hlrs.de/`
- **NIC Jülich:** `http://www.kfa-juelich.de/nic/`
- **Top500:** `http://www.top500.org/`
- **Earth Simulator:** `http://www.gaia.jaeri.go.jp/`
- **MPI:** `http://www.mpi-forum.org/`
- **OpenMP:** `http://www.openmp.org/`
- **W. Schönauer: "Scientific Supercomputing"**
  - `http://www.uni-karlsruhe.de/Uni/RZ/Personen/rz03/book/`
- **W. Press et al: "Numerical Recipes in C"** (Fortran…)
  - **Cambridge University Press**

Hochleistungssysteme am RRZE

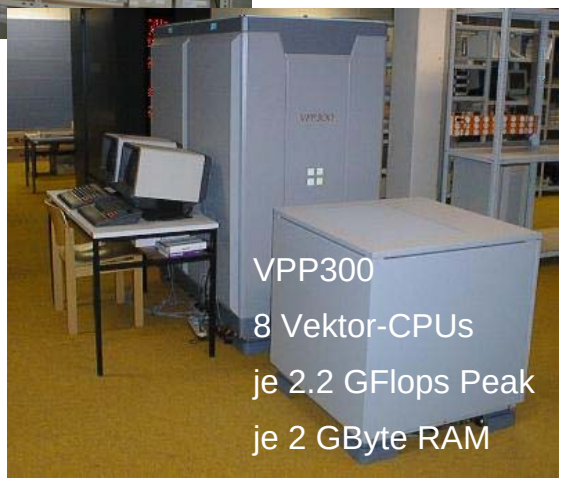
Workstation-Cluster

17 Rechner, je 128 MByte RAM

Memory-Server

28 CPUs, je 1 GFlop Peak

56 GByte gemeinsamer Hauptspeicher



VPP300

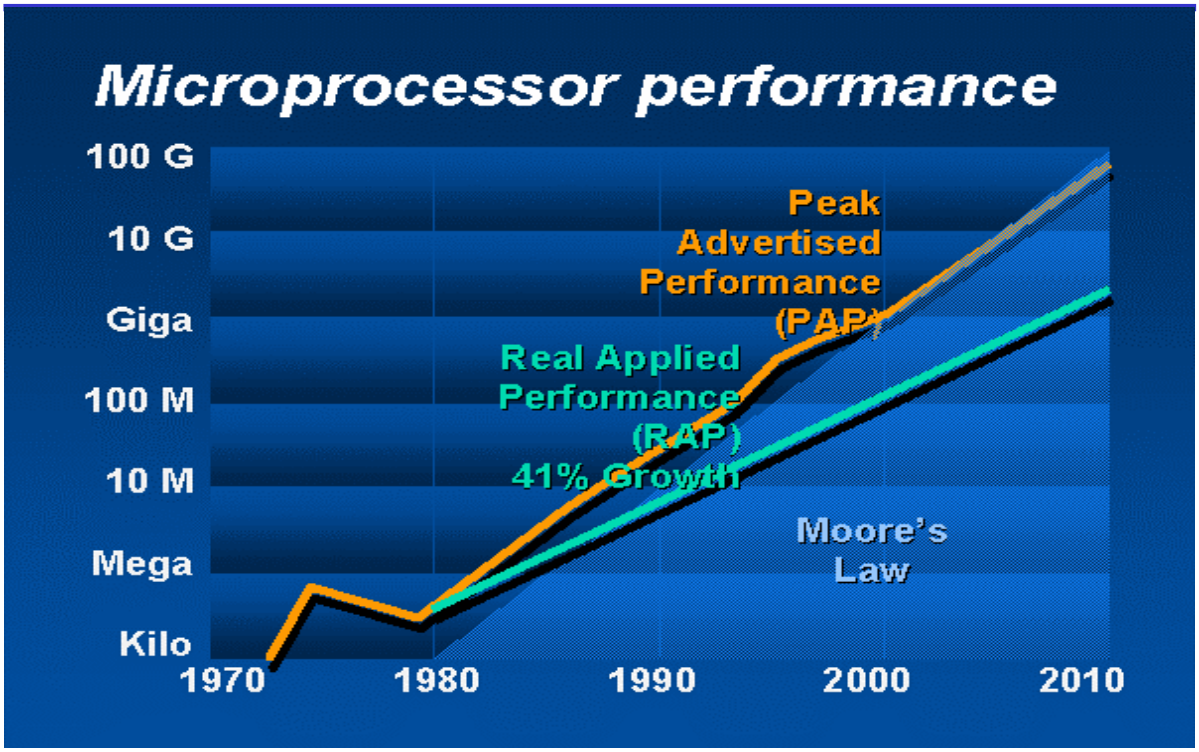
8 Vektor-CPUs

je 2.2 GFlops Peak

je 2 GByte RAM

Vielen Dank!

Motivation:
Schere zwischen Spitzen- und Realleistung



Motivation:
Möglichkeiten zur Leistungssteigerung

Leistungssteigerung : Hardware, Software *und* Algorithmen

- **Bessere Hardware**
  - **Prozessorgeschwindigkeit verdoppelt sich etwa alle 18 Monate**
  - **Aber: Speicherzugriff wächst deutlich langsamer!**
- **Bessere Software**
  - **Bessere Compiler**
  - **Bessere Bibliotheken**
- **Bessere Algorithmen**
  - **Fourier-Transformation → Fast-Fourier-Transformation**
  - **Gauß-Seidel → Multigrid-Verfahren**