→Grundbegriffe der Parallelität/Parallelverarbeitung
→historische und aktuelle Aspekte der Parallelverarbeitung

1.1 Was ist Parallelität?

- griechisch: *pará* = „entlang“, „neben“; *allélon* = „einander“
- mehrere räumlich und/oder zeitlich nebenher verlaufende Aktionen
- ggf. auch unabhängig voneinander ablaufend

Parallele Programmausführung:
- zum selben Zeitpunkt wird mehr als 1 Befehl ausgeführt
- und/oder mehr als eine Dateneinheit verarbeitet

(bisherige Ausbildung [Betriebsysteme u.a.] fokussierte oft nur auf Quasi-Parallelität ...)

1.2 Warum Parallelverarbeitung? (I)

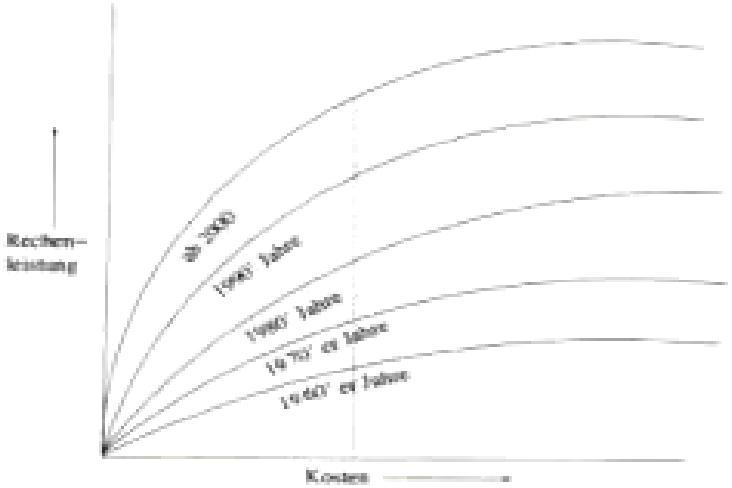
- vor allem: schnellere Lösung eines Problems erwünscht (Ziel: Geschwindigkeitsgewinn)
- Lösung eines sehr großen Problems erwünscht: extremer Rechenaufwand → High Performance Computing
- hohe Ausfallsicherheit/hohe Verfügbarkeit erwünscht: → High Availability Computing durch Redundanz
- große Datenmengen liegen nur verteilt vor bzw. können nicht zentral gespeichert werden (Bsp. SETI@home,...)

1.2 Warum Parallelverarbeitung? (II)

- Lichtgeschwindigkeit ist obere Schranke für Verarbeitungs-geschwindigkeit, daher Erhöhung der Leistung bei sequentieller Verarbeitung begrenzt

Bsp.: Annahme einer max. Leitungslänge von 1 cm
Lichtgeschwindigkeit 3 x 10^{10} cm/s
→ Chip könnte max. mit 3 x 10^{10} = 30 GHz getaktet werden

Vergleich: aktuelle Taktraten: ca. 2-3 GHz



Steigerungsraten in der sequ. Verarbeitungsgeschwindigkeit wie in der Vergangenheit sind künftig nicht mehr möglich!

Bild: Kosten-/Leistungsrate sequentieller Rechner
Quelle: Speckenmeyer. VO Parallele Algorithmen, 07/08

1.2 Warum Parallelverarbeitung? (III)

Bsp. Wettervorhersage (Quelle: Bauke/Mertens: Clustercomputing. Springer 2006)

Modell der Atmosphäre als 3D-Gitter
Berechnung von Temperatur, Druck, Windgeschwindigkeit an jedem Punkt
Aktualisierung an jedem Punkt in Abhängigkeit von den Daten an den Nachbarpunkten
Problem: möglichst großes Gebiet erfassen, aber kleine Maschenweite
→Große Zahl an Gitterpunkten!

Grob vereinfachendes Rechenbeispiel:
- Maschenweite 1 km, Erdatmosphäre bis 20 km Höhe: 10^{10} Gitterpunkte
- und Aktualisierung alle 10 Sekunden
→ Für 3 Tagesvorhersage: 26 000malige Aktualisierung aller Punkte nötig!
(Annahme: Aktualisierung eines Punktes: 100 Operationen nötig)
= 2,6 x 10^{16} Operationen für die gesamte Simulation
bei PC mit 10^9 Ops/s: 300 Tage Rechenzeit! (= 100 mal zu langsam!)
besser wäre: Rechner mit 10^{12} Ops/s: knapp 8 Stunden Rechenzeit

1.2 Warum Parallelverarbeitung? (IV)

Bsp. Wettervorhersage (Forts.) (Quelle: Bauke/Mertens: Clustercomputing.)

besser wäre: Rechner mit 10^{12} Ops/s: knapp 8 Stunden Rechenzeit

Aber: benötigte Daten stehen nicht schnell genug zur Verfügung!

Annahme: 1 Datenzugriff zum Speicher pro Operation
dann müßte ein Datum in max. 10^{-12} s beschafft werden
in 10^{-12} s kommt selbst Licht nur 0,3 mm weit
zur Laufzeitminimierung müßten alle Daten in einem Kreis mit 0,3 mm Radius um den Prozessor gespeichert sein
Speicherbedarf/Gitterpunkt: ca. 20 Zahlen von je 32 Bit
Summe für alle 10^{10} Punkte: 6,4 x 10^{12} Bit
d.h. pro Bit max. eine Kreisfläche mit Radius von 10^{-10} m verfügbar
entspricht Größe eines Atoms!!
Speicherdichte von 1 Bit/Atom unmöglich ...

... läßt sich nur durch parallele Verarbeitung annähernd lösen

1.2 Warum Parallelverarbeitung? (V)

Bsp. Wettervorhersage (Forts.) (Quelle: Bauke/Mertens: Clustercomputing.)

... läßt sich nur durch parallele Verarbeitung annähernd lösen:

Rechner mit 10^{12} Ops/s existieren, sind allerdings massiv parallel!
„Wenn die Daten nicht schnell genug zu einem Prozessor kommen, dann müssen eben viele Prozessoren zu den Daten gehen“

Erneute grobe Berechnung:
1000 Prozessoren mit nur 10^9 Ops/s (= Geschwindigkeit eines PCs)
Jeder Prozessor berechnet nur 1/1000 der Daten (=10^7 Gitterpunkte)
Weg. langsamerem Prozessor könnten Daten nun bis 300 mm entfernt liegen
Weg. Datenaufteilung geringere Speicherdichte nötig: 6,4 x 10^9 Bit = 800 MB
Ein akt. PC hat (mind.) soviel Speicher
→1000 PCs könnten das Problem lösen...

1.2 Warum Parallelverarbeitung? (VI)

Weitere Gründe (I):

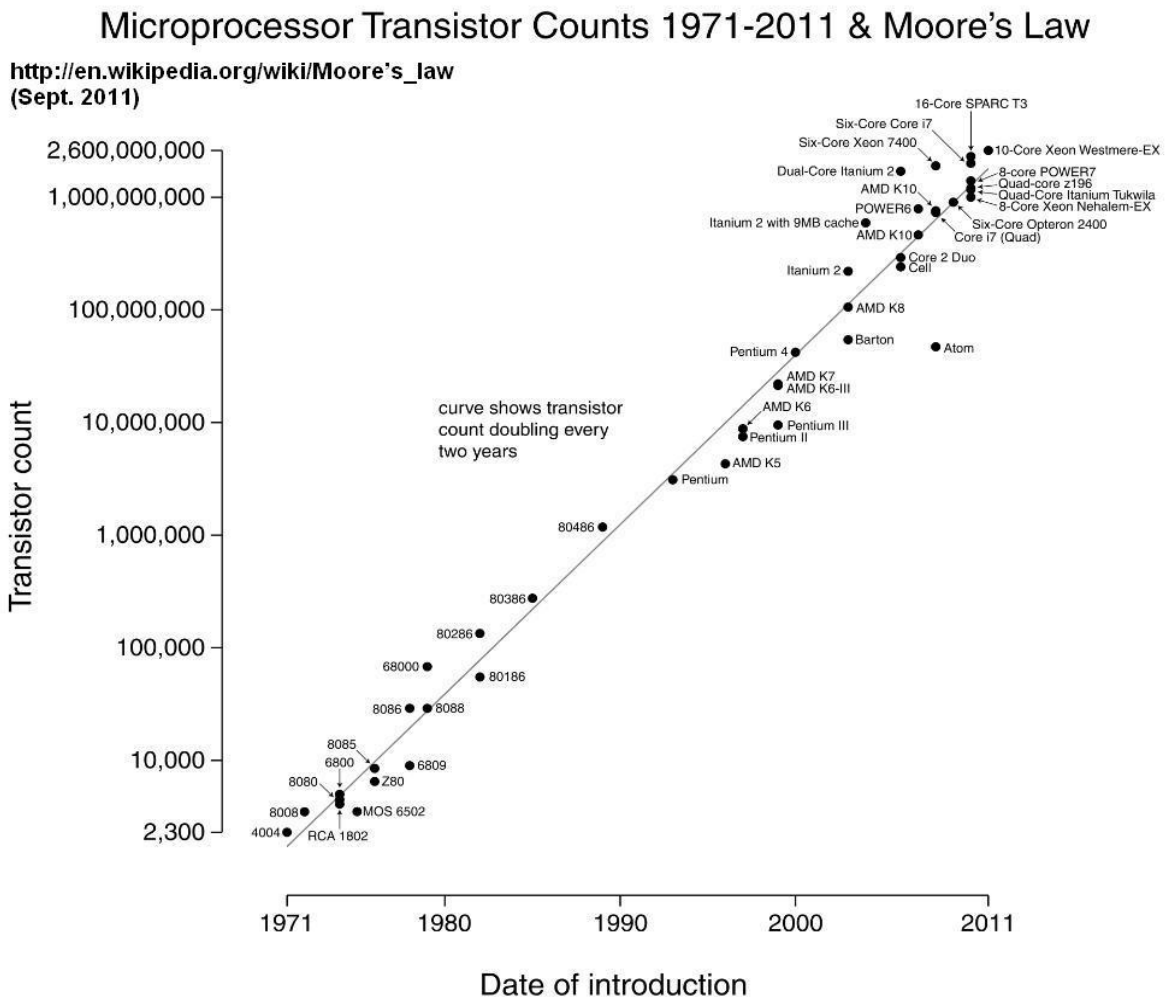
Moore's Law *(1965):* Komplexität integrierter Schaltkreise (= Anzahl der Schaltkreiskomponenten auf einem Computerchip) verdoppelt sich etwa alle zwei Jahre.

Heute abgewandelte Auslegung: Anzahl an Transistoren auf einem handelsüblichen Prozessor verdoppelt sich etwa alle 1,5 Jahre.

2007 sagte Moore das Ende seines „Gesetzes“ für etwa 2020 voraus.
2008 prognostizierte P. Gelsinger (Chef der Digital-Enterprise-Sparte von Intel) eine längere Gültigkeit des Moore‘schen Gesetzes bis etwa 2029.
Also: Ende der Leistungssteigerung bei Prozessoren „in Sicht“

Und: Erhöhung der Speichergeschwindigkeit bleibt deutlich zurück!!

1.2 Warum Parallelverarbeitung? (VII)



1.2 Warum Parallelverarbeitung? (VIII)

Weitere Gründe (II):

Sogar Einzelprozessoren können heute parallel rechnen (Multicore-CPUs)!

Billige Standardprozessoren können teure Spezialarchitekturen ersetzen
→ Wirtschaftlicher Antrieb für Parallelverarbeitung
(siehe Cluster!)

Häufig liegt Parallelität schon im Problem (inhärent) vor
(sie wurde leider bisher zumeist durch lineare Programmierung „versehentlich beseitigt“, und musste anschließend aufwändig „ wiederhergestellt“ werden ...)

1.3 Historie (I)

- Entwicklung der Rechnerarchitekturen ist schon länger mit Aspekten der Parallelität verbunden
- Bsp.: selbständige E/A-Prozessoren, symmetrische Multiprozessorsysteme
- Typische Vertreter:
  - 60er Jahre: CDC 6600 Control Data Corp.
    mit 10 Funktionseinheiten für verschiedene Operationen in der CPU
    + 10 E/A-Prozessoren
  - 70er Jahre: erster Vektorrechner: Cray-1 (1976)
    mehrere gleiche Ops über Vektordaten
    10 Pipelinestufen für Gleitkomma-Ops

    erste massiv parallele Systeme (SIMD)
    ILLIAC (1972, 64 Knoten), ICL DAP (1977)
    CM-2 (Thinking Machines, 1985, einige Tausend Knoten)

1.3 Historie (II)

- 80/90er Jahre: Verbreitung von RISC-CPUs, → MIMD
  gemeinsamer Speicher für alle CPUs
  (bei großer Anzahl problematisch…):
  C.mmp (16 PDP/11-40 Prozessoren)
  RP3 (IBM, 1985, bis 512 CPUs)
  KSR-1 (Kendall Square Research, 1992)

  oder ohne gemeins. Speicher: message passing
  Intel iPSC
  nCUBE Hypercube
  Suprenum (GMD Deutschland)

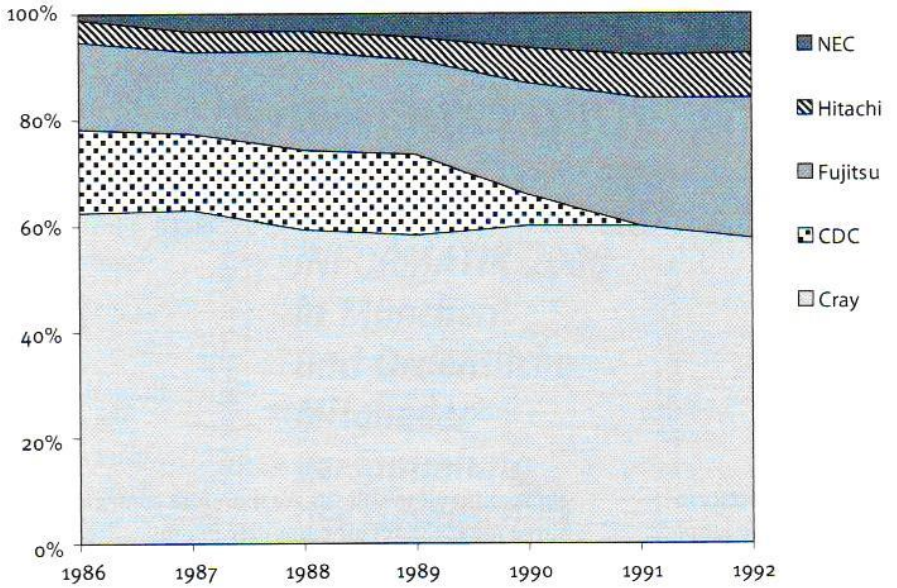
  Zwischenstellung: CRAY T3D (1994)

- 90/2000er Jahre: viele Cluster aus Standard-CPUs/PCs, + GPUs,
  Grid Computing, Cloud Computing,...

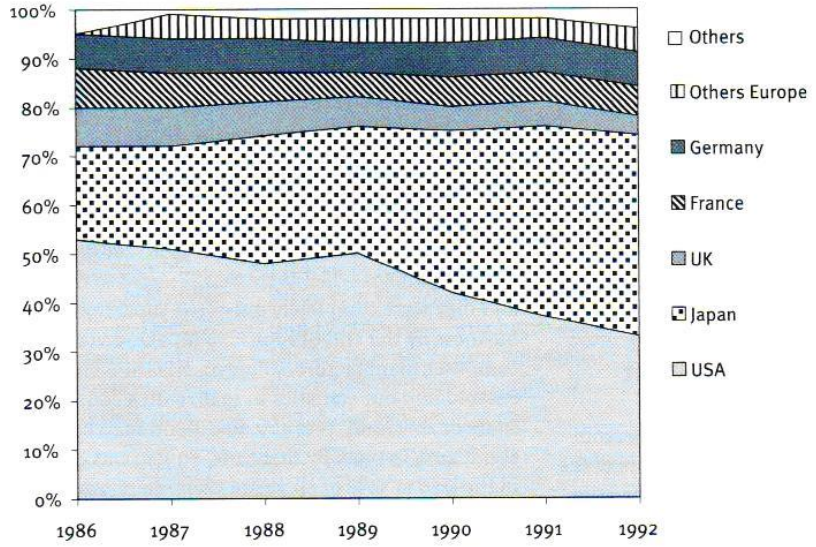
- etwa ab 2010: GPU-Computing

1.3 Historie (III)

- 1986-1992: „Mannheim Supercomputer Statistics“
  - Prof. Meuer, Uni Mannheim
  - Liste der schnellsten Rechner
  - anfänglich vor allem Vektorrechner
  - keine klaren Regeln zur Auflistung/Bewertung (v.a. b. neueren Syst.)



Verteilung der Hersteller



Verteilung nach Ländereinsatz

Quelle: Meuer: The TOP500 Project. in: Informatik Spektrum Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (IV)

- ab 1993: „TOP 500“
  - H. Meuer († ehem. Uni Mannheim)
    + E. Strohmaier (Uni Mannheim, Lawrence Berkeley Nat. Lab.)
    + H. Simon (Lawrence Berkeley Nat. Lab.)
    + J. Dongarra, (Uni Tennessee, „Vater“ des Linpack-Benchmarks)
  - Liste der 500 leistungsstärksten Rechner
  - Benchmark: R_{max} (= beste Linpack Leistung)
    - dieser Benchmark ist populär und leicht verfügbar,
    - allerdings steht R_{max} eben nur für die Fähigkeit, ein großes lineares Gleichungssystem zu lösen …
  - alle Bewertungsdaten öffentlich zugänglich
  - 45. TOP500-Liste: Juni 2015

→ TOP500-Liste ist unter Herstellern, Nutzern und in den Fachmedien (immer noch) *DAS* Instrument zur Analyse des Marktes für High Performance Computing (HPC)

1.3 Historie (V)

- TOP500: Wettbewerb zwischen Einsatzländern (I)
  - 1992: Kopf-an-Kopf-Rennen zw. USA und Japan erwartet, aber praktisch: 45% USA zu 22% Japan in der ersten TOP500-List
  - Vergleich zur 30. Liste (2007): Anteil der USA noch größer!

1st TOP500 List, 06/1993

Country	Count	Share
USA	225	45.0%
Japan	111	22.2%
Germany	59	11.8%
France	26	5.2%
U.K.	25	5.0%
Australia	9	1.8%
Italy	6	1.2%
Netherlands	6	1.2%
Switzerland	4	0.8%
Canada	3	0.6%
Denmark	3	0.6%
Korea	3	0.6%
Others	20	4.0%
Total	**500**	**100.0%**

30th TOP500 List, 11/2007

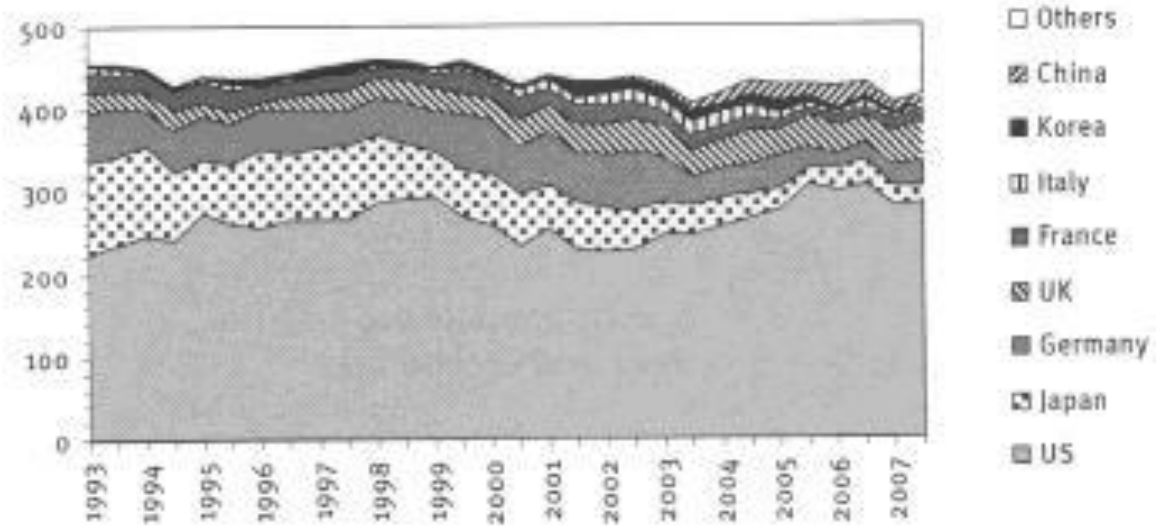
Country	Count	Share
USA	283	56.6%
Japan	20	4.0%
Germany	31	6.2%
France	17	3.4%
U.K.	48	9.6%
Australia	1	0.2%
Italy	6	1.2%
Netherlands	6	1.2%
Switzerland	7	1.4%
Canada	5	1.0%
Denmark	1	0.2%
Korea	1	0.2%
China	10	2.0%
India	9	1.8%
Others	55	11.0%
Total	**500**	**100.0%**

Vergleich bzgl. Einsatzländer in der 1. und 30. TOP500-Liste

Quelle: Meuer: The TOP500 Project. in: Informatik Spektrum Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (VI)

- TOP500: Wettbewerb zwischen Einsatzländern (II)



Vergleich bzgl. Einsatzländer von HPC-Systemen

Quelle: Meuer: The TOP500 Project. in: Informatik Spektrum Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (VII)

- TOP500: Wettbewerb zwischen Herstellern (I)
  - in 1. Liste 1993: Cray Research klar dominant, IBM und HP gar nicht vertreten
  - in der 30. Liste 2007: IBM führt vor HP; Cray weit abgeschlagen

1st TOP500 List, 06/1993

Manufacturer	Count	Share
Cray Research	205	41.0%
Fujitsu	69	13.8%
Thinking Machines	54	10.8%
Intel	44	8.8%
Convex	36	7.2%
NEC	32	6.4%
Kendall Square Res.	21	4.2%
MasPar	18	3.6%
Meiko	9	1.8%
Hitachi	6	1.2%
Parsytec	3	0.6%
nCube	3	0.6%
Total	**500**	**100.0%**

30th TOP500 List, 11/2007

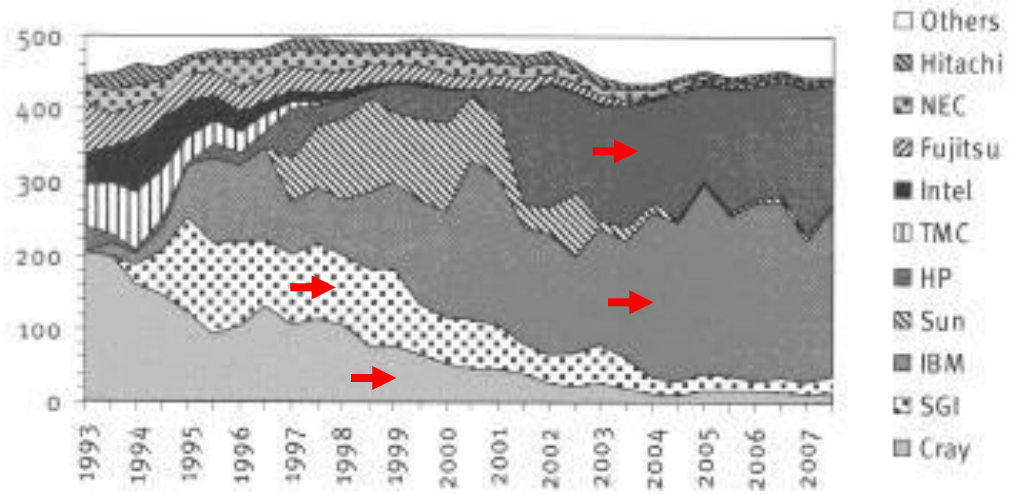
Manufacturer	Count	Share
Cray Inc.	14	2.8%
Fujitsu	3	0.6%
Thinking Machines	–	–
Intel	1	0.2%
Hewlett-Packard	166	33.2%
NEC	2	0.4%
Kendall Square Res.	–	–
MasPar	–	–
Meiko	–	–
Hitachi/Fujitsu	1	0.2%
Parsytec	–	–
nCube	–	–
IBM	232	46.4%
SGI	22	4.4%
Dell	24	4.8%
Others	35	7.0%
Total	**500**	**100.0%**

Vergleich bzgl. Hersteller in der 1. und 30. TOP500-Liste

Quelle: Meuer: The TOP500 Project. in: Informatik Spektrum Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (VIII)

- TOP500: Wettbewerb zwischen Herstellern (II)
  - HPC-Markt sehr dynamisch!!
  - in nur 15 Jahren komplette Änderung:
    - Cray vom Marktführer zum Nischenanbieter
    - IBM: Anfang der 90er Jahre ohne HPC-Bedeutung, heute Marktführer in allen HPC-Anwendungsbereichen
    - HP (anfänglich nur über Convex in der Liste), ist stabile Nr. 2
    - Sun, ehemals gut positioniert, danach stark zurückgefallen
    - gegenwärtig: „Aufholjagd“ von Sun, Cray, u.a. mit starken Hybrid-Systemen sogar in den Top10!



Vergleich bzgl. Hersteller von HPC-Systemen

Quelle: Meuer: The TOP500 Project.
in: Informatik Spektrum
Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (IX)

- TOP500: Vergleich bzgl. Anwendern/Einrichtungen
  - USA haben die 4 stärksten HPC-Einrichtungen und dominieren auch die Tabelle der 20 stärksten Systeme über alle 30 TOP500-Listen mit etwa 2/3 Anteil, Anteil Japan: ca. 20%

Rank	Site	Country	Over time
1	Lawrence Livermore National Laboratory	USA	5.39%
2	Sandia National Laboratories	USA	3.70%
3	Los Alamos National Laboratory	USA	3.41%
4	Government	USA	3.34%
5	The Earth Simulator Center	Japan	1.99%
6	National Aerospace Laboratory of Japan	Japan	1.70%
7	Oak Ridge National Laboratory	USA	1.39%
8	NCSA	USA	1.31%
9	NASA/Ames Research Center/NAS	USA	1.25%
10	University of Tokyo	Japan	1.21%
11	NERSC/LBNL	USA	1.19%
12	Pittsburgh Supercomputing Center	USA	1.15%
13	Semiconductor Company (C)	USA	1.11%
14	Naval Oceanographic Office (NAVOCEANO)	USA	1.08%
15	ECMWF	U.K.	1.02%
16	ERDC MSRC	USA	0.91%
17	IBM Thomas J. Watson Research Center	USA	0.86%
18	Forschungszentrum Jülich (FZJ)	Germany	0.84%
19	Japan Atomic Energy Research Institute	Japan	0.83%
20	Minnesota Supercomputer Center	USA	0.74%

TOP20 der HPC-Anwender über alle 30 TOP500-Listen

Quelle: Meuer: The TOP500 Project. in: Informatik Spektrum Jg. 31, H. 3/2008, S. 203-222

1.3 Historie (X)

drei „Einzelfall-Beispiele“ (I)

1. „Deep Blue“:

- = Rang 259 in der 9. TOP500-Liste 1997
- R_{max} = 11,37 GFlops
- Hersteller: IBM, SP2 P2SC
  mit 32 Prozessoren mit je 120 MHz (!)
- allerdings hatte jeder Prozessor
  15 spez. VLSI Schach-Chips
- Anwender: IBM Watson Res. Center Yorktown Hights/USA
- erster Schachcomputer, der den akt. Schachweltmeister (Kasparow) bezwang
- (2006 gewann auch „Deep Fritz“ gegen Weltmeister Kramnik…)

Bildquelle: http://www.tvfilmprops.co.uk/det/442/IBM-RS/6000-SP2-Deep-Blue-Supercomputer/

1.3 Historie (XI)

drei „Einzelfall-Beispiele“ (II)

2. „ASCI Red“

Bildquelle: http://www.new-npac.org/projects

- = Rang 1 in der 9. TOP500-Liste 1997
- R_{max} = 1068 GFlops → erster Teraflops-Computer“,
- Hersteller: Intel
- 7264 Intel Pentium Pro CPUs mit 200MHz, später aufgerüstet auf 9632 Intel Pentium II Overdrive CPUs mit 333MHz, gut skalierbar
- 104 Schränke, auf 230 m^2
- letzter allein von Intel produzierter Supercomputer
- Anwender: Sandia National Labs in Albuquerque/USA, Department of Energy; Simulation von Nukleartests ohne reale Versuche
  (zur Erinnerung: Frankreich zündete auf dem Mururoa-Atoll von 1966-1996 41 atmosphärische und 147 unterirdische Kernwaffen)
- Beginn einer neuen Supercomputer-Ära: legte Grundstein für US-Dominanz in der „Nach-Vektorrechner-Zeit“
- war bis zu seinem Ersatz (2005) 17 mal in den TOP500

1.3 Historie (XII)

Bildquelle: http://www.flickr.com/photos/kielbryant/361553366/

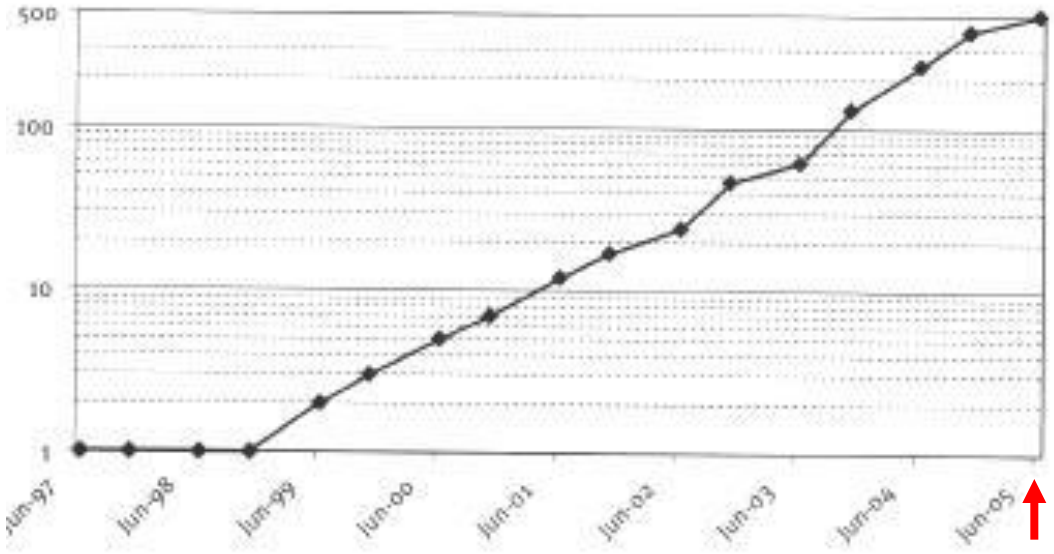
drei „Einzelfall-Beispiele“ (III)

3. „Earth Simulator“

- = Rang 1 in der TOP500-Liste
  !von 2002 bis 2004!
- R_{max} = ca. 85 TFlops
- Hersteller: NEC
- (hybride) SX-6 Architektur mit 640 Rechnerknoten
  - mit je 8 Vektorprozessoren und 16GByte Speicher
  - + 130 Kommunikationsknoten
- in einem 3200 m^2 großen Gebäude
- Ende 2008 stillgelegt und durch gleichnamigen Nachfolger auf Basis SX-9 ersetzt, in Top500 (Juni 2009) mit 122 TFlops auf Pl. 22

1.3 Historie (XIII)

- Erster „Megaflops-Rechner“: CDC 7600, 1971
  (Vorläufer der Vektorrechner)
- Erster „Gigaflops-Rechner“: Cray-2, 1986
- Erster „Terafops-Rechner“: ASCI Red, 1997



Anzahl von
„Teraflops-Rechnern“
in den TOP500

Quelle: Meuer:
The TOP500 Project.
in: Informatik Spektrum
Jg. 31, H. 3/2008, S. 203-222

Seit Juni 2005 sind in den TOP500 nur noch „Teraflops-Rechner“ enthalten!

1.3 Historie (XIV)

- Erster „Petaflops-Rechner“: IBM Blue Gene/P „Roadrunner“, 2008

  - insgesamt 122.400 Prozessoren:
  - vor allem AMD-Opteron-CPUs
  - sowie 12.240 XCell-8i-Prozessoren von IBM (→ Playstation 3!!)
  - via Infiniband-Interconnect miteinander verknüpft

- Erster „Exaflops-Rechner“: ???

- usw…

1.4 Anwendungsgebiete der Parallelverarbeitung (I)

- „**Grand Challenges**“

Begriff stammt aus den 80er Jahren (= Ziele im US-HPC-Programm als Antwort auf Japans „5th Generation Project“:

"A grand challenge is a fundamental problem in science or engineering, with broad applications, whose solution would be enabled by the application of high performance computing resources that could become available in the near future.“

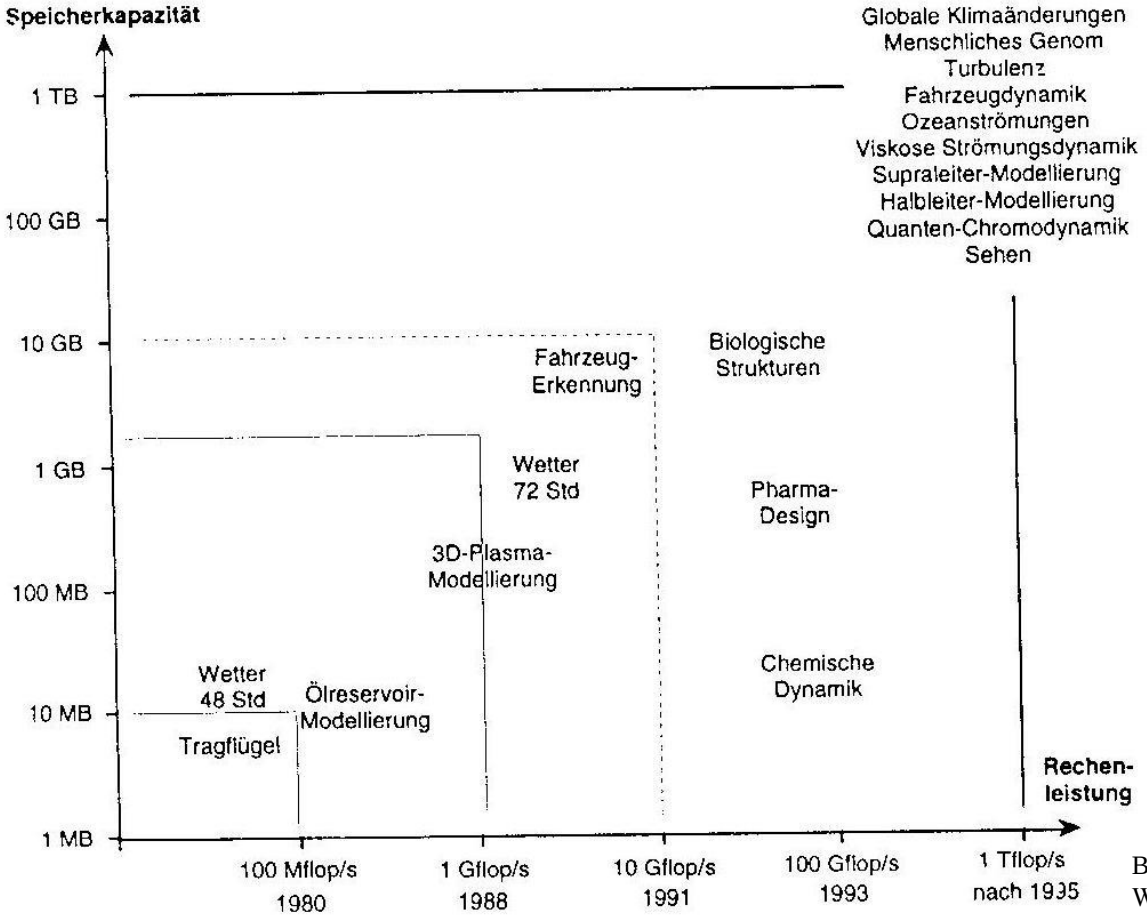
Quelle: A Research and Development Strategy for High Performance Computing", Executive Office of the President, Office of Science and Technology Policy, November 20, 1987

Damals wurde dort genannt:

- Computational fluid dynamics
- Electronic structure calculations for the design of new materials
- Plasma dynamics for fusion energy technology and for safe and efficient military technology;
- Calculations to understand the fundamental nature of matter
- Symbolic computations

1.4 Anwendungsgebiete der Parallelverarbeitung (II)

- „Grand Challenges“ (Abbildung ist relativ alt! Vgl. akt. Stand!!)



Noch eine Definition:

„A grand challenge problem is one that cannot be solved in a reasonable amount of time with today's computers.“ (Wikipedia)

Bild-Quelle:
Waldschmidt: „Parallelrechner“. Teubner 1995

1.4 Anwendungsgebiete der Parallelverarbeitung (III)

- weitere aktuelle Anwendungsgebiete (bzw. Herausforderungen):
  - Vorhersage physikal. Phänomene, v.a. Atomphysik, Halbleitertechnik,usw.
  - Vorhersage/Modellierung/Simulation bzgl. Natürlicher Vorgänge (Wetter/Umwelt, Erdöl/Erdgas-Exploration, allg. Biosysteme,…)
  - Risiko-Analysen, Kurs-Simulationen bei Banken,Versicherung ☹
  - techn. Simulationen (z.B. Motorsteuerung, Crash-Tests,…)
  - Bauteil-Optimierung, Festigkeitsanalysen
  - Proteinfaltung u.ä. Verfahren der Bio-Wissenschaften
  - Suchmaschinen, Data Warehousing, Data Mining (v.a. Bilder..)
  - Customer Relationship Management
  - Berechnung von Filmen/Filmsequenzen, verteilte Online-Spiele
  - (natürl.) Spacherkennung/-verarbeitung
  - Primzahlenberechnung (→ Verschlüsselung!)
  - Sicherung von Hochverfügbarkeit

1.5 parallele vs. verteilte Programmierung (I)

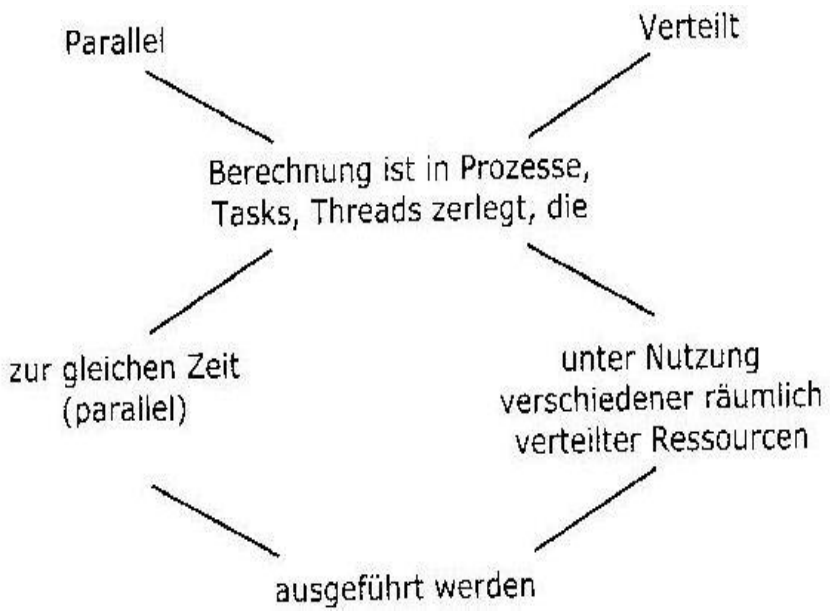
- Parallele Verarbeitung ([true] parallel processing):
  - Hauptziel: Geschwindigkeitsgewinn, also vor allem effizienzbetont
  - gleichzeitige Ausführung mehrerer Ablaufeinheiten innerhalb einer Anwendung
  - enge Kopplung zwischen Teilberechnungen typisch
  - kann (muss aber nicht) auf mehreren Prozessoren ausgeführt werden
  - traditionell eher homogene Architekturen, oft gemeinsamer Speicher

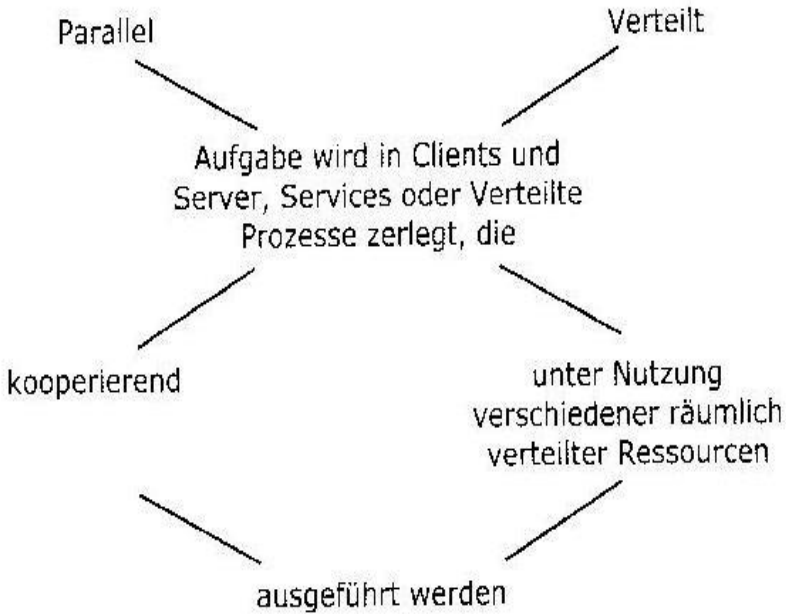
1.5 parallele vs. verteilte Programmierung (II)

- Verteilte Verarbeitung (distributed processing):
  - Ausführung verschiedener Teilaufgaben auf verschiedenen Rechnern
  - Hauptaspekt: Nutzung verschiedener, räumlich getrennter, aber über Netzwerk gekoppelter Ressourcen, z.B. WWW
  - dabei Wahrung von Transparenz (z.B. Ort, Art des Zugriffs, ...), Skalierbarkeit (z.B. Lastverteilung, geografische Entfernung) und Offenheit (dynamische Erweiterbarkeit um andere Subsysteme, vgl. Grid)
  - eher heterogene Umgebung, meist ohne gemeinsamen Speicher (Hardware, BS, Programmiersprache usw. können verschieden sein)
  - weitere Vorteile: gewisse Ausfalltoleranz, Fehlertoleranz
  - Höhere Verfügbarkeit durch Redundanz möglich

1.5 parallele vs. verteilte Programmierung (III)

- verschiedene Sichtweisen:





aus Sicht der parallelen Verarbeitung

aus Sicht der verteilten Verarbeitung

Quelle: Bengel u.a.: Masterkurs parallele und verteilte Systeme. Vieweg/Teubner, 2008

1.6 Ebenen der Parallelität (I)

(es gibt keine einheitl. Klassifikation)

- Ebene der Jobs/Benutzerprogramme

  (Job = in sich abgeschlossenes, i.a. aus mehreren Prozessen bestehendes Benutzerprogramm)

  → Realisierung z.B. durch Multi-user-Betrieb,
  vor allem zur Durchsatzsteigerung, weniger zum Geschw.-gewinn

- Ebene der Prozesse/Tasks/Threads

  → Multitasking/Multithreading, könnte „gewinnbringend“ umgesetzt werden durch Zuordnung auf parallele Verarbeitungseinheiten

- Ebene der Daten/Datenstrukturen

  → parallele Ausführung von (gleichartigen) Operationen über den Elementen geeigneter Datenstrukturen (z.B. Vektoren, Matrizen)

1.6 Ebenen der Parallelität (II)

- Ebene der Anweisungen/Schleifen) der Programmiersprache
  → Iterationen von Schleifen („Schleifendurchläufe“) werden parallel (oder zumindest überlappend) auf mehreren Prozessoren ausgeführt (je nach Datenabhängigkeit)

- Ebene der Maschinen-Befehle/Instruktionen
  → Ausnutzung mehrerer unabhängiger, paralleler Funktionseinheiten innerhalb des Prozessors (z.B. ALU, FPU, Branch,...)
  z.B. bei VLIW- und superskalaren Prozessoren

- Ebene der Befehlsphasen
  → z.B. Parallelität bzgl. Laden, Dekodieren, Ausführen: Pipelining

- Bitebene
  → Anzahl parallel verarbeitbarer Bits (Wortbreite: 8, 16,32,64.)

1.6 Ebenen der Parallelität (III)

- Bewertung:

  - Parallelitätsebenen überlappen sich zum Teil

  - Ausnutzung der Parallelität auf den einzelnen Ebenen mit unterschiedlichen Techniken, aber nicht immer im gleichen Maße möglich (z.B. „Behinderung von Parallelität“ auf Grund von Datenabhängigkeiten!)

1.6 Ebenen der Parallelität (IV)

- aus verschiedenen Sichten (vgl. auch Waldschmdt: Parallelrechner. Teubner 1995, Wismüller, Uni Siegen, VO Parallelverarbeitung, 2006)
  - Sicht des Anwendungs-Entwicklers (Designphase):
    - "natürlicher Parallelismus“
      - z.B. Berechnung der Kräfte für alle Sterne einer Galaxie
      - oft zu feinkörnig
    - Datenparallelität (Gebietsaufteilung)
      - Schritte des Algorithmus werden *in gleicher Weise* auf viele Daten angewendet, die sich *im selben Bearbeitungszustand* befinden
      - z.B. sequentielle Bearbeitung aller Sterne eines Raumgebiets
    - Funktionsparallelität, Taskparallelität (Aufgabenaufteilung)
      - *verschiedene* Schritte eines Algorithmus werden auf Teilmengen von Daten angewendet, die sich in *unterschiedlichen Bearbeitungszuständen* befinden
      - z.B. Vorverarbeitung, Berechnung, Nachbearbeitung, Visualisierung

1.6 Ebenen der Parallelität (V)

- Sicht des Programmierers:
  - Explizite Parallelität
    - Datenaustausch (Kommunikation/Synchronisation) muss selbst programmiert werden,
    - Konstrukte zur parallelen/verteilten Verarbeitung müssen (direkt oder als Erweiterung/Bibliothek [vgl. PVM, MPI]) in der Programmiersprache enthalten sein
  - Implizite Parallelität
    - für Anwendung/Programmierer nicht sichtbar
    - Umsetzung durch Compiler: er generiert parallelen Code, ggf. auch Code für die Kommunikation
      - direktivengesteuert oder automatisch
      - Schleifenebene / Anweisungsebene
    - innerhalb einer (nach außen hin sequentiellen) CPU
      - Superskalarität, Pipelining, ...
  - Mischformen

1.6 Ebenen der Parallelität (VI)

- Sicht des Systems (Rechner/Betriebssystem):
  - Programmebene (Jobebene)
    - unabhängige Programme
  - Prozessebene (Taskebene)
    - kooperierende Prozesse
    - meist mit explizitem Nachrichtenaustausch
  - Blockebene
    - leichtgewichtige Prozesse (Threads)
    - Kommunikation über gemeinsamen Speicher
    - oft durch Compiler erzeugt (z.B. Parallelisierung von Schleifen)

(Forts.)

1.6 Ebenen der Parallelität (VII)

(Forts.)

- Anweisungsebene (Befehlsebene)
  - elementare Anweisungen (in der Sprache nicht weiter zerlegbare Datenoperationen)
  - Scheduling automatisch durch Compiler und/oder zur Laufzeit durch Hardware
  - z.B. bei VLIW, superskalaren Prozessoren

- Suboperationsebene
  - elementare Anweisungen werden durch den Compiler oder in der Maschine in Suboperationen aufgebrochen, die parallel ausgeführt werden z.B. bei Vektor- oder Feldoperationen

1.6 Ebenen der Parallelität (VIII)

- andere (gröbere) Unterteilung: *Granularität*

  → bzgl. Verhältnis von Berechnung zu Synchr./Kommunikation, beeinflusst den Leistungsgewinn

  - grobkörnige (coarse grain) Parallelität zwischen relativ großen Teilen von Programmen (Job-Ebene, Prozess-Ebene)

  - (mittelkörnige Parallelität, Datenebene, evtl. Blockebene)

  - feinkörnige (fine-grain) Parallelität (Anweisungsebene bis Bitebene)
    hier sind die Bearbeitungselemente einfacher,
    man kennt den Ablauf und die Dauer der Operationen besser