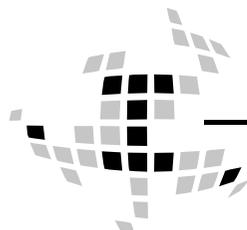


Universität Karlsruhe (TH)

Forschungsuniversität · gegründet 1825

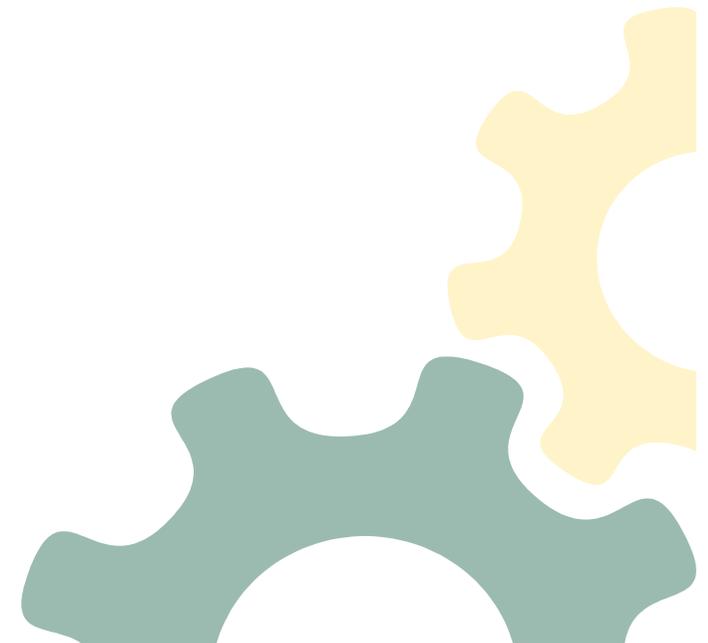
Herausforderung Multikern-Systeme

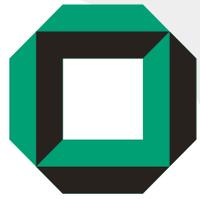
Walter F. Tichy



Fakultät für **Informatik**

Lehrstuhl für Programmiersysteme





Wir erleben gerade eine Umwälzung der Informatik

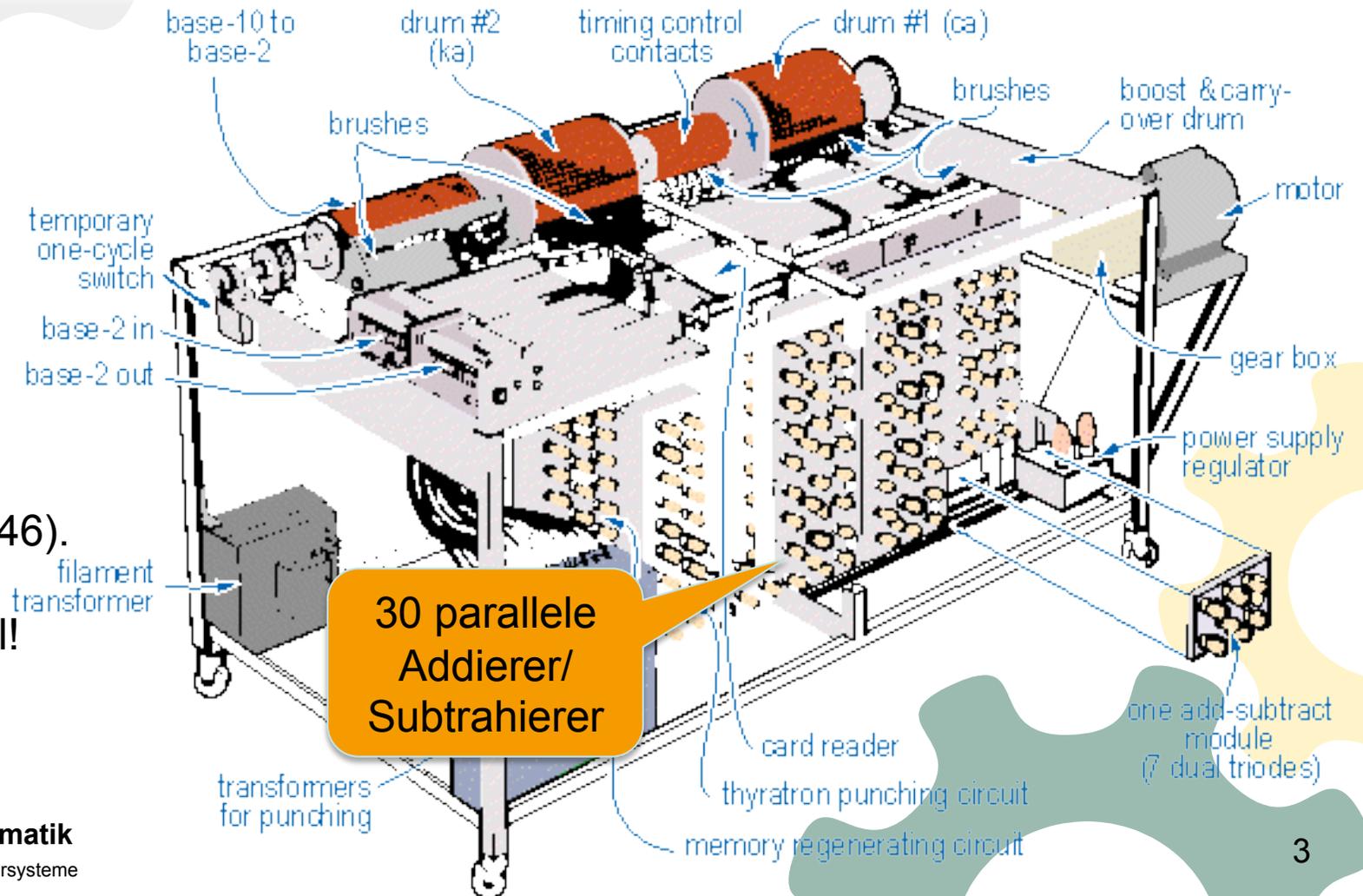
- Von Anfang an war die Informatik geprägt vom sequenziellen Rechner.
- Parallelität war nur in Nischen präsent:
 - Numerisches Rechnen
 - Betriebssysteme, Datenbanken
 - Parallelität auf Instruktionsebene
- Mit Mehrkern-Chips wird Parallelität flächendeckend zur Verfügung stehen.
- Es ist bereits schwierig, einen Rechner mit nur einem Prozessor zu kaufen.

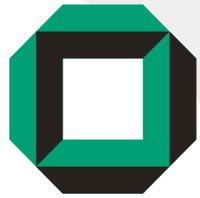


Einige wichtige Parallelrechner

Atanasoff-Berry-Rechner von 1942

Erster digitaler,
elektronischer
Rechner.
Vor ENIAC (1946).
Schon
30-fach parallel!



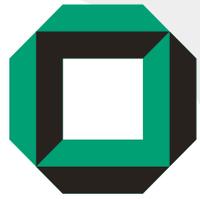


Einige wichtige Parallelrechner

Illiac-IV: SIMD, verteilter Speicher

- Nur ein Exemplar gebaut: 1976
64 Prozessoren
- Schnellster Rechner bis 1981

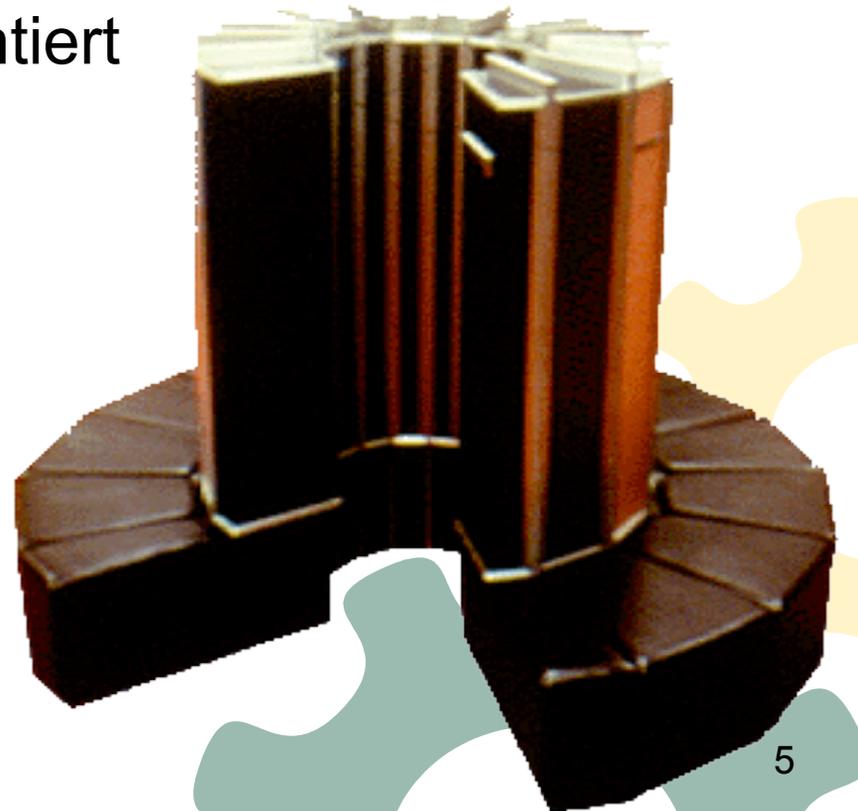


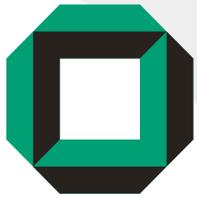


Wichtige Parallelrechner

Cray-1 Vektorrechner

- gemeinsamer Speicher
- Vektorregister mit 64 Elementen
- Vektorinstruktionen implementiert mit Fließbändern.
- Erstmals geliefert 1976
- Damals der zweitschnellste Rechner nach Illiac-VI

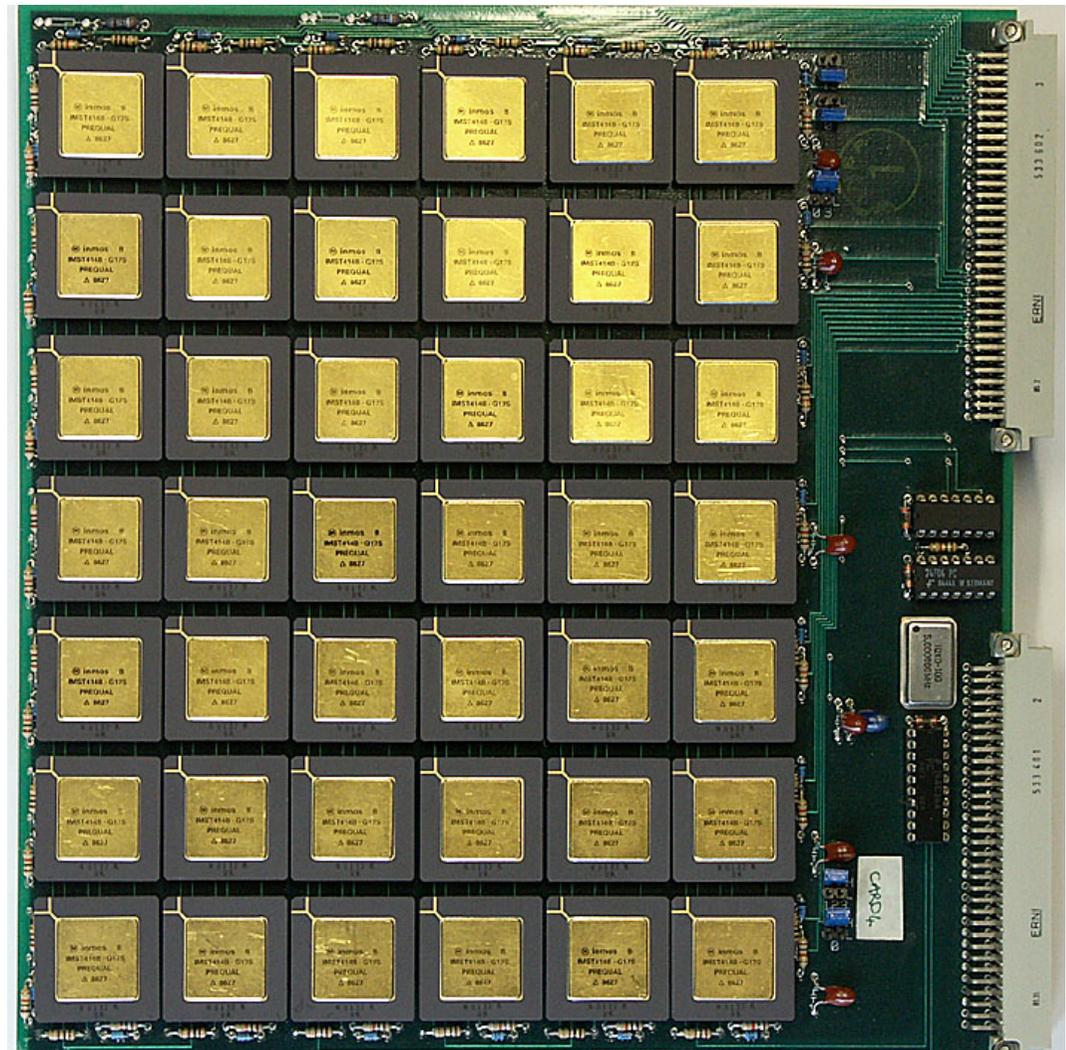


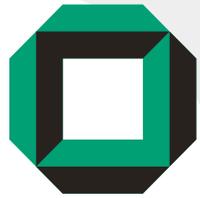


Wichtige Parallelrechner

Transputer

- MIMD-Rechner mit verteiltem Speicher
- Prozessoren mit vier schnellen Verbindungen
- Erstmals geliefert 1984
- Bis zu 1024 Prozessoren
- Occam als Programmiersprache
- Entwickelt von Inmos, Bristol, GB

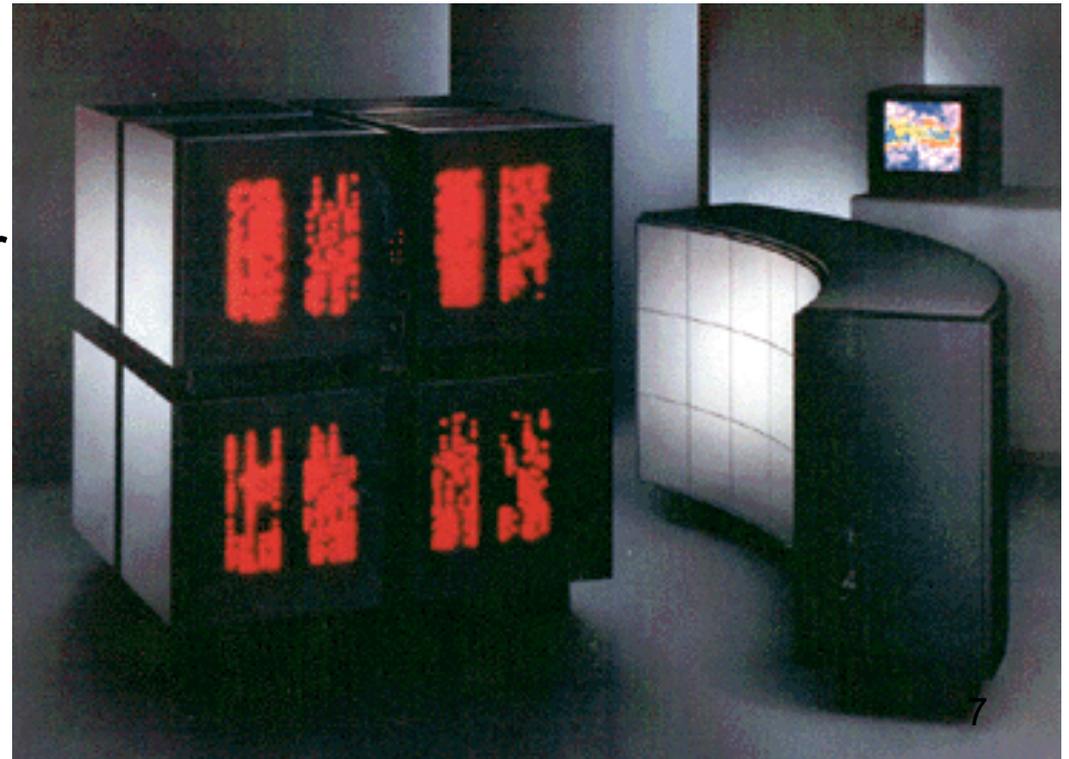


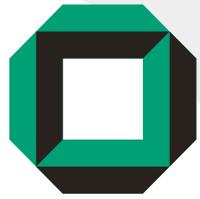


Wichtige Parallelrechner

CM-1 Connection Machine

- SIMD-Rechner, verteilter Speicher
- 65.536 Prozessoren (1-Bit)
- Hyperkubus als Verbindungsnetz
- Erstmals geliefert 1986
- Erster massiv paralleler Rechner



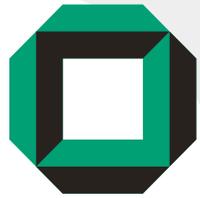


Wichtige Parallelrechner

Rechnerbündel (Cluster Computers)

- Standard PCs verbunden mit Standard-Netzen
- MIMD-Rechner mit verteiltem Speicher
- Kostenvorteile durch Komponenten aus dem Massenmarkt
- Heute die schnellsten Rechner
- Hunderttausende von Prozessoren
- Siehe top500.org





Ihr Laptop— ein Parallelrechner?

Preise Juni 2007



Inspiron™ 6400
15" Notebook für vielseitige
Unterhaltung & 1 GB RAM.

~~729 €~~
659 €
inkl. MwSt., zzgl. 78 €
Versand

Prozessor ?
Intel® Pentium® Dual-Core
T2080 Prozessor (1,73 GHz,
533 MHz, 1 MB L2-Cache)



Inspiron™ 1520
Stylischer Denker, der es
genießt seine Qualitäten
zeigen zu können und gerne
im Mittelpunkt steht.

~~999 €~~
899 €
inkl. MwSt., zzgl. 78 €
Versand

Prozessor ?
Intel® Core™ 2 Duo T5450
Prozessor (1,66 GHz, 667
MHz, 2 MB L2-Cache)



Inspiron™ 1720
Unterhaltung & Spaß
garantiert! Technologie
genau angepasst für Ihren
Lifestyle. Jetzt in 8 Farben.

1.049 €
inkl. MwSt. und Versand

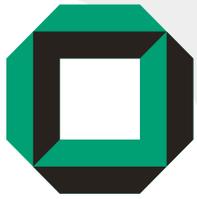
Prozessor ?
Intel® Core™ 2 Duo
T5250 Prozessor (1,5
GHz, 667 MHz, 2 MB
L2-Cache)



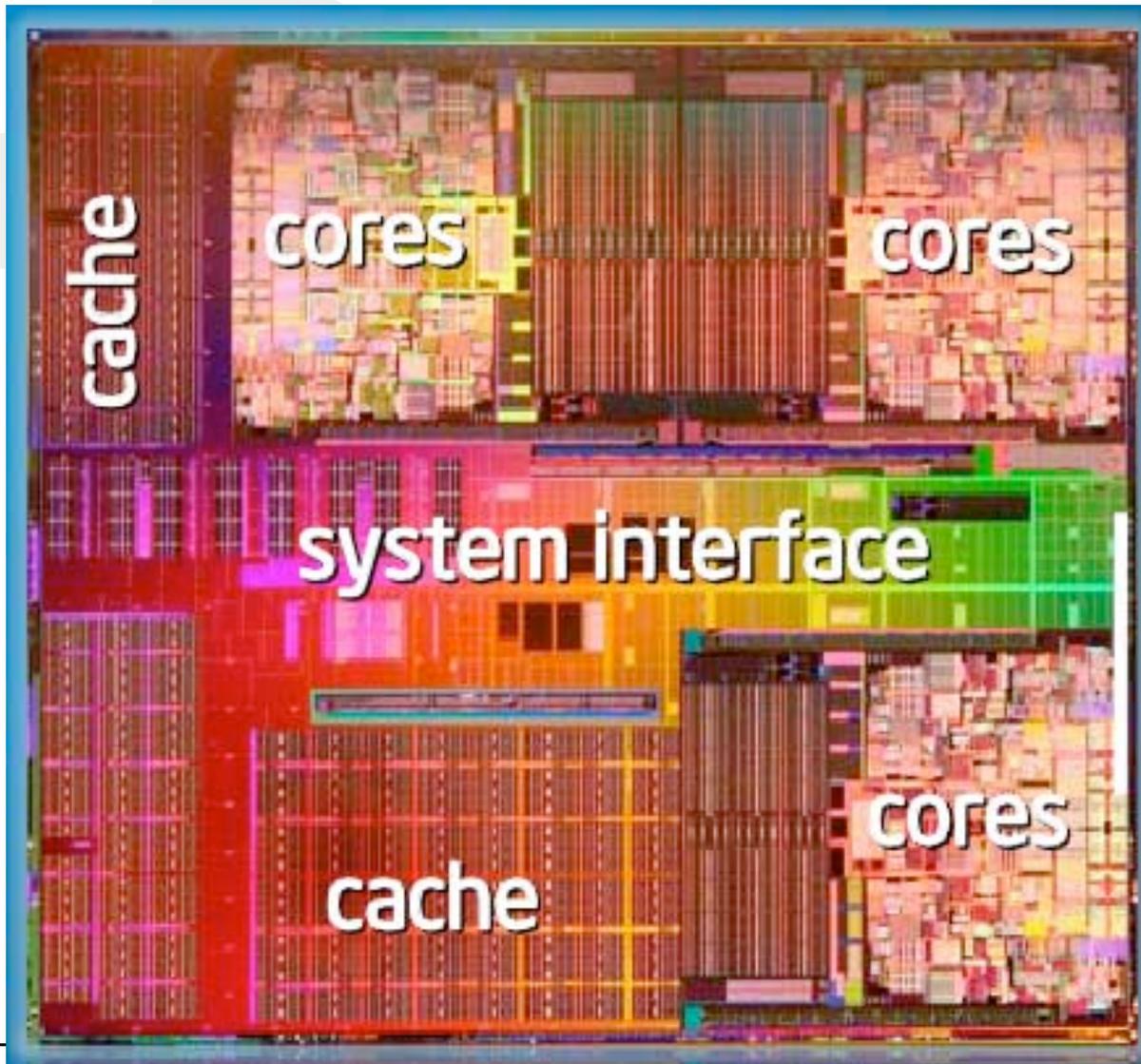
Inspiron™ 1520
Schlank, elegant und noch
etwas schlauer und stärker.
Jetzt in 8 Farben.

~~1.129 €~~
1.079 €
inkl. MwSt., zzgl. 78 €
Versand

Prozessor ?
Intel® Core™ 2 Duo T7100
Prozessor (1,8 GHz, 800
MHz, 2 MB L2-Cache)



Intel Dunnington 6 Prozessoren auf einem Chip



**3 x 2 Xeon
Prozessoren**

**(nicht HW-
mehrfädig)**

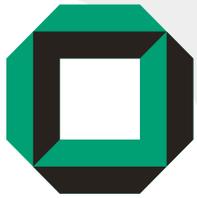
2,6 GHz

130 W

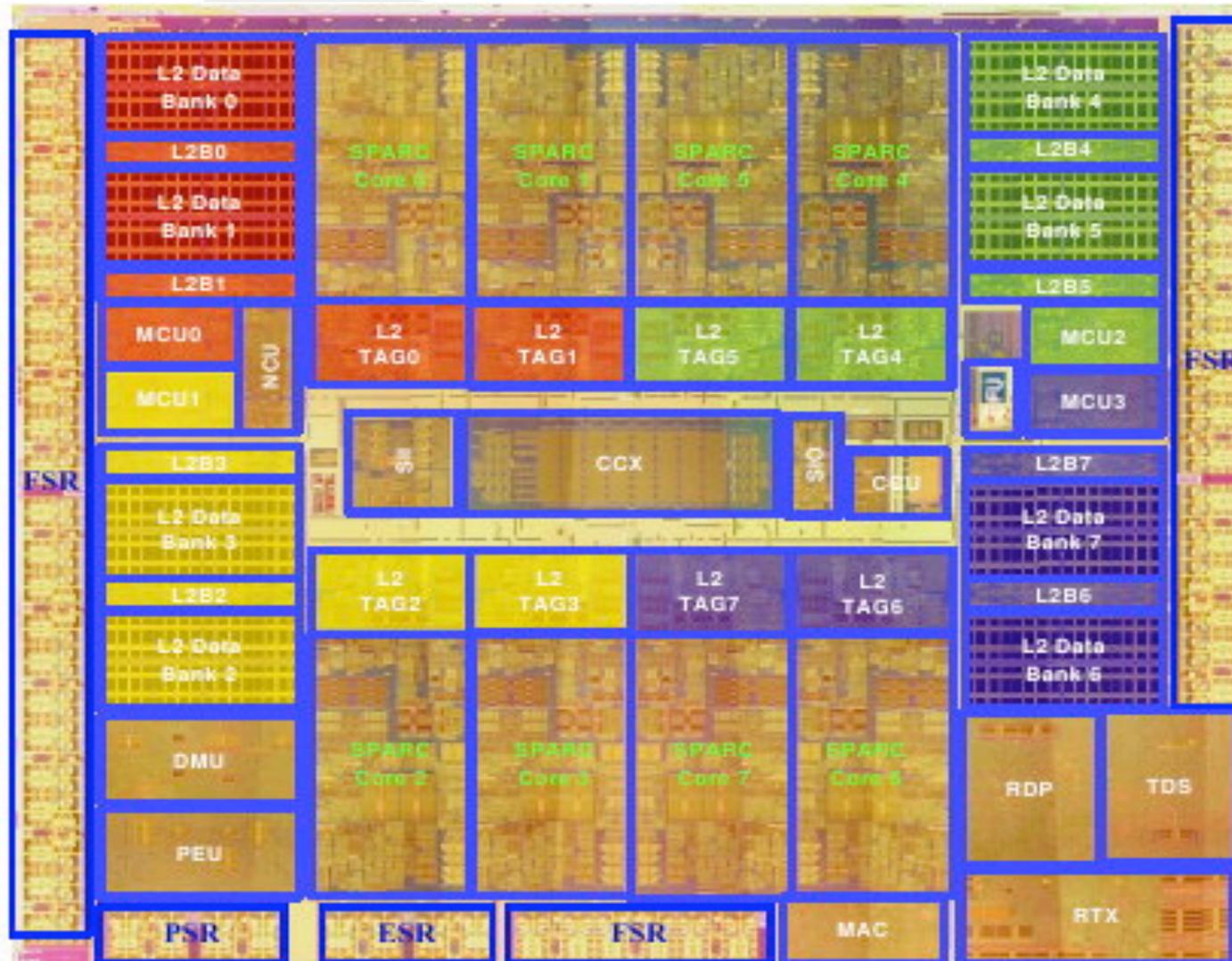
45nm Technik

Lieferbar 2008





Sun Niagara 2: 8 Prozessoren auf 3,42 cm²



**8 Sparc
Prozessoren**

**8 HW-Fäden pro
Prozessor**

8x9 Kreuzschiene

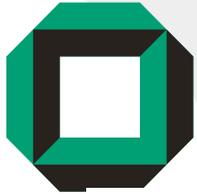
1,4 GHz

75 W

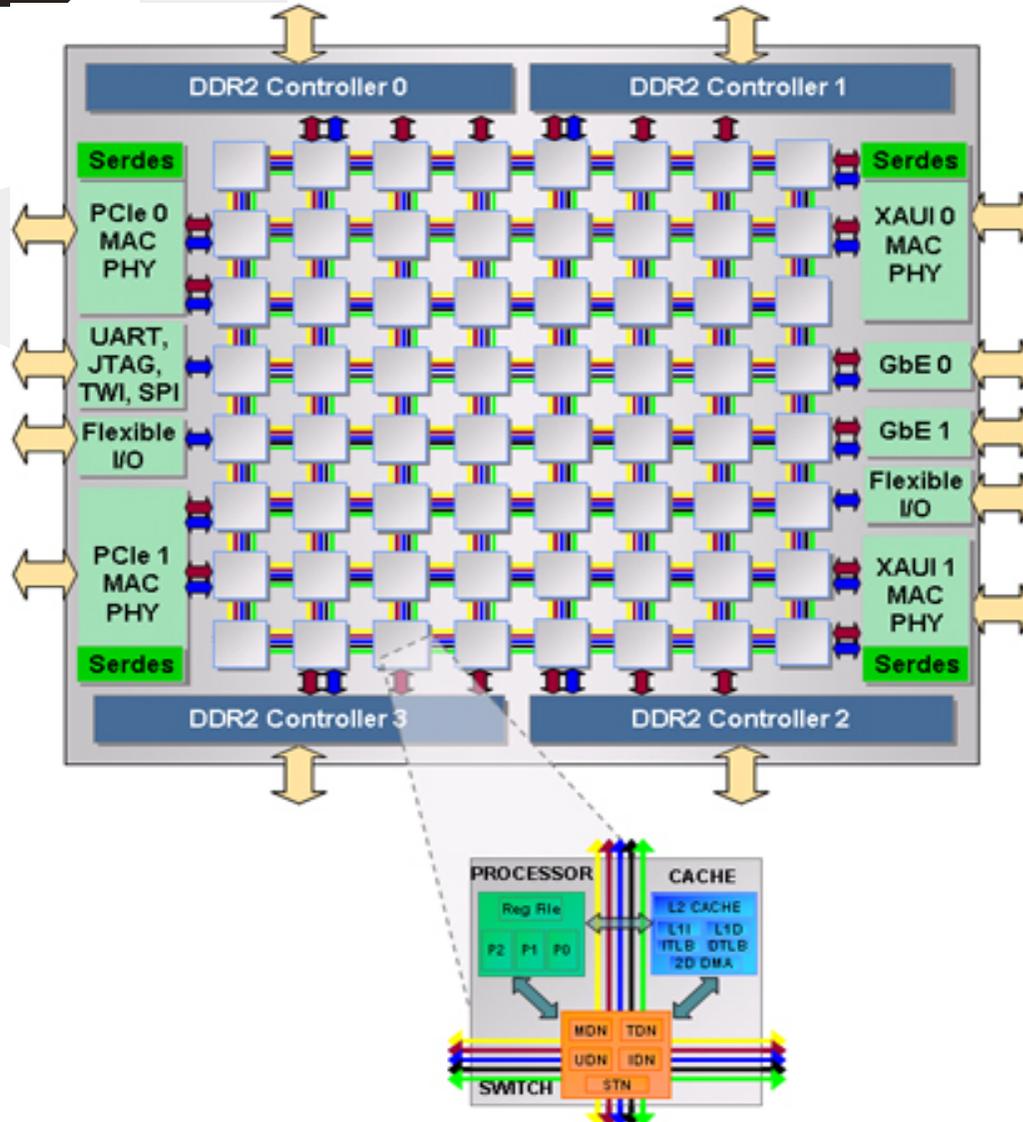
65nm Technik

Lieferbar 2007

Erste Version 2005



Tilera's TILE64



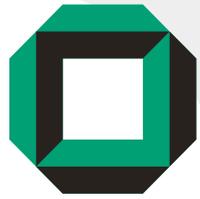
**64 VLIW Prozessoren
plus Gitter auf einem
Chip**

**Für eingebettete
Anwendungen
(Netz- und
Videoanwendungen)**

700 MHz

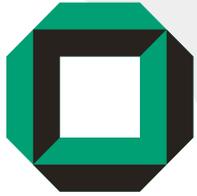
22 W

Lieferbar 2007



Aber der Rekord ist das nicht...

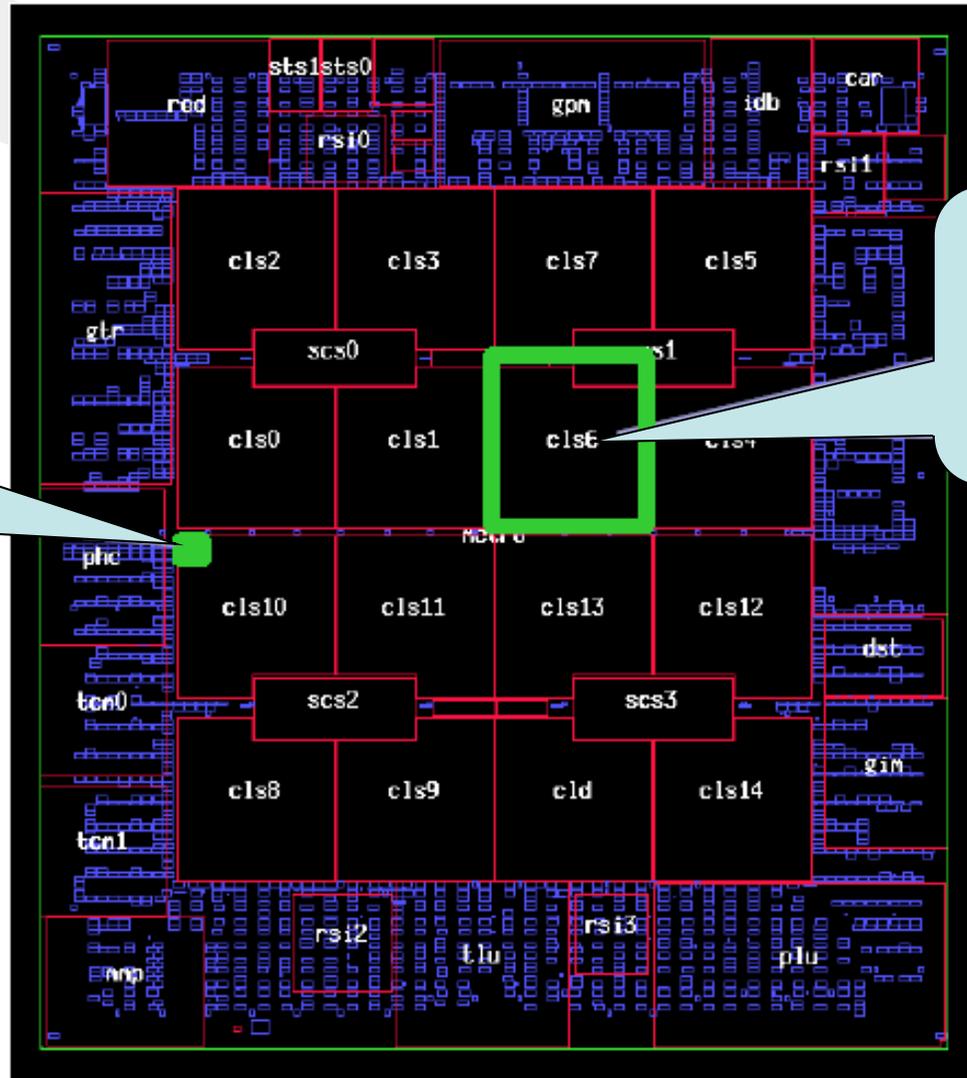
- Fast unbemerkt hat Cisco bereits 2005 ein Paket-Vermittlungs-Chip gebaut



Cisco Metro: 192 Prozessoren auf 3,24 cm²

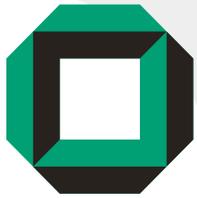
0,5 mm²
pro
Prozessor

**192 Tenselica
Prozessoren
10 Gips
250 MHz
35W
130nm Technik
(2005)**

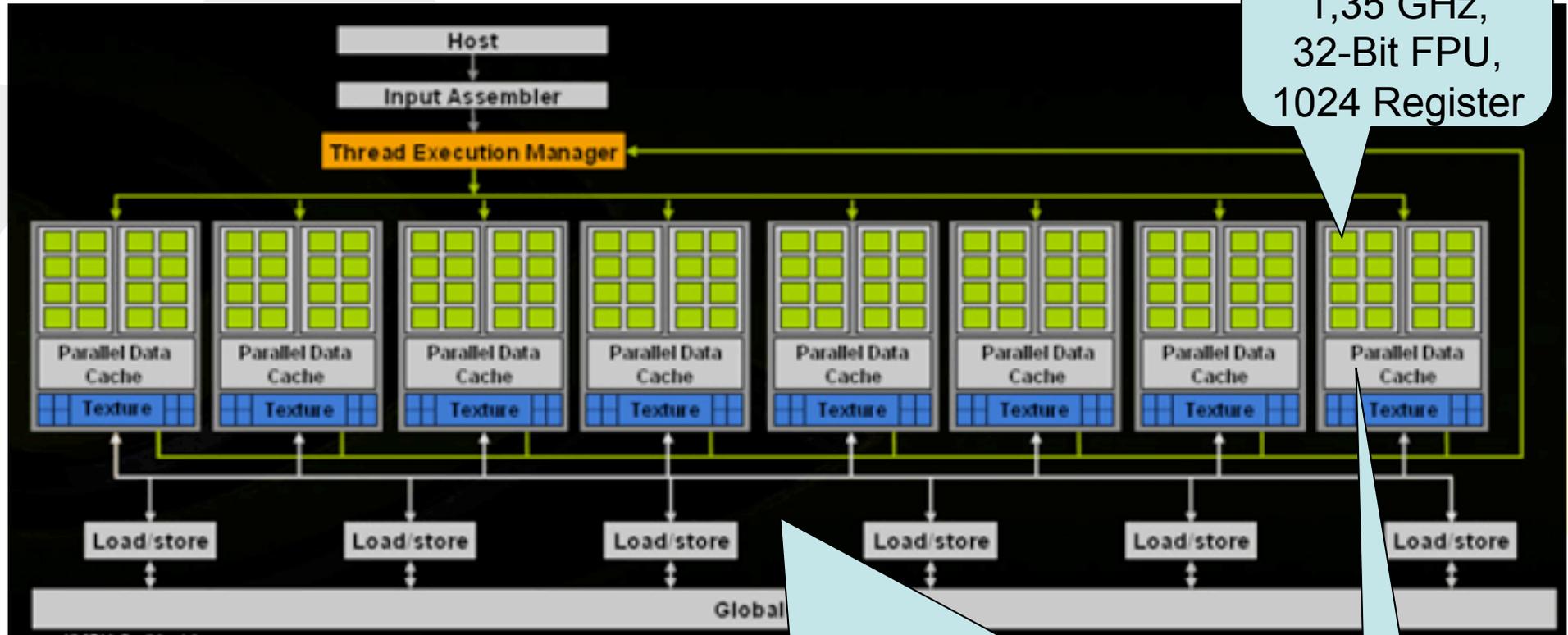


16 Gruppen
zu je
12 Prozessoren
(192 insgesamt)

Bei 45 nm Technik
wäre Raum für
das Achtfache an
Prozessoren!
Und bei 32nm?



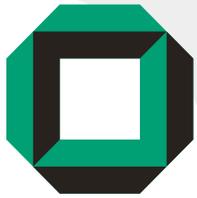
Nvidia GeForce 8 Graphics Processing Unit



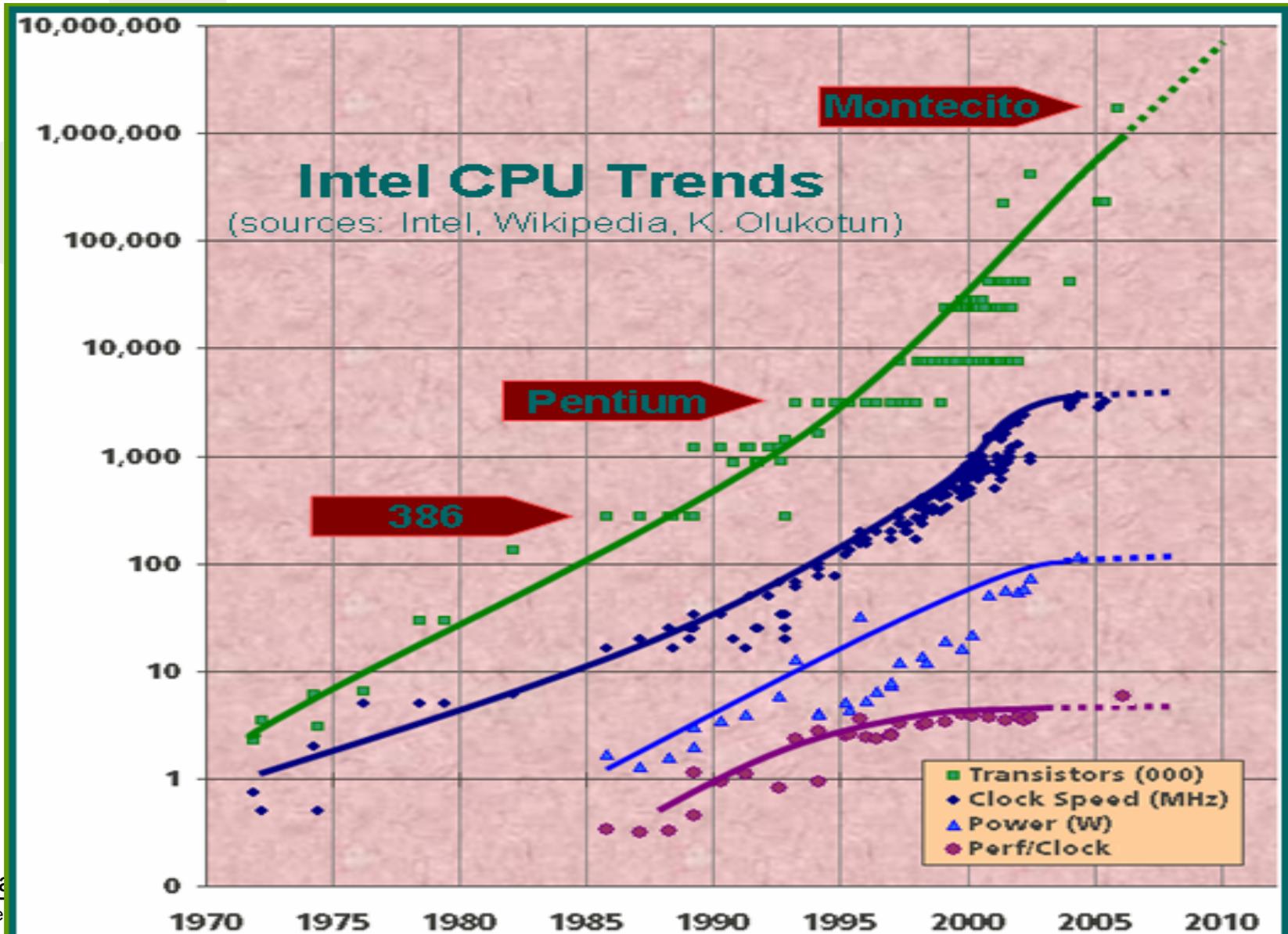
Prozessor,
1,35 GHz,
32-Bit FPU,
1024 Register

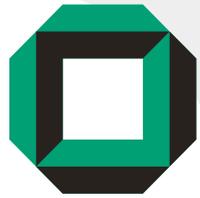
16 KB

128 Prozessoren insgesamt, jeder mit 96 Fäden in HW
Insgesamt 12288 HW-Fäden!
SIMD



Was ist passiert?





Moore'sche Regel, neue Version

Verdopplung der Anzahl
Prozessoren pro Chip
mit jeder Chip-Generation,
bei etwa gleicher Taktfrequenz

Parallelrechner werden in naher Zukunft
flächendeckend zur Verfügung stehen.



Was tun mit all den Kernen?

- „Wer braucht 100 Prozessoren für M/S Word?“
 - Mangel an Vorstellungskraft, Informatik-Ausbildung?
 - Gesucht werden überzeugende Anwendungen, die Hunderte von Prozessoren gebrauchen können.
- Wovon könnten alle Nutzer von PCs, eingebetteten Systemen und Mobiltelefonen profitieren?

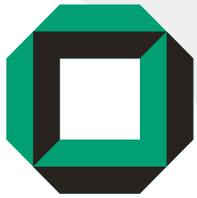


Automatischer Protokollführer

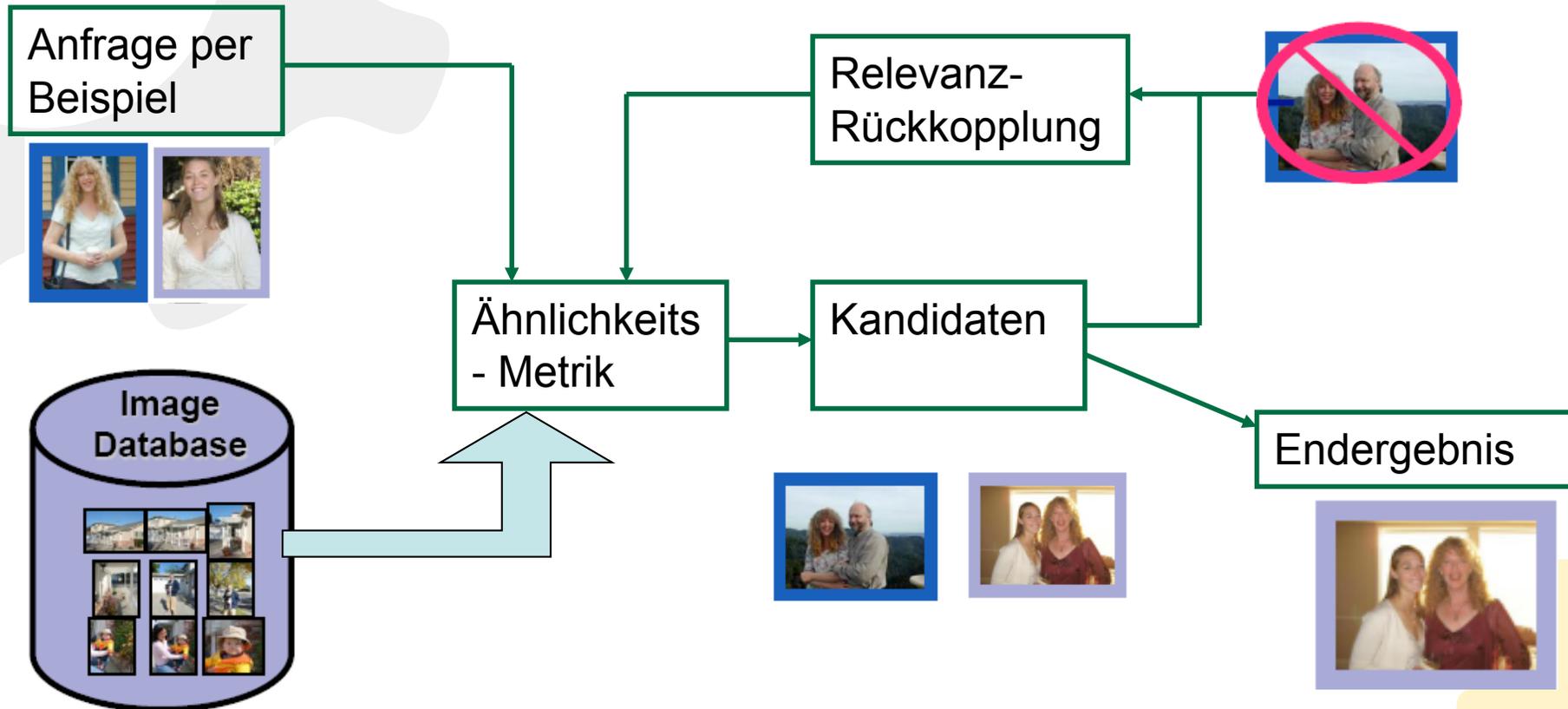
Tragbare Geräte
kooperieren
bei Besprechung:

- protokollieren Aussagen
- identifizieren Sprecher,
- übersetzen,
- kondensieren
- suchen Gewünschtes

The screenshot displays the AMI Meeting Browser interface. The top-left panel shows a list of findings for a curved case, categorized into materials (Plastic, Metal, Wood) and chips (Simple -> buttons, regular, advanced chip on print). The top-right panel shows an automatic transcript of the meeting, with a search bar and a manual transcript option. The bottom-left panel shows a video feed of the meeting in progress. The bottom-right panel shows a timeline of the meeting from 09:30 to 18:00, with colored bars indicating speaker activity. The bottom-most panel shows a control panel with 'Talking' and 'Nodding' options.



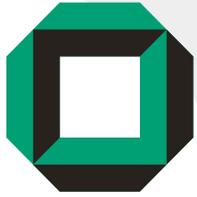
Beispiel 1: Inhalt-basierte Bildsuche



Tausende
Bilder

Angepasst an Charakteristiken persönlicher Bildbanken:

- Bilder nicht gekennzeichnet
- Viele Bilder von wenigen Personen
- Komplexe Bilder mit Personen, Objekten, Orten, Ereignissen

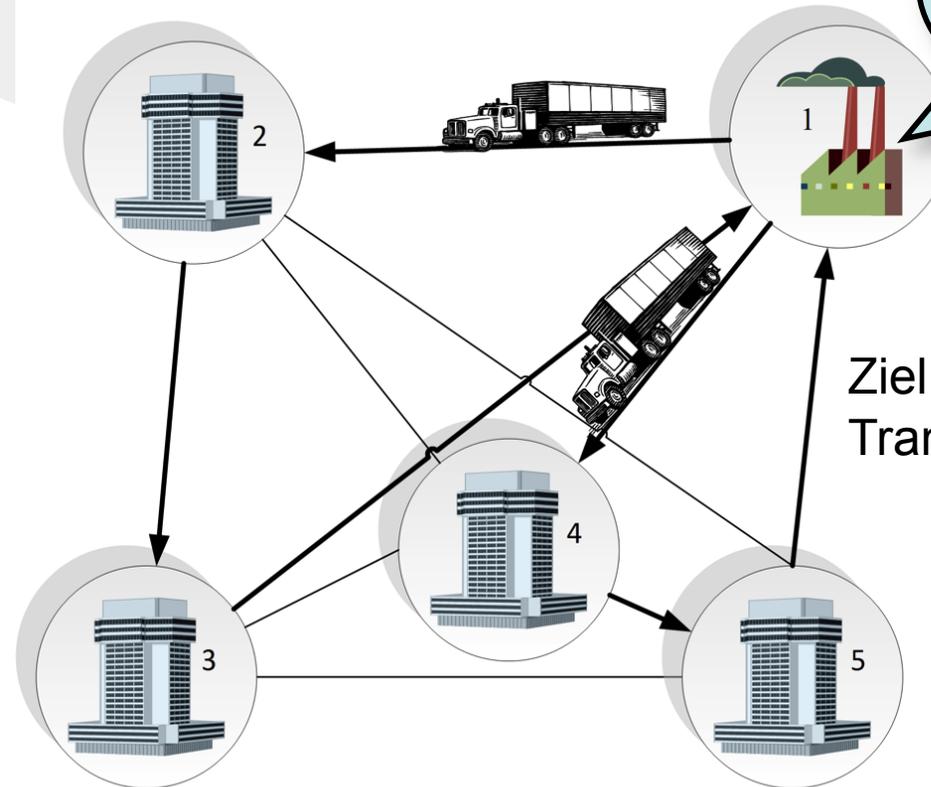


Beispiel 2: Transportproblem (Diplomarbeit bei SAP)

Welche
Aufträge?

Auf welchen
Fahrzeugen?

Über welche
Strecken?

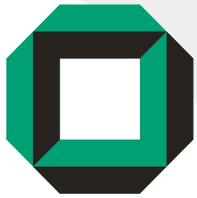


Gegeben:

- Lieferaufträge
- Fahrzeuge
- Streckennetz

Ziel: kostenoptimaler
Transportplan

Verallgemeinerung des Problems des Handlungsreisenden



Warum parallelisieren?

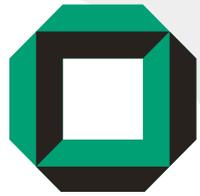
Reale Transportszenarien

Szenarien	1	2	3
Aufträge	804	1177	7040
Ladedimensionen	3	2	4
Aufladestationen	1	1	3
Abladestationen	31	559	1872
Zwischenstationen	0	5	0
Fahrzeuge	281	680	2011
Fahrzeugtypen	7	3	10
Zeitfenster		1	64

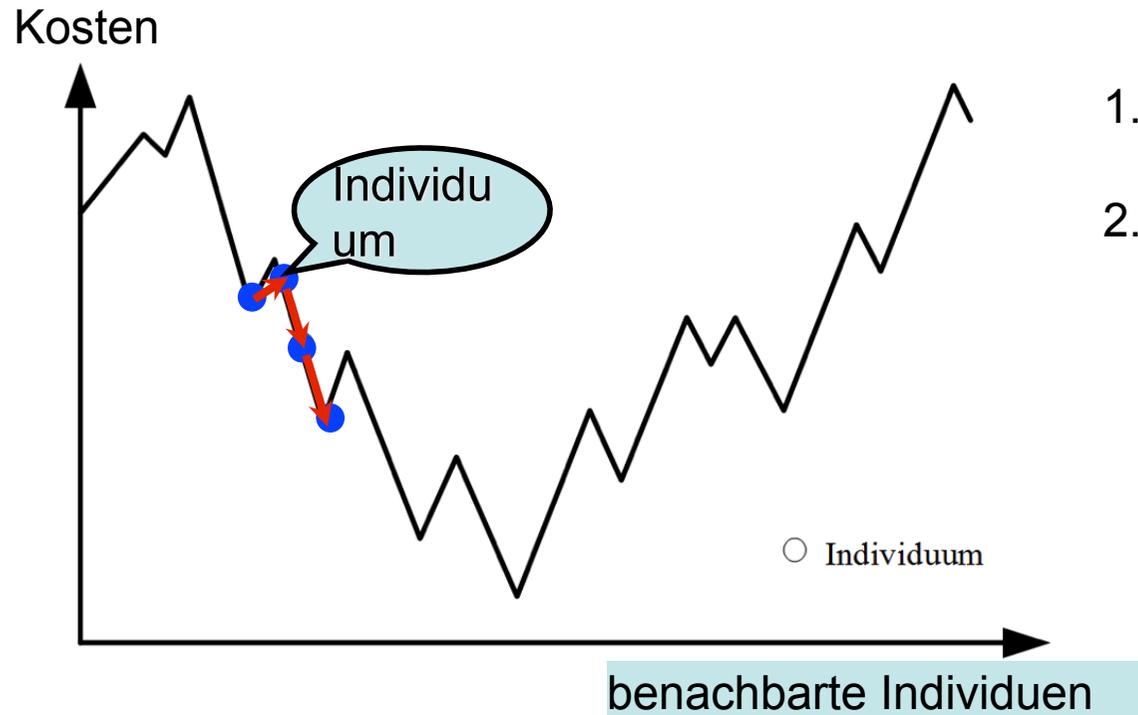
Noch zu berücksichtigen:
Ruhezeiten der Fahrer,
Ladezeiten, Anhänger,
mit/ohne Kühlung,
Fähren, Schiffe....

Sequenzielle Lösung:
-150.000 Zeilen C++
-Evolutionärer
Algorithmus

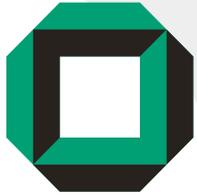
mehrere Stunden
Rechenzeit
für gute Lösungen nötig!



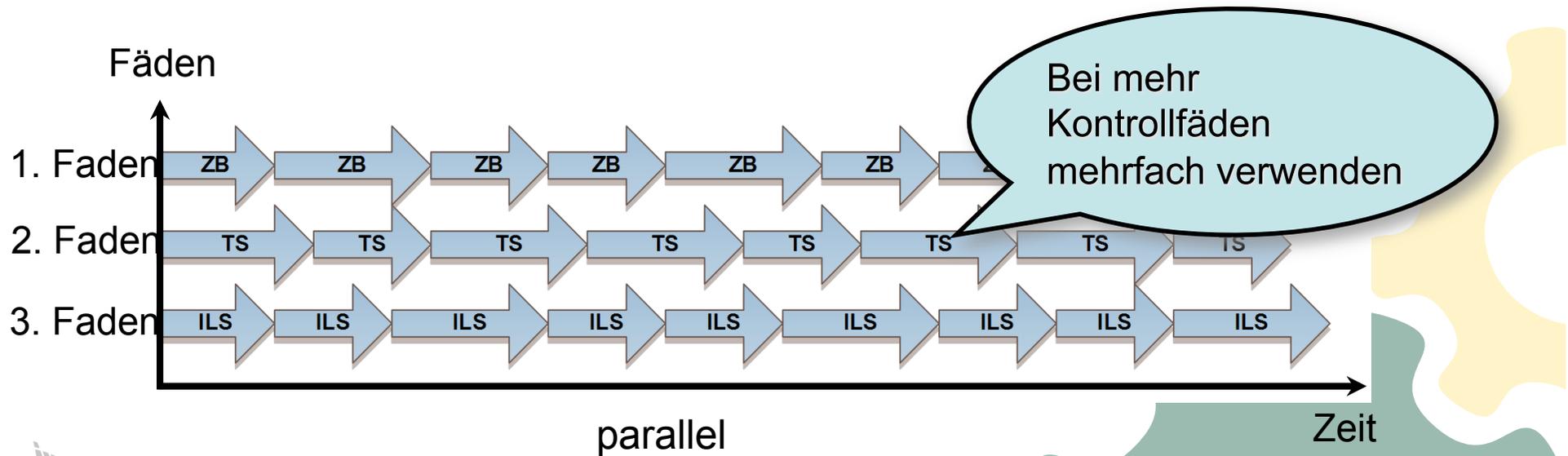
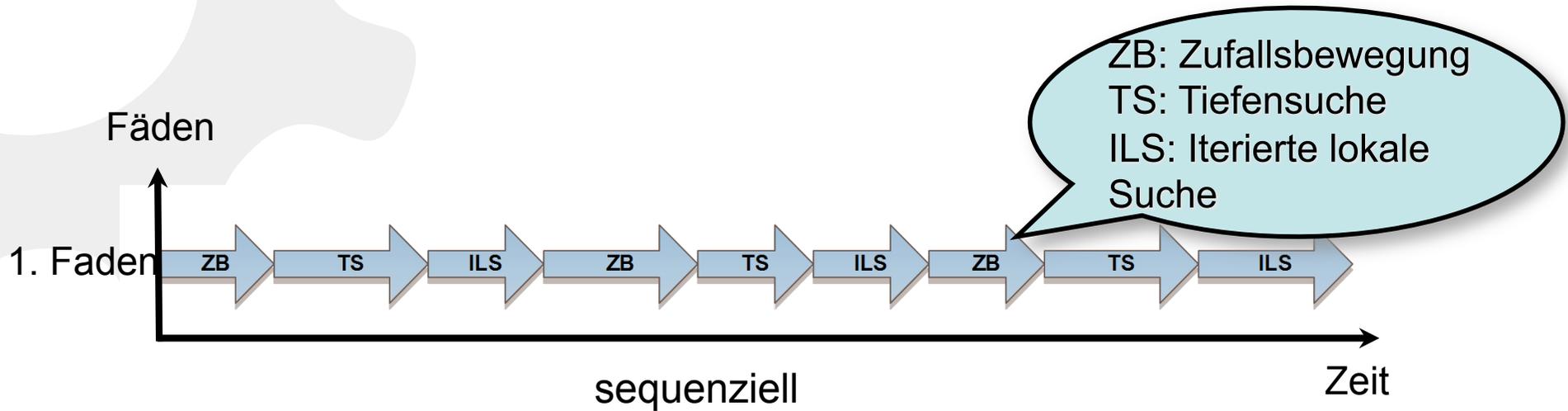
Suchalgorithmus

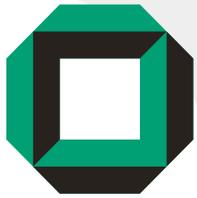


1. Beginne mit initialer Lösung
2. Solange Kostenschranke nicht erreicht:
 - Optimierte Lösung mit lokalen Verbesserungen (genaue Untersuchung der Nachbarschaft)
 - Entkommen aus lokalem Minimum durch zufälligem Sprung im Lösungsraum (Führt zu breiter Abdeckung des Suchraums)



Prinzipieller Ablauf

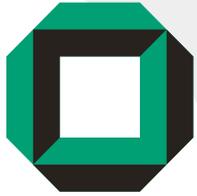




Untersuchte Lösungen in 2 Minuten

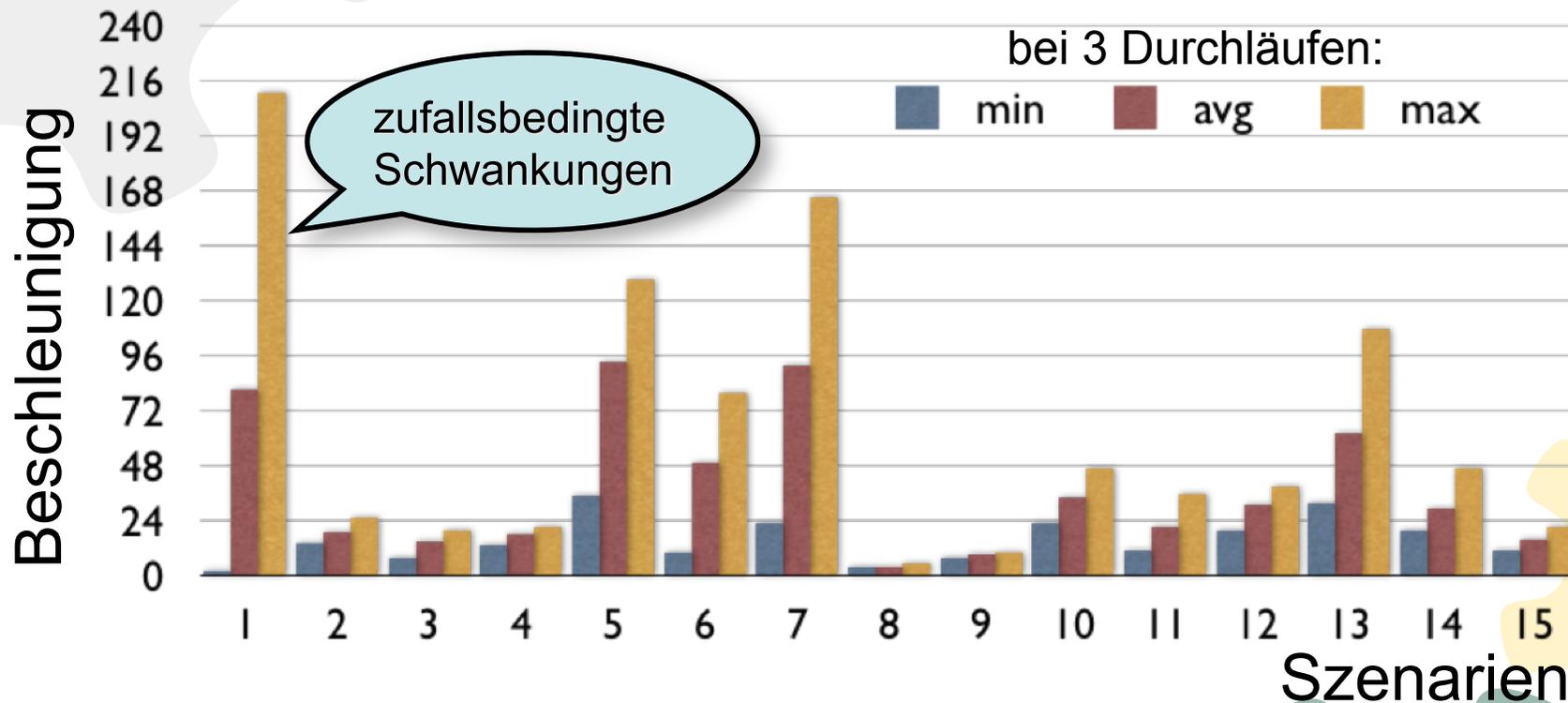


Auf Rechner mit 4 Intel Dunnington Chips
(4 * 6 Prozessor-Kerne)

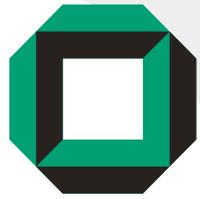


Beschleunigung bei 24 Fäden

Beschleunigung: Erreichen einer festgelegten Kostenschranke

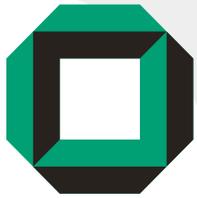


Beschleunigung bei Durchführung aller Szenarien: 17,37
Superlineare Beschleunigung bei frühzeitigem Finden
guter Kandidaten

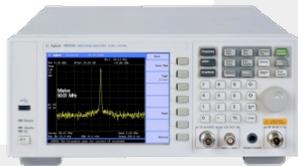


Was ist das generelle Problem bei der Parallelisierung?

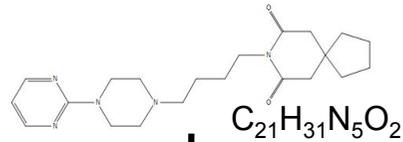
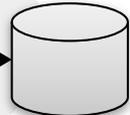
- Für den Erfolg müssen Beschleunigung, Programmier-Produktivität und Qualität gleichzeitig gemeistert werden.
 - Parallelisierung nur bei Beschleunigung interessant
 - Produktivität und Qualität dürfen nicht schlechter werden!
- Sprachen, Debugger, Analysewerkzeuge sind unzureichend für Parallelität (Thread \approx Goto?)
- Die meisten Programmierer sind unzureichend auf Parallelität vorbereitet.



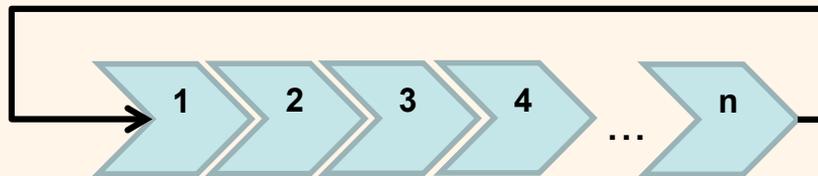
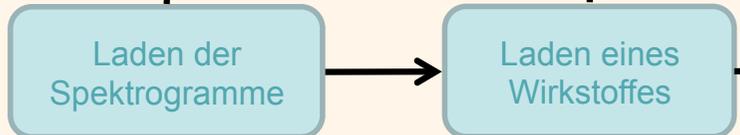
Beispiel 3: Analyse von Massenspektrogrammen



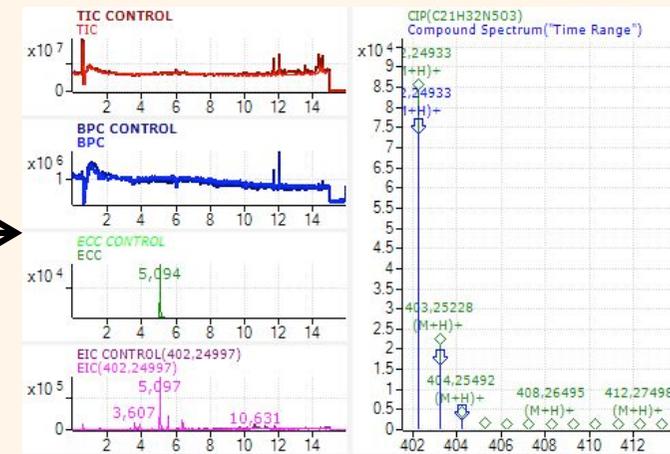
Massenspektrogramme



Agilent Technologies

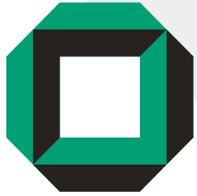


Mehrstufiges Algorithmen-Fließband:
Suchen von Metaboliten in den Spektrogrammen
Parallelisierungspotential

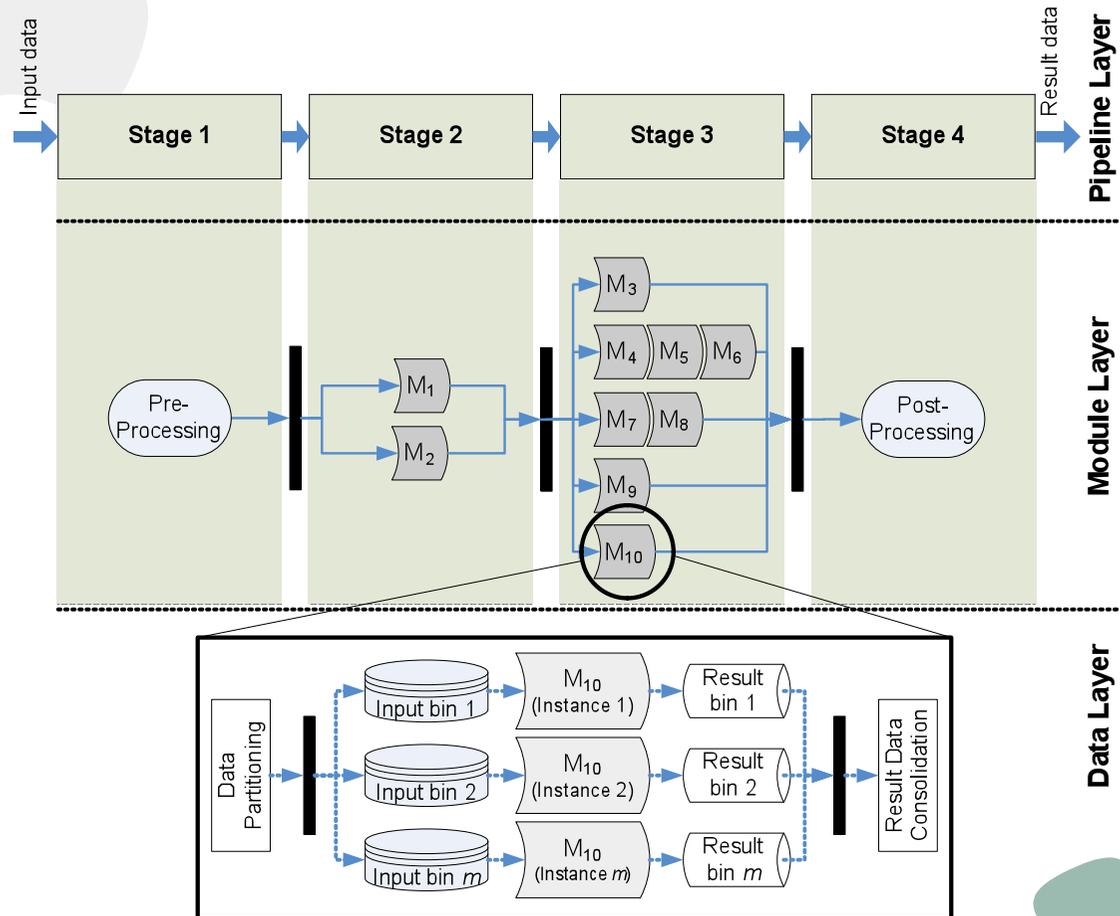


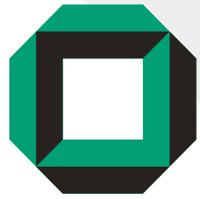
Ergebnis: Zeitabhängige Darstellung der Metaboliten

Desktop-Applikation



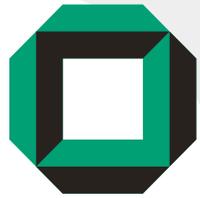
Mehrschichtige Parallelisierung mit Entwurfs-Mustern





Auto-Tuning

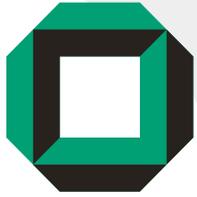
- **Problem:** Bestimme alle leistungsrelevanten Parameter in allen Ebenen um globales Leistungsoptimum zu finden.
- Parameterwerte abhängig von HW und SW (z.B. Anzahl Kerne, Ebenen, Anzahl Fließband-Stufen, Fließband-Struktur, Größe der Datenpartitionen)
- Einstellung per Hand zu aufwendig



Auto-Tuning (2)

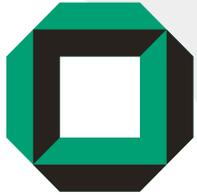
- **Lösung:** Automatischer Parameter-Optimierer
 - Als Bibliothek, die einfach in eine Anwendung integrierbar ist, und die Parameterkomb. durchprobiert
 - Nimmt Stichproben, um den Parameterraum zu reduzieren (kann i.d.R. nicht alles durchprobieren)
 - Leistungsunterschiede zwischen bester und schlechtester Konfiguration der Massenspektrogramm-Analyse: Faktor 1,9 (bei einer Gesamtbeschleunigung von 3,1 auf 8 Kernen)
 - bei einer Graphersetzungsanwendung trägt Autotuning sogar den Faktor 4,2 bei Gesamtbeschleunigung 7,7
bei.

Getestet auf 8-Kern-Rechner (2x Intel Xeon E5320 Quad-Core 1,86 GHz)

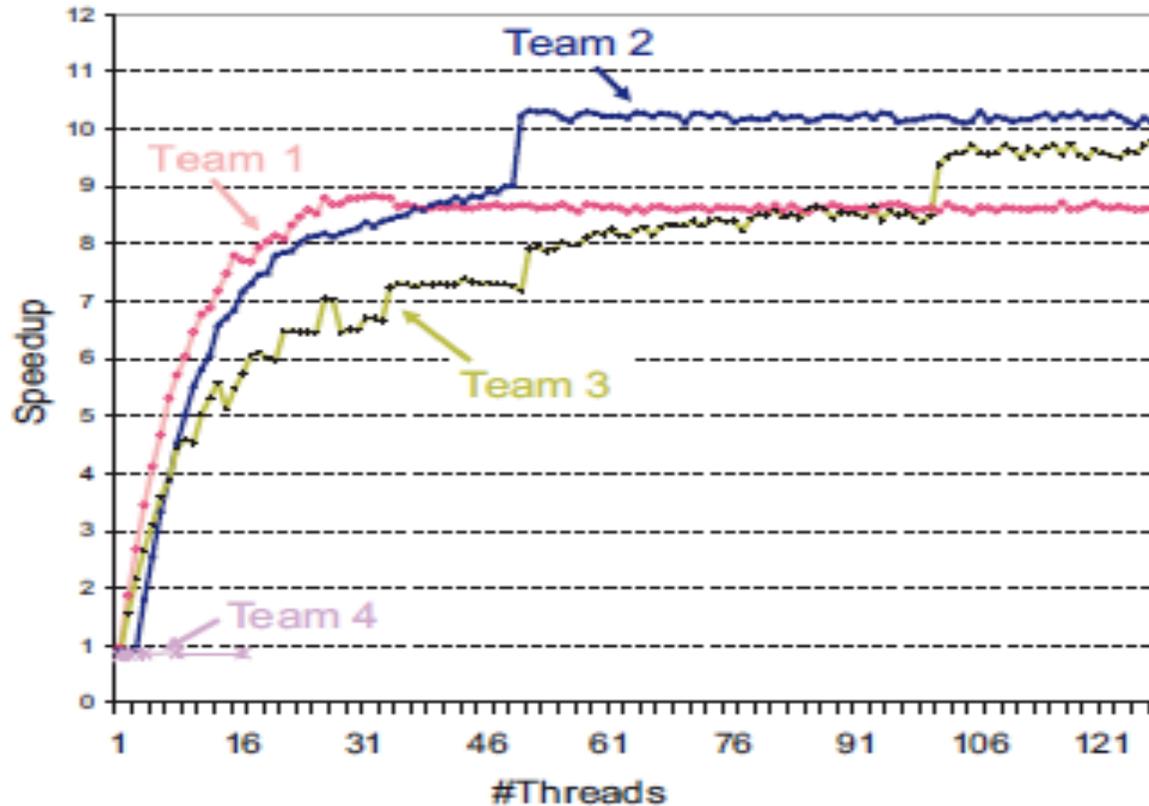


Beispiel 4: BZip2

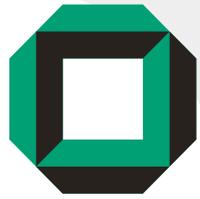
- Kompressionsprogramm
- Vielfach im täglichen Gebrauch
- Ca 8000 Zeilen, quelloffen
- Im Wettbewerb parallelisiert
 - 4 Teams von je zwei Personen gegeneinander
 - 3 Monate Training in OpenMP und Posix-Fäden
 - Wettbewerb dauerte 3 Wochen (Abschlussarbeit)



Beschleunigung

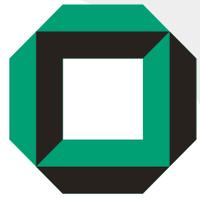


Gewinner erreichte 10-fachen Beschleunigung auf Sun Niagara T1 (acht Prozessoren, 32 HW-Fäden).



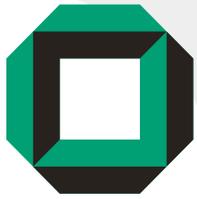
Was hat funktioniert?

- Massive Umstrukturierung war erforderlich
 - Teams, die hier wenig investierten, hatten keinen Erfolg.
 - Die Gewinner parallelisierten erst am Tag vor der Abgabe; die übrige Zeit von 3 Wochen wurde nur dazu benutzt, das Programm für die Parallelisierung vorzubereiten.
 - Viele Abhängigkeiten, Seiteneffekte und Optimierungen für den sequentiellen Ablauf mussten beseitigt werden, bevor überhaupt Parallelisierung möglich war.



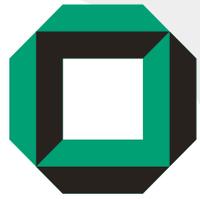
Was hat nicht funktioniert?

- Parallelisierungskonstrukte nach und nach hinzu zu fügen hat bei keinem Team funktioniert.
- Nur den Laufzeit-kritischen Pfad zu parallelisieren reichte nicht.
- Feingranulare Parallelisierung innerer Schleifen hat keine nennenswerte Beschleunigung erzielt.
 - Parallelisierte Einheiten müssen größere Arbeitsabschnitte umfassen.
- Algorithmen waren so spezielle, dass geeignete, parallelisierte Bibliotheksroutinen kaum zu erwarten sind.



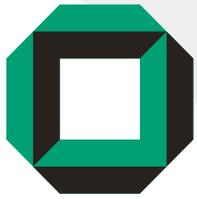
Parallelisierung ist keine Geheimkunst

- Arbeite nach Plan! Herumprobieren reicht nicht.
- Entwickle Hypothesen und Abschätzungen, wo Parallelisierung am meisten Gewinn bringen könnte!
- Berücksichtige mehrere Parallelisierungsebenen!
- Benutze Parallelisierungsmuster!
 - Auftraggeber/Auftragnehmer, Fließband, Gebietszerlegung, Erzeuger/Verbraucher.
- Verzweifle nicht bei der Restrukturierung!
- Ärmel hochkrempeln und anfangen!



Wozu die Rechenleistung einsetzen?

- Intuitive Schnittstellen, mit Bild- und Sprachverarbeitung
- Vorausschauende Anwendungen, die „ahnen“, was der Benutzer will
- Eingehende Modellierung des Benutzers und der Umgebung
- Berechnungen, die heute zu langsam sind
- Erhöhung der Zuverlässigkeit



Forschungsthemen

Parallele Softwaretechnik

- Bessere Programmiersprachen zum direkten Ausdruck paralleler Abläufe
- Parallele Entwurfsmuster und Architekturen
- Parallele Algorithmen und Bibliotheken
- Robustheit, Suche von Parallelisierungsfehlern
- Leistungsvorhersagen für parallele Architekturen
- Werkzeuge zur Restrukturierung und Parallelisierung
- Neue Anwendungsklassen
- U.v.a.m.



XJava: Parallelitätsmuster kompakt ausgedrückt

Operator „=>“ verknüpft Prozesse in Fließbandstruktur, wie in Unix.

```
compress (File in, File out) {  
    read(in) => compress() => write(out);  
}
```

Liest Datei,
gibt Folge von
Blöcken aus

Datenstrom

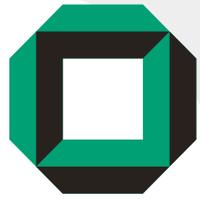
Liest Blöcke,
gibt kompr.
Blöcke aus

Datenstrom

Schreibt
Blöcke in
Datei

Filter laufen parallel, bis Ende der Eingabe.
Geeignet für Fließbänder, Auftraggeber/Arbeiter, Erzeuger/Verbraucher.
Datenströme sind typisiert und typsicher.





XJava

Operator „|||“ lässt Prozesse parallel laufen:

```
compress(f1, f1out) ||| compress(f2, f2out);
```

Methoden laufen in eigenen Fäden parallel ab,
implizite Barriere am Schluss.

Für Prozess- und Datenparallelismus.



Auftraggeber/Arbeiter in XJava

Eingabetyp

=>

Ausgabetyyp

- Deklaration für 1 Auftraggeber, 3 Arbeiter:

```
void => X master() { /* Auftraggeber*/ }  
X => void w()      { /* Arbeiter*/ }  
X => void gang()   { w() ||| w() ||| w(); }
```

- Deklaration für i Arbeiter (dynamisch):

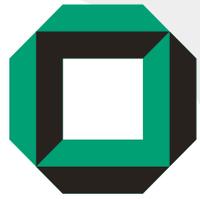
```
X => void gang()   { w():i; }
```

- **master()** => **gang()**

`master` übergibt den `w` in `gang` im Umlaufverfahren Elemente vom Typ `X`.

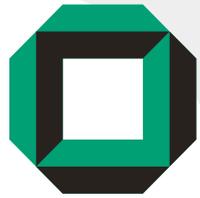
- **master()** =>* **gang()**

übergibt alle Elemente an alle Arbeiter.



XJava Erweiterungen

- Einfach zu verstehende Parallelität
- Voll integriert in Java
- Typsicher
- Einfacher zu handhaben als Fäden oder Bibliotheken
- Weniger Code, weniger Fehlerquellen
- Keine Synchronisationsfehler bei Übergabe von Daten über Puffer (Fließbänder)
- Vollautomatisches Autotuning möglich



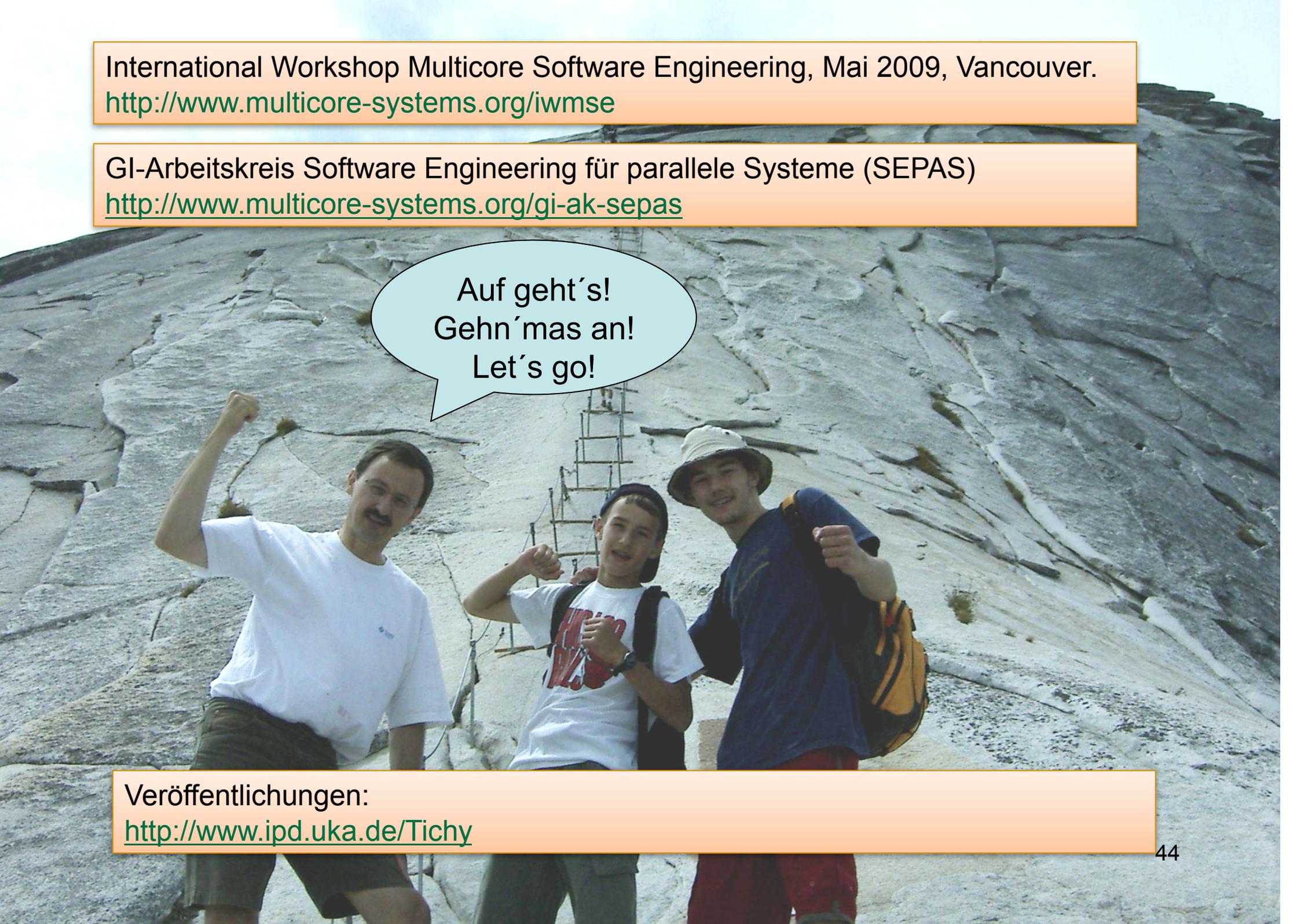
Schluss

- Zukünftige Leistungssteigerungen nur über Parallelität
- Ziel: Beschleunigung von Anwendungen bei gleichbleibender Produktivität, Qualität
- ... während sich die Anzahl der Prozessorkerne pro Chip alle zwei Jahre verdoppelt.
- Viele der Grundlagen der Informatik müssen neu überdacht werden.

„Reinventing Computing“

International Workshop Multicore Software Engineering, Mai 2009, Vancouver.
<http://www.multicore-systems.org/iwmse>

GI-Arbeitskreis Software Engineering für parallele Systeme (SEPAS)
<http://www.multicore-systems.org/gi-ak-sepas>

A photograph of three people climbing a steep, light-colored rock face. The person on the left is a man in a white t-shirt and dark shorts, raising his right fist. The person in the middle is a younger man in a white t-shirt with red text and a backpack, also raising his right fist. The person on the right is a man in a blue t-shirt, a white bucket hat, and a yellow and black backpack, with his right arm around the middle person's shoulder. A wooden ladder is visible on the rock face behind them. A light blue speech bubble is positioned above the three people, containing the text 'Auf geht's! Gehn' mas an! Let's go!'.

Auf geht's!
Gehn' mas an!
Let's go!

Veröffentlichungen:
<http://www.ipd.uka.de/Tichy>