Leistungsanalyse

- Problemstellung
- Ziele der Leistungsanalyse
- Grundüberlegungen und Historie
- Leistungsmaße
- Amdahls Gesetz
- Einfluß der Problemgröße
- Superlinearer Speedup

Leistungsanalyse
Die zehn wichtigsten Fragen

- Was wollen wir mit der Leistungsanalyse bewerten?
- Welche Ursachen von Leistungsverlusten gibt es?
- Welche pessimistische Abschätzung machte Minsky?
- Welche zwei Maße charakterisieren typischerweise die Qualität einer Parallelisierung?
- Was beschreibt Amdahls Gesetz?
- Welchen Effekt hat ein sequentieller Anteil von n% auf die Leistungsausbeute?
- Wie verändern sich Aufwendungen von Berechnung und Kommunikation bei veränderten Datenstrukturen oder veränderter Hardware?
- Was ist superlinearer Speedup?
- Welche Formen der Parallelisierung widersetzen sich einer eindeutigen Speedup-Ermittlung?
- Wie sehen typische Speedup-Kurven aus für reale Anwendungen aus?

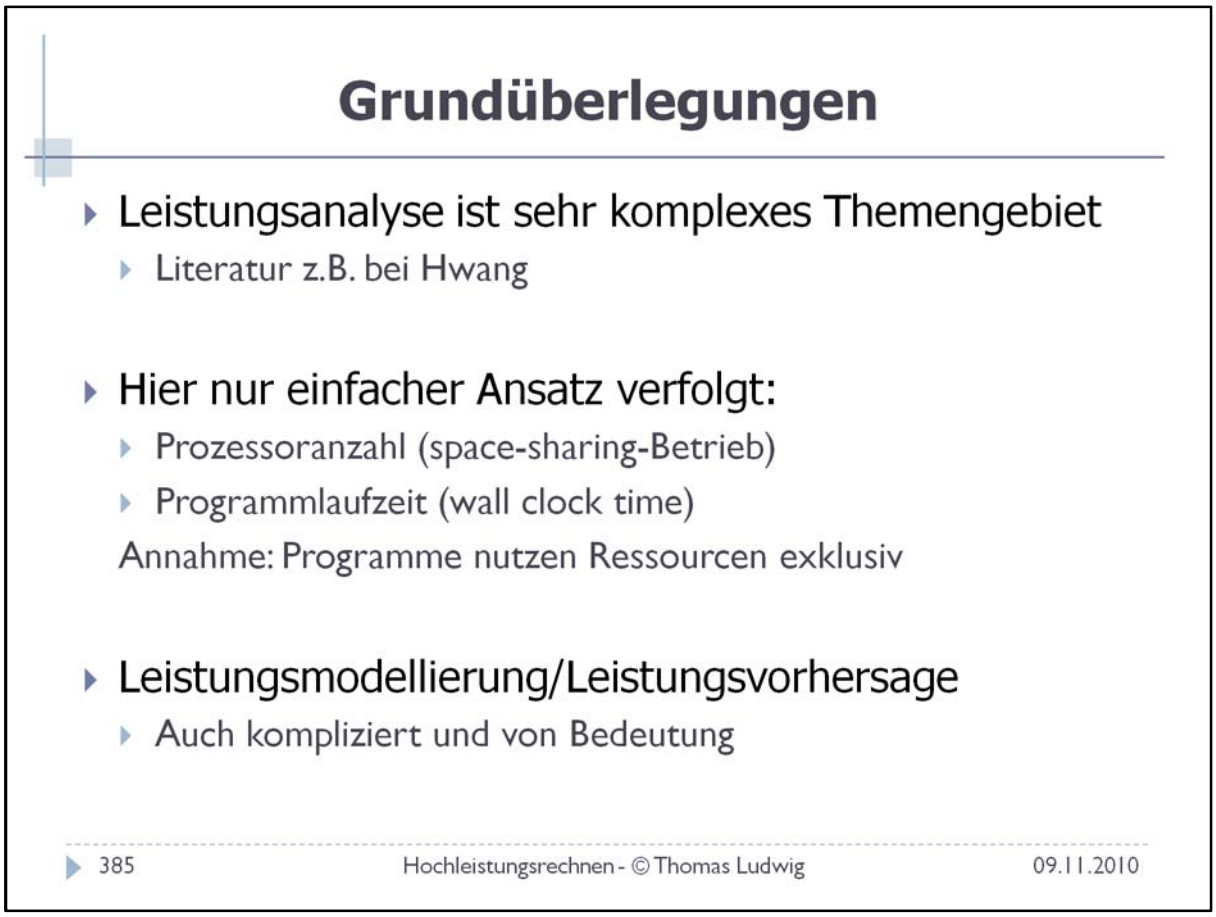
Problemstellung

- Beurteilung der Leistungsfähigkeit des Hochleistungsrechners
  Hardware, Betriebssystem, Compiler
- Beurteilung der Qualität der Parallelisierung
  Programmierbibliotheken, Programme

- Zur Kaufentscheidung
  Wieviel kostet es, wenn mein Programm in Zeit t berechnet sein muß

Ziele der Leistungsanalyse

- Optimierung der Ressourcenauslastung
  Prozessoren, Speicher, Netzwerk, Platten
- Leistungsoptimierung bei exklusiven Ressourcen
  Minimale Laufzeit des Programms
- Verweilzeitoptimierung bei nicht-exklusiver Ressourcennutzung
  Ebenso wichtig: Fairness zwischen den Programmen
- Visualisierung des dynamischen Ablaufs



Space-sharing setzt sich hier vom Time-Sharing ab, das wir auf den einzelnen Rechnern haben: Bei Letzterem wird die Prozessorzeit auf die Prozesse zugeteilt. Die Zuteilung von Rechnerknoten zu einem parallelen Rechenjob eines Benutzers nennt man Space-Sharing, weil hier ein Teil der Maschine dem Benutzer zugewiesen wird.

Siehe: http://en.wikipedia.org/wiki/Time_sharing

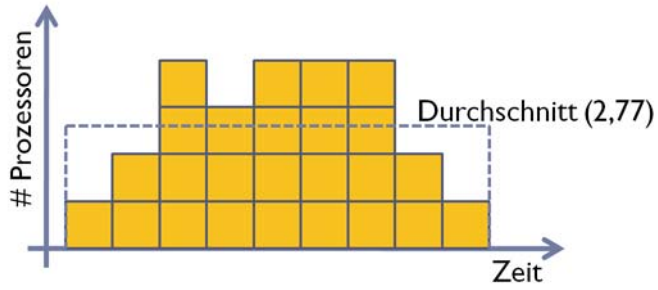
„Wall clock time“ ist die Zeit, die die Uhr an der Wand anzeigt. Also üblicherweise eine Genauigkeit lediglich im Sekundenbereich. Das genügt auch für die parallelen Programme, deren Laufzeiten ja mindestens im Minutenbereich liegen. Man mißt heri die während des Programmlaufs verstrichene Zeit, d.h. wie lange wir auf das Ergebnis warten. Nicht zu verwechseln mit der Prozessorzeit, der Zeit also, die der Prozessor am Programm gearbeitet hat.

Grundüberlegungen...

n Prozessoren, m ist maximaler Parallelismus (m<=n), l ist Leistung

Gesamte geleistete Arbeit $W = l \cdot \sum_{i=1}^{n} i \cdot t_i$

Durchschnittlicher Parallelismus $A = \sum_{i=1}^{m} i \cdot t_i \; / \sum_{i=1}^{m} t_i$



Beachten Sie: diese merkwürdige Summenformel addiert über die y-Achse!

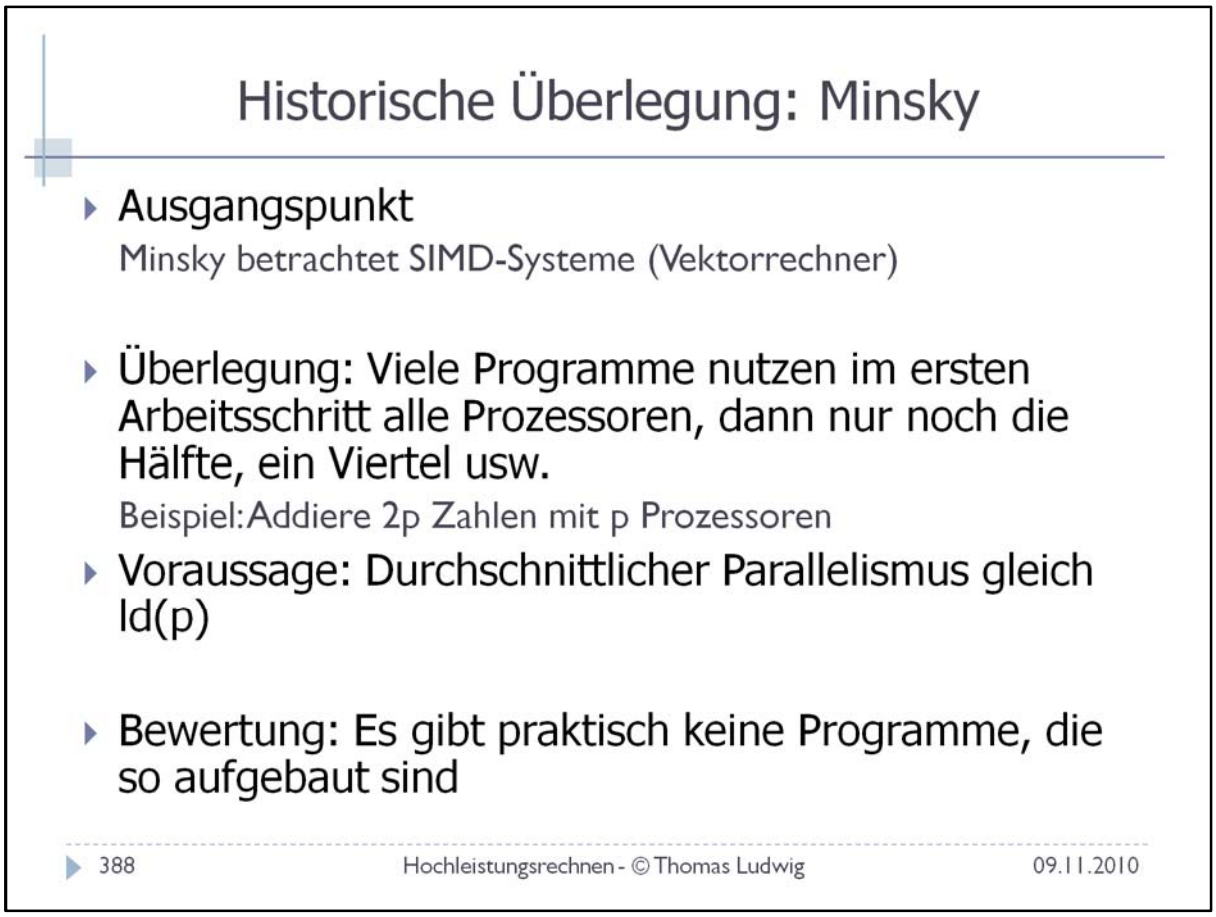
Historische Überlegungen

Anfänglich gab es sehr pessimistische Überlegungen zum Leistungspotential von Parallelrechnern

Es wurde generell das Potential zur Leistungssteigerung kritisch gesehen bzw. verneint

Aus der Menge der Aussagen betrachten wir zwei:

- Minsky, 1971
- Amdahl, 1967



Das Beispiel entspricht dem Prinzip „mit Kanonen auf Spatzen“. So bekommt man aus keinem Rechner hohe Leistung. Die beschriebene Situation darf bestenfalls in der Endphase dieses Additionsvorganges auftreten.

Siehe:
• http://en.wikipedia.org/wiki/Marvin_Minsky
• http://en.wikipedia.org/wiki/The_Society_of_Mind

“What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle”. – Marvin Minsky, *The Society of Mind*, p. 308

Leistungsmaße: Speedup und Effizienz

- Speedup: S = T(1)/T(p)
  T(1) ist die Rechenzeit auf einem Prozessor
  T(p) ist die Rechenzeit auf p Prozessoren
- Forderung
  Wähle jeweils den schnellstmöglichen Algorithmus (kritisch!!)
- Effizient: E = S/p
  Normalisierung nach der Prozessorzahl. E gibt an, wieviel Prozent des maximalen Speedup S=p erreicht werden
- Vermutung: S < p und damit E < 1

Siehe: http://en.wikipedia.org/wiki/Speedup

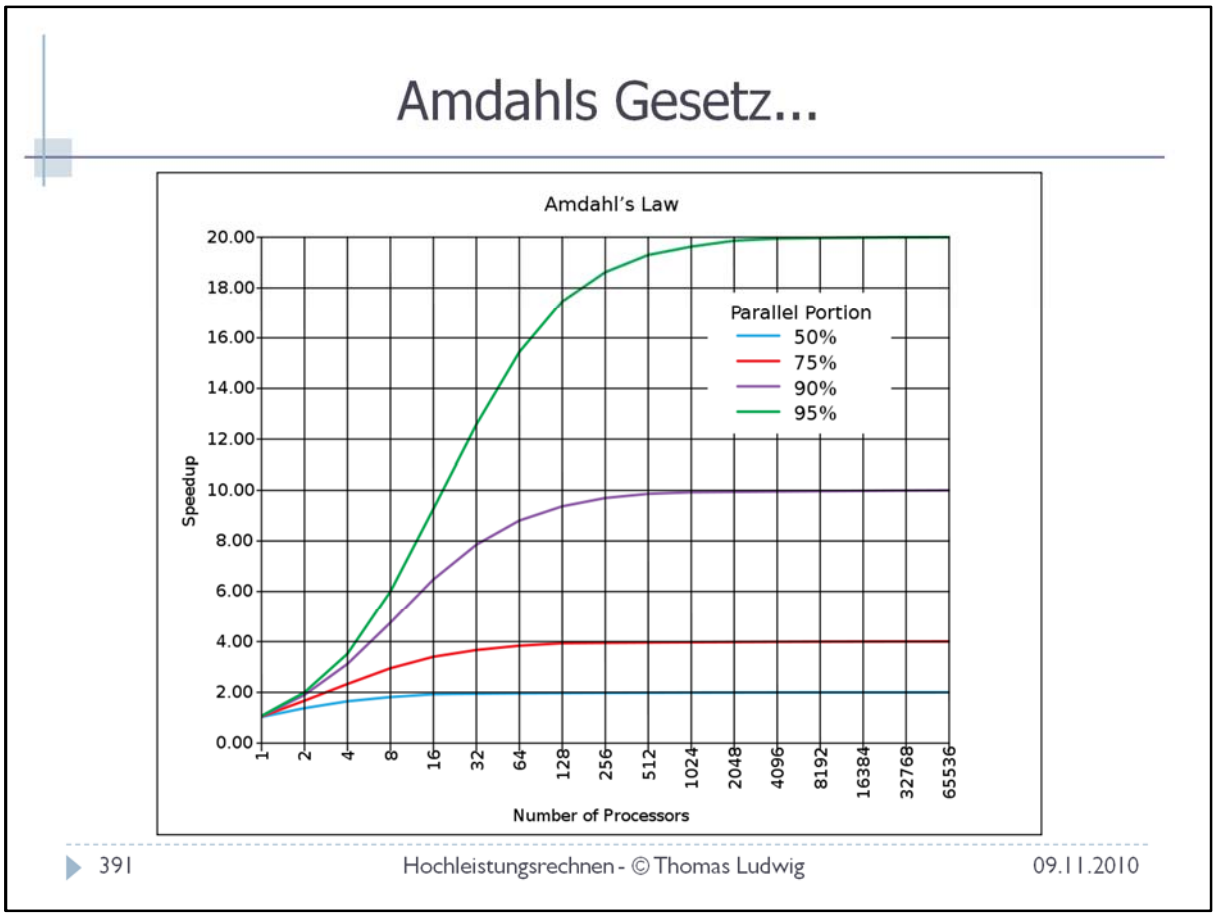
Amdahls Gesetz

- Ausgangspunkt: Jedes Programm enthält einen Bruchteil f an Operationen, die nur sequentiell ausgeführt werden können
- Es gilt daher für den Speedup

  $S <= 1/(f+(1-f)/p) \Rightarrow S_{max} <= 1/f$
- Beispiel: $f=0.01 \Rightarrow S_{max}=100$
- Praktische Erfahrung

  Sequentieller Anteil meist sehr gering
- Bewertung: Amdahls Gesetz gilt! Man muß versuchen, den sequentiellen Anteil klein zu halten

Siehe: http://en.wikipedia.org/wiki/Amdahl%27s_law



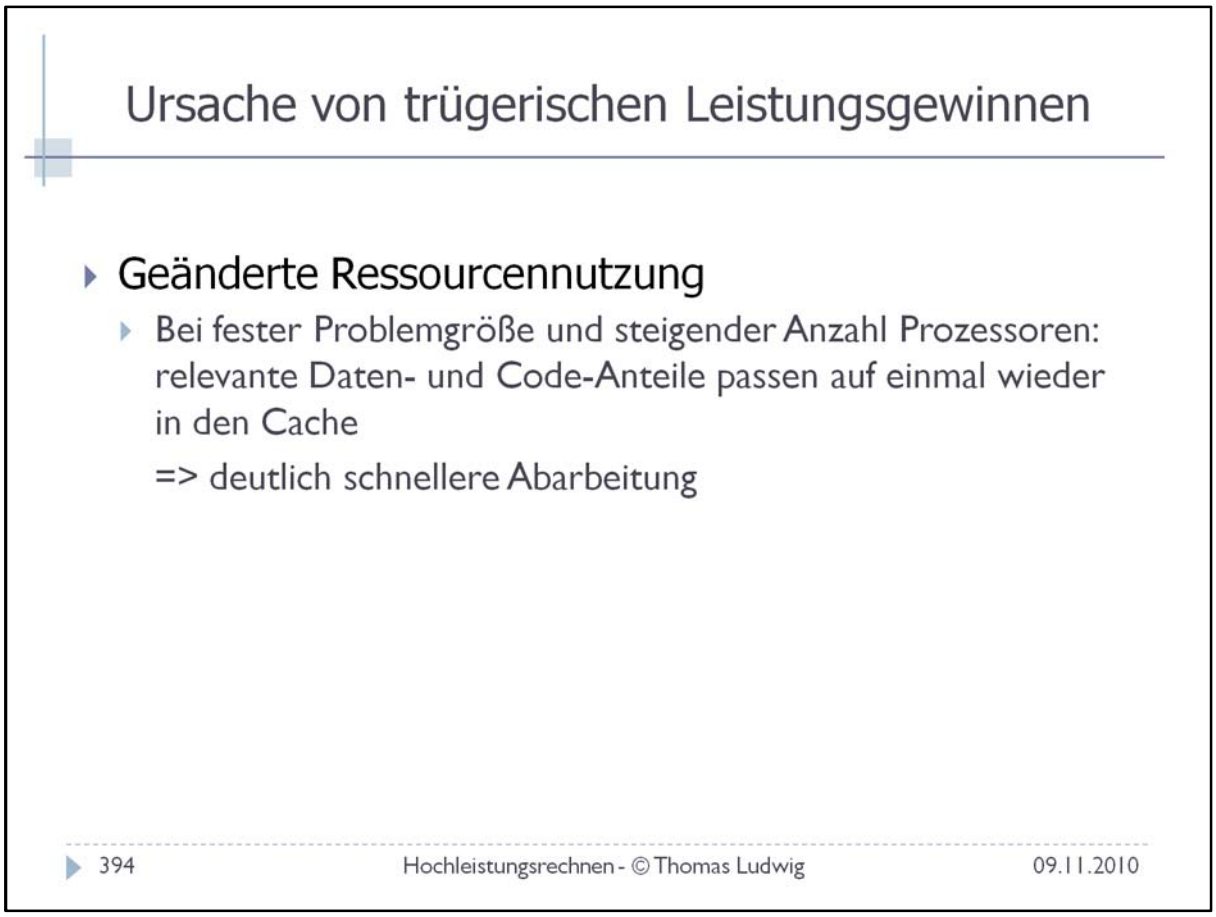
Quelle: Wikimedia Commons (http://en.wikipedia.org/wiki/File:AmdahlsLaw.svg)

Ursachen von Leistungsverlusten

- Zugriffsverluste
  - Bei der Überwindung von Distanzen beim Datenaustausch zwischen Systemkomponenten
    - z.B. Nachrichtenaustausch oder entfernter Speicherzugriff bei NUMA
- Konfliktverluste
  - Bei der gemeinsamen Nutzung von Ressourcen durch mehrere Programmeinheiten
    - z.B. beim Bus- und Netzzugriff
- Auslastungsverluste
  - Bei zu geringem Parallelitätsgrad des Programms
    - z.B. permanente oder zeitweilige Lastungleichheit

Ursachen von Leistungsverlusten...

- Bremsverluste
  - Beim Beenden gewisser Berechnungen, wenn bereits eine Lösung gefunden wurde
    - z.B. Suchbaumverfahren
- Komplexitätsverluste
  - Durch Zusatzaufwand im parallelen Programm gegenüber dem sequentiellen
    - z.B. Aufteilung der Daten
- Wegwerfverluste
  - Ergebnis mehrfach berechnet, aber nur eines von ihnen weiterbenutzt
    - z.B. Doppelberechnungen bei globalen Operationen



Möglicherweise entsteht hier superlinearer Speedup. Das ist dann leicht zu erkennen. Aber meistens wird einfach nur die Kurve nach oben gedrückt, was sich eigentlich erstmal gar nicht erkennen läßt. Es sieht dann womöglich aus wie guter Speedup.

Einfluß der Problemgröße

Wir wollen nun betrachten, wie sich in verschiedenen Situationen die Aufwände für Kommunikation und Berechnung zueinander verhalten

Schwache Skalierung (weak scaling)

- Bei schwacher Skalierung behalten wir die Problemgröße **pro Prozessor** bei und erhöhen die Anzahl der Prozessoren. Die Problemgröße nimmt dabei also zu!

Starke Skalierung (strong scaling)

- Bei starker Skalierung behalten wir die gesamte Problemgröße bei und erhöhen die Anzahl der Prozessoren. Jeder hat dann zunehmend weniger zu tun – dies wird bei der Speedupbestimmung zugrunde gelegt.

Einfluß der Problemgröße...

Beispiel: Berechnung einer Matrix

- Größe: 1000 x 1000 Elemente
- Betrachtung eines Iterationsschrittes
- Berechnung dauert 1000 x 1000 x 1 Zeiteinheit

Parallelisierung mit p=5

- Aufteilung in Blöcke von Zeilen
- Berechnung: 200 x 1000 x 1 Zeiteinheit
- Kommunikation: benachbarte Blöcke tauschen eine Zeile aus (alle gleichzeitig)
  Aufwand: 2 x 1000 x 1 Zeiteinheit
- Kommunikations/Berechnungs-Verhältnis: 1/100

Einfluß der Problemgröße...

Wir verwenden mehr Prozessoren

Parallelisierung mit p=10

- Aufteilung in Blöcke von Zeilen
- Berechnung: 100 x 1000 x 1 Zeiteinheit
- Kommunikation: benachbarte Blöcke tauschen eine Zeile aus (alle gleichzeitig)
  Aufwand: 2 x 1000 x 1 Zeiteinheit
  Kommunikations/Berechnungs-Verhältnis: 1/50

Parallelisierung mit 100 Prozessoren

- Kommunikations/Berechnungsverhältnis: 1/5

Einfluß der Problemgröße...

Wir vergrößern das Problem

Parallelisierung mit p=10 und doppelt so großer Matrix

- Berechnung: 141 x 1414 x 1 Zeiteinheit
- Kommunikation: 2 x 1414 x 1 Zeiteinheit
- Kommunikations/Berechnungs-Verhältnis: 1/70
  (p=5 und Matrix 1000x1000: 1/100)

Trotz doppelter Prozessoranzahl und doppelter Problemgröße verschlechtert sich das K/B-Verhältnis

Einfluß der Problemgröße...

Wir kaufen neue Prozessoren

Parallelisierung mit p=10 und doppelt so großer Matrix

- Berechnung: 141 x 1414 x **0.3** Zeiteinheiten
- Kommunikation: 2 x 1414 x 1 Zeiteinheit
- Kommunikations/Berechnungs-Verhältnis: 1/21

Superlinearer Speedup

Aufgabenstellung

- n Sortieralgorithmen mit Laufzeiten ti
- Finde den besten Sortieralgorithmus

Sequentielles Programm

- Lasse den ersten Algorithmus laufen; liefert $t_{min}=t_1$
- Starte nächsten Algorithmus; wenn er t_{min} überschreitet, dann sofort stoppen; andernfalls $t_{min}=t_j$
- Wiederhole mit allen Algorithmen
- Zeitbedarf $t_{seq} >= t_{min} \times n$

Superlinearer Speedup...

Paralleles Programm

- Nimm n Prozessoren
- Starte n Algorithmen gleichzeitig auf den Prozessoren
- Stoppe die Berechnung, wenn der erste fertig ist
- Zeitbedarf: $t_{par} = t_{min}$

Erzielter Speedup: $S=t_{seq}/t_{par}$ => $S>=n$!?

Perpetuum mobile des parallelen Rechnens?

Superlinearer Speedup...

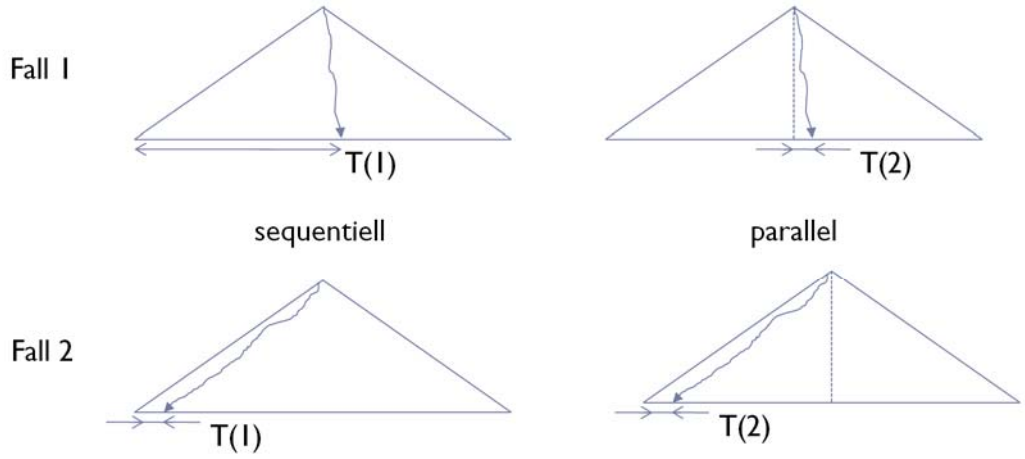
Analyse des Beispiels:

- Sequentielles Programm ist nicht „das optimale“ sequentielle Programm
- Speedup-Definition fordert „optimales“ Programm
- „Optimal“:hier Quasiparallele Abarbeitung des Algorithmenvergleich auf einem Prozessor
- Damit verschwindet dann der superlineare Speedup

Superlinearer Speedup...

Suchbaum-Verfahren sequentiell/parallel

- Wir suchen **eine** Lösung



403 Hochleistungsrechnen - © Thomas Ludwig 09.11.2010

Wir verwenden hier Tiefensuche. Das Verfahren findet an den Blättern des Baumes (horizontale Linie) die erste Lösung im Fall 1 eher spät, im Fall 2 sehr früh.

Superlinearer Speedup...

Analyse des Beispiels:

- Für breite Bäume geht im Fall a) der Speedup gegen unendlich
- Für breite Bäume geht im Fall b) der Speedup gegen eins

Angabe des Speedup sinnlos!
Parallelisierung ebenso?

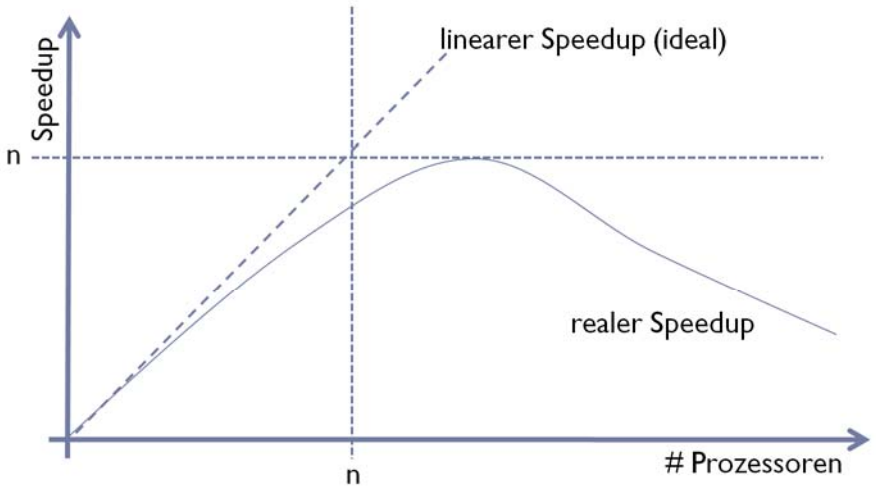
Bei Suche nach allen Lösungen problemlos!

- Hier verschwindet das Problem

Der Anwendungsfall findet sich leider häufig bei nichtnumerischen Verfahren, die Baumsuche verwenden: diskrete Optimierung, Formelbeweiser usw.

Speedup-Bestimmung

Verwendet in wissenschaftlichen Artikeln, in denen die Verfahren parallelisiert wurden



Speedup-Bestimmung...

Speedup-Kurven angegeben für festen Datensatz

Kurvenverlauf

- Bei guten numerischen Verfahren

  S nahe bei p für alle p, die wir einsetzen

  Effizienz zwischen 80% und 100%

- Bei schlechten Verfahren

  Effizienz von 50% oder schlechter

Was tun, wenn die Kurven nicht optimal aussehen?

Speedup-Bestimmung...

David Bailey

Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers

Supercomputer Review, August 1991

Liste sehr beliebter (Selbst-)Täuschungsmethoden zur Erzielung besserer Ergebnisse

Siehe: http://crd.lbl.gov/~dhbailey/dhbpapers/twelve-ways.pdf

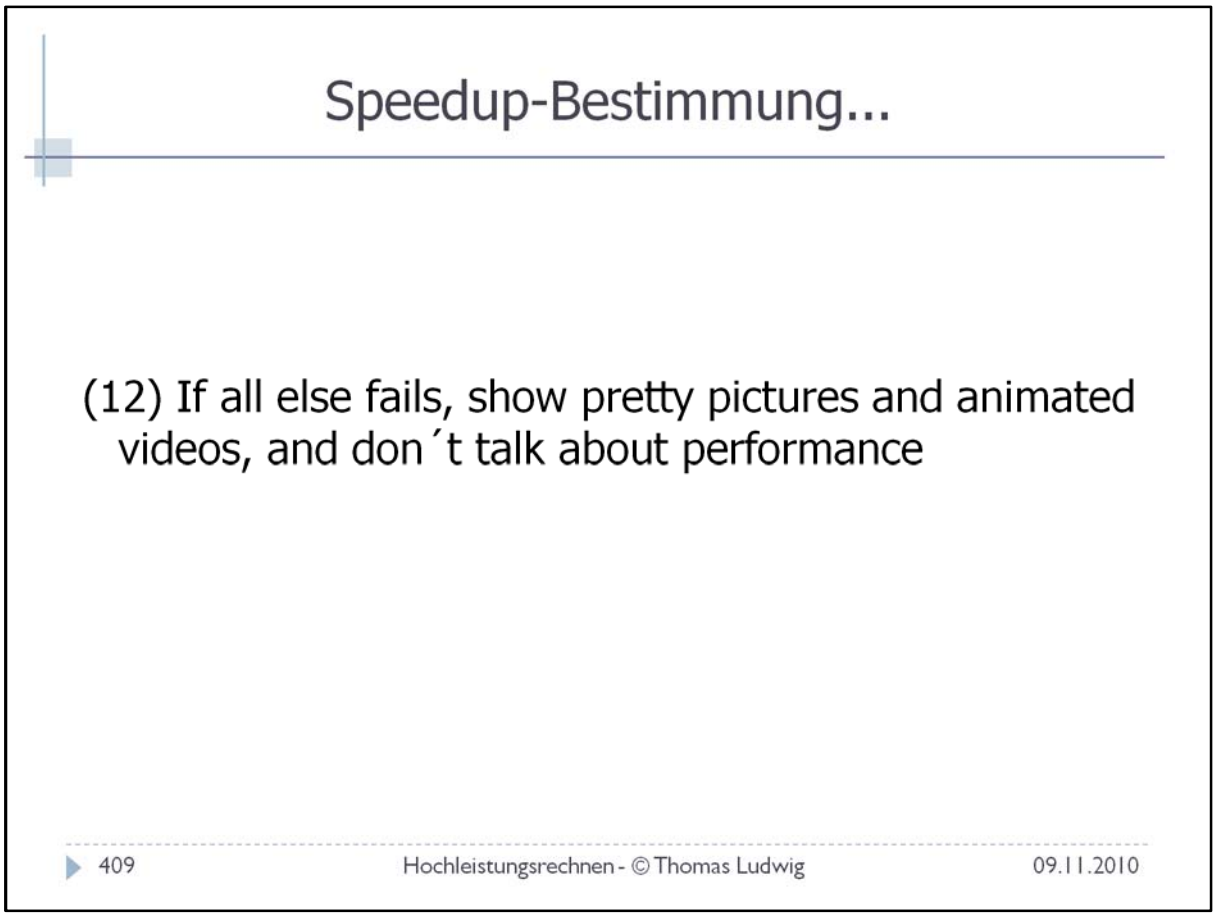
Speedup-Bestimmung...

(2) Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application

(4) Scale up the problem size with the number of processors, but omit any mention of this fact

(5) Quote performance results projected to a full system

(7) When direct run time comparisons are required, compare with an old code on an obsolete system



„It sometimes happens that the audience starts to ask all sorts of embarrassing questions.
These people simply have no respect for the authorities of our field. If you are so unfortunate as to be the object of such disrespect, there is always a way out --- simply conclude your technical presentation and roll the videotape. Audiences love razzle-dazzle
color graphics, and this material often helps deflect attention from the substantive technical issues.“ David Bailey

Leistungsanalyse
Zusammenfassung

- Leistungsanalyse dient der Rechner- und Programm-bewertung
- Es gibt unterschiedliche Ursachen für Leistungsverluste
- Speedup und Effizienz charakterisieren die Qualität des parallelen Programms
- Amdahl: der Speedup ist nach oben begrenzt
- Problemgröße beeinflußt den erreichbaren Speedup
- Superlinearer Speedup ist nicht systematisch nutzbar
- Es gibt viele Möglichkeiten für irrtümlich oder absichtlich falsche Speedup- bzw. Leistungs-Angaben