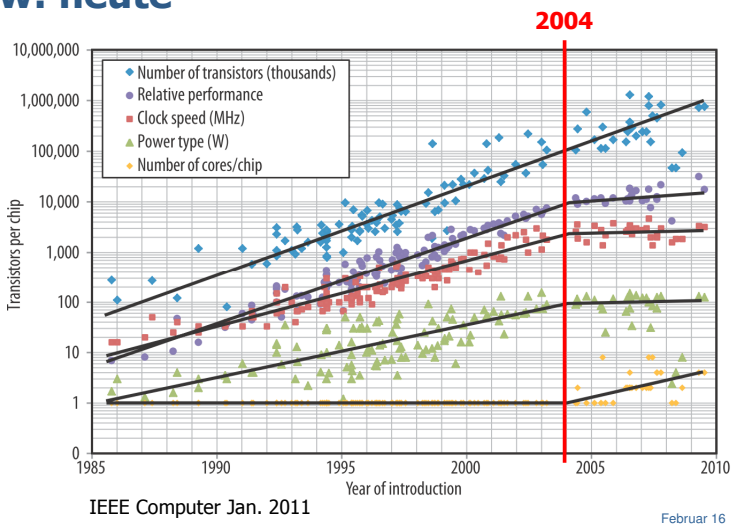
Multicore and Parallel Computing

intro

M. Thaler, TG208, tham@zhaw.ch
www.zhaw.ch/~tham

Was ist passiert ?

- **Moore's Law: 1965**
  - Verdoppelung der Anzahl Transistoren pro Chip alle 1.5 (2) Jahre
- **Moore's Law: heute**

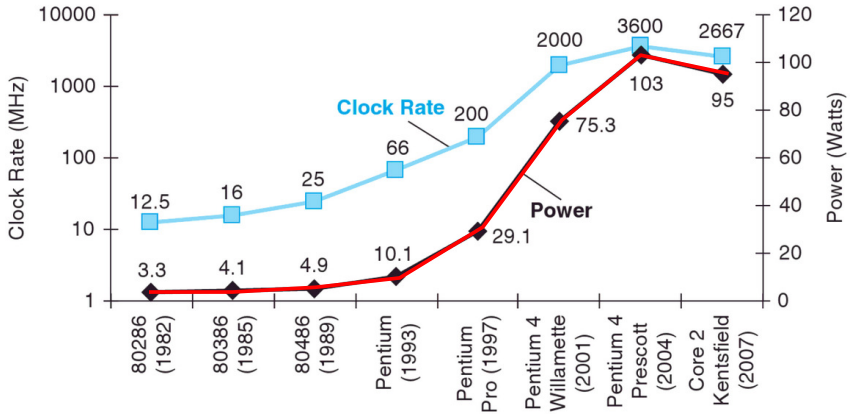


IEEE Computer Jan. 2011

- Das Moor'sche Gesetz beschreibt Skalierung resp. Entwicklung
  - der Anzahl Transistoren pro Chip
  - der Clockfrequenz
- Das Moor'sche Gesetz gilt immer noch, aber
  - seit 2004 jedoch statt Skalierung der Clockfrequenz
    → Skalierung der Anzahl Cores (Recheneinheiten) pro Chip
- zwei wesentliche Gründe
  - **Power Wall**: Energieverbrauch hat Grenze erreicht
    - Zuverlässigkeit der Chips gefährdet (Wärmeprobleme)
    - Energiedichte in Prozessoren bis ~100 Watt pro cm^2
      (20 Watt pro cm^2 entspricht in etwa einer Herdplatte)
    - Spannung lässt sich nicht weiter senken: Leckströme
  - architektonische Möglichkeiten weitgehend ausgeschöpft
    - pipelining
    - branch prediction
    - out of order execution
    - hyperthreading (<50% Rechenleistungsgewinn)
- Quellen
  - IEEE Computer, Jan. 2011
  - Kathy Yellick, Berkeley

Power Wall

- **Power and Clock**



Computer Organization and Design
D.Patterson, J. Hennessy

- P4 Prescott (Pentium 4)
  - sie size = 112 mm2
  - P = 103W
  - Leistungsdichte: 92 W / cm2
- Kochplatte
  - Leitsungsdichte: ~ 8W / cm2 (2kW, 18cm Durchmesser)

Wie geht's weiter?

- **Anzahl Transistoren pro Chip nimmt zu**
  - mehrere Cores pro Chip:
    - wie viele?
  - einfachere, aber sehr viele Cores
    - z.B. GPU's
  - **heterogene Cores**
    - unterschiedliche Funktionalität
    - unterschiedliche Taktfrequenzen
  - "Flat Energy Budget"
    - Verpackung (Wärme) und Mobile Computing (Batterien)
- **Weitere Informationen**
  - siehe ACM-Communications May 2011:
    "The Future of Microprocessors"

- Anzahl Cores pro Chip
  - Kommunikation benötigt Zeit und Energie
  - setzt Limite für Anzahl komplexer CPU's auf einem Chip

- Zur Verfügung stehende Energiemenge
  - limitiert durch Packaging
  - mobile Anwendungen (Batteriebetrieb)
  - steigende Anforderungen an Anwendungen (Rechenaufwand)

- Dark Silicon (http://darksilicon.org/)
  - das Moor'sche Gesetzt bewirkt, dass die nutzbare Chipfläche innerhalb des Leistungs-Budgets exponentiell sinkt → es gibt eine Utilization Wall: die nicht genutzte Chipfläche wird Dark Silicon genannt

Sicht des Programmierers

- **Bis 2004**
  - Performance kein (grosses) Problem:
    "die nächste Prozessorgeneration bringt genügend Leistung"
- **Seit 2004**
  - Wie nutze ich mehrere/viele Cores sinnvoll/effizient?
- **Zur Zeit**
  - keine allgemeinen Lösungen verfügbar
  - teilweise automatische Lösungen möglich
    - auch Hints (Direktiven) durch Programmierer
  - "Hauptlast" nach wie vor beim Programmierer
  - ... ist das schwierig?
- **Zukunft?**

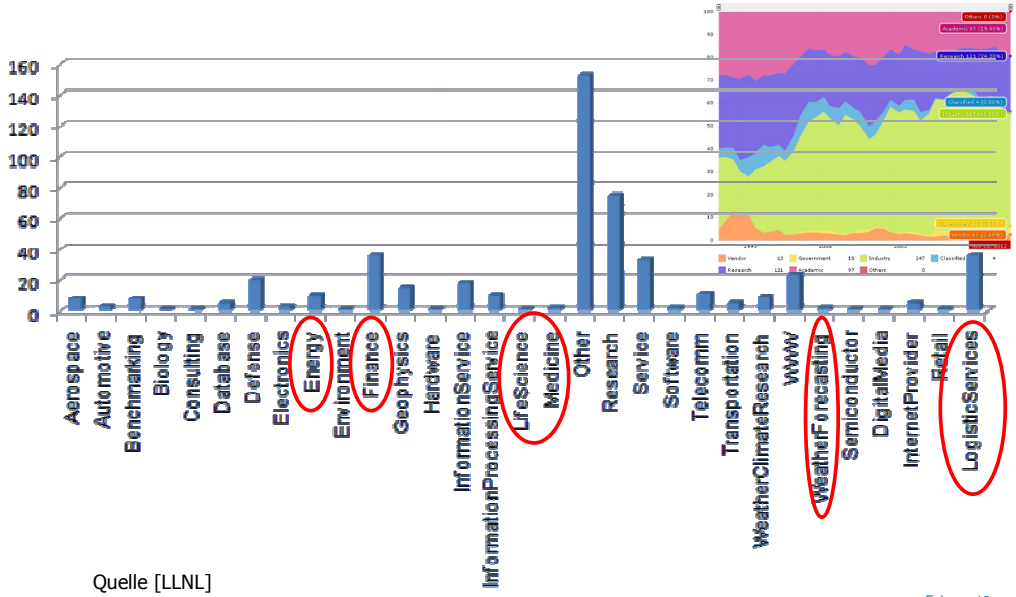
- Die Hardwarehersteller
  - haben das Performance Problem auf die Software abgeschoben

- Die Compilerentwickler
  - haben bis jetzt noch keine allgemein gültige Lösung gefunden
  - bessere Hardwarenutzung implementiert
  - alternative Sprachkonzepte (hoher Abstraktionsgrad) unterstützen teilweise Parallelverarbeitung
    - Aktoren, Funktionale Programmierung, etc.

- Zukunft
  - schwierig vorauszusagen
  - vieles ist im Wandel

Anwendungen

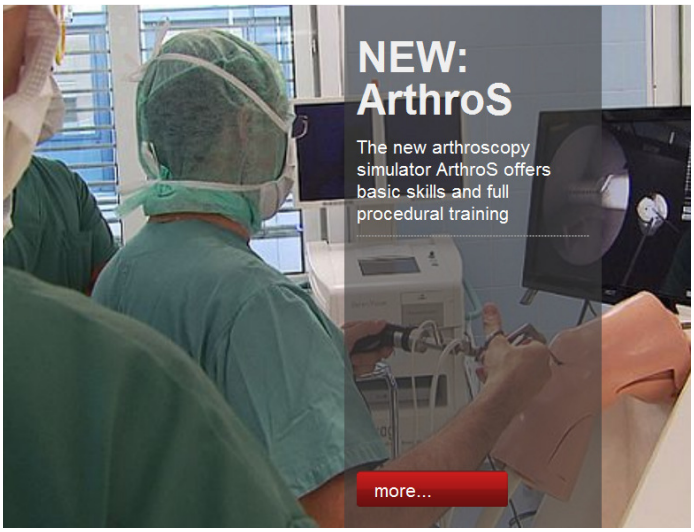
- **Anwendungen für High Performance Computing**



Quelle [LLNL]

... Anwendungen

- **Z.B. Trainingsimulator (div. NF und KTI Projekte)**
  - minimal invasive chirurgische Eingriffe
    - Multicore
    - GPU



www.virtamed.ch

- Erstes Projekt
  - Lasso 1996: Kooperation mehrerer Inst. an der ETH
  - Cluster von Workstation mit Myrinet
- CO-ME (ETH / ZHAW / Uni-Spital ZH)
  - Phase 1: 2001 - 2005
    - Silicon Graphics (ETHZ)
  - Phase 2: 2005 - 2009 -> Spin Off "Virtamed"
    - single PC und GPU
- Arthro (ETH / ZHAW / Balgrist und Virtamed)
  - KTI-Projekt 2009 - 2011
- INTERSIM (ZHAW / Virtamed)
  - KTI-Projekt 2012 - 2013

Ziele: Parallel Computing

- **Motivation**
  - (fast) alle modernen Prozessoren sind parallel
    - Nutzung moderner Mehrprozessor / Multicore Technologie
    - zunehmend
      - mobile Systeme
      - heterogene Systeme

- **3 Ziele**
  - kürzere Rechenzeit, gleiche Problemgrösse
  - gleiche Rechenzeit, grösseres Problem
  - Reduktion Energieverbrauch

- Parallel Computing
  - dazu gehören
    - Analyse: SW und HW Aspekte
    - Design
    - Implementierung
    - Testing und Verifikation

- Parallel Programming
  - Implementierung
  - Umsetzung auf einer spezifischen Plattform

Was werden wir tun?

- **Multicore-Prozessoren & GPU's**
  - wichtigste Hardware-Modelle ... im Moment
    - Shared Memory Modell (gemeinsamer Speicher)
    - Single Instruktion Multiple Data (datenparallele Verarbeitung)
  - Problemstellungen in Verteilten Systemen (Cloud) → sehr ähnlich
- **Patterns für Parallel Programming**
  - gute Grundlage für Verständnis der Parallel-Programmierung
  - unabhängig von Programmiersprache und Hardware
  - ausgewählte Beispiele/Anwendungen (Fallstudien)
- **Umsetzung (Praktika) mit**
  - PThreads Programmierung mit Threads (Einfluss HW, first steps)
  - OpenMP API für Shared Memory Programmierung (Direktiven)
  - OpenCL API für Programmierung von GPU's (SIMD)

MPC ZHAW, MPC FS16, M. Thaler

Februar 16 9

- Multicore-Prozessoren und GPU's
  - alle neueren PC's gehören dazu (Intel Prozessoren)
    - mehrere Cores (2-8) und eine GPU
  - vermehrt auch in Mobilen Geräten
  - die zwei wichtigsten Hardware-Modelle werden unterstützt
  - viele weitere Multicore-Prozessoren ähnlich aufgebaut
    - z.B. Signalprozessoren und Embedded Prozessoren
- Patterns für Parallelprogrammierung
  - Muster gute Grundlage zum Verständnis der Parallelprogrammierung
  - nur wenige Muster notwendig
- Umsetzung
  - häufig eingesetzte, standardisierte und frei verfügbare Programmierumgebungen, C- basiert
  - explizit kein Java
    - möglichst nahe an HW (Einfluss der JVM und JIT)
    - deckt nur (kleinen) Teil der Möglichkeiten ab (threading)
    - späteres Einarbeitung in concurrent package für Informatiker einfach
    - es geht nicht um Programmiersprache

Inhalt

- Begriffe und Grundlagen
- OpenMP
- Testing / Profiling
- Task- and Datenflussgraphen
- Parallel Programming und Patterns
- Parallele Hardware
- OpenCL
- Shared Objects
- Advanced Topics

Lehrziele

- **Sie kennen**
  - die wichtigsten aktuellen parallelen HW-Architekturen und können ihre Anwendung diskutieren
  - die wichtigsten Methoden zur Parallelisierung von Anwendungen und Algorithmen und können sie bezügliche Einsatzmöglichkeiten diskutieren und bewerten.
- **Sie können**
  - Software für parallele Systeme konzipieren, implementieren und testen, sie kennen dabei die wichtigsten Fallstricke und können Sie vermeiden
  - sich schnell und effizient in neue Problemstellungen im Zusammenhang mit parallelen Prozessoren und Anwendungen einarbeiten

Anmerkungen zum Kurs

- **Kurs → Auswahl von Themen**
  - **Einführung** in Parallel Computing/Programming
    - wichtigste Aspekte
    - guter, fundierter Überblick (Breite)
  - Parallel Computing/Programming
    - umfangreiches Gebiet
    - im Wandel
  - ev. Anpassungen während Kurs
    - Ablauf
      - viele **Huhn - Ei** Probleme
    - roter Faden
      - teilweise schwierig: Auswahl von Themen
      - Praktika beeinflussen Ablauf

... Anmerkungen zum Kurs

- **Beispiele im Kurs**
  - eher **Fallstudien** als Beispiele
  - betonen Aspekte im Zusammenhang mit Parallel Computing
  - zeigen Konzepte, Ideen, mögliche Anwendungen
  - wichtig: das was hinter dem Beispiel steckt

- **Darstellung der Beispiele**
  - auf das Notwendigste reduziert → "we will take some shortcuts"
    - einige Aspekte der Programmierung nicht berücksichtigt
  - im Vordergrund stehen
    - Lesbarkeit, Verständlichkeit, Darstellungsmöglichkeiten
    - Einfluss auf Naming, etc.

Literatur


Structured Parallel Programming
Patterns for Efficient Computation
Michael McCool, Arch D. Robison, James Reinders
ANANTH GRAMA • ANSHUL GUPTA
GEORGE KARYPIS • VIPIN KUMAR
Introduction to
Parallel Computing
Second Edition
PATTERNS FOR PARALLEL PROGRAMMING
TIMOTHY G. MATTSON
T. Rauber · G. Rünger
INFORMATIK IM FOKUS
Multicore: Parallele Programmierung
Springer
Darryl Gove
Multicore Application Programming
For Windows, Linux, and Oracle® Solaris
Developer's Library
THE ART OF MULTIPROCESSOR PROGRAMMING
Maurice Herlihy & Nir Shavit

Literatur: Books

[Mattson] T. Mattson, B. Sanders., N. Massingill, "Patterns for Parallel Programming", Addison-Wesley, 2005.

[McCool] M. McCool, A. Robison, J. Reinders "Structure Parallel Programming", Patterns for Efficient Computation, Morgan Kaufmann, 2012.

[Grama] A. Grama, A. Gupta, G. Karypis, V. Kumar, "Introduction to Parallel Computing", Addison-Wesley, 2nd ed., 2003.

[Gove] D. Gove, "Multicore Application Programming", Addison-Wesley, 2011.

[Herlihy] M. Herlihy, N. Shavit, "The Art of Multiprocessor Programming", Morgan Kaufmann, 2008.

[Rauber] T. Rauber, G. Rünger, "Multicore: Parallele Programmierung", Springer, 2008.

[Huges] C. Hughes, T. Hughes, "Professional Multicore Programming", Wiley, 2008.

Literatur: Paper

[Keutzer] K. Keutzer, T.Mattson, "A Design Pattern Language for Engineering (Parallel) Software", http://parlab.eecs.berkeley.edu/wiki/patterns/patterns

Literatur: WEB Seiten / Online Books

[Parlab]	http://parlab.eecs.berkeley.edu Summer courses (Slides, Videos)
[LLNL]	https://computing.llnl.gov/tutorials Lawrence Livermore National Laboratory, Tutorials
[Matloff]	http://heather.cs.ucdavis.edu/parprocbook Norm Matloff, "Programming on Parallel Machines"
[McKenney]	http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html P. McKenney, "Is Parallel Programming Hard, And, If So, What Can You Do About It?"
[Foster]	http://www.mcs.anl.gov/~itf/dbpp/ Ian Foster, online book, Designing and Building Parallel Programming

Vorlesungsunterlagen

[Arbenz] Advanced Parallel Computing for Scientific Applications, Peter Arbenz, ETHZ.

[Gross] Parallel Programming, etc., Thomas Gross, ETHZ.

[David] MVP, Alexandre David, Aalborg University.

[Kastens] Parallel Programming, Uwe Kastens, Universität Paderborn.

[Mellor] COMP 422, Parallel Computing, John Mellor- Crummey, Rice University.

[Parlab] Materialien aus Summer Courses und Videos.

[Praun] Chr. von Praun, Parallel Programming, Georg-Simon-Ohm Hochschule.

[Yelick] CS 194 Parallel Programming, Katherine Yelick, UC Berkeley.