
Rechnerarchitektur SS 2019

Amdahl's Law and Performance Measures

Jian-Jia Chen

TU Dortmund

Teilweise basierend auf Material von Michael Engel, Gernot A. Fink und
R. Yahyapour

02 April 2019

Mehrprozessorsysteme

Amdahl'sche Gesetz

Mehrprozessorsysteme

“The turning away from the conventional organization came in the middle 1960s, when the law of diminishing returns began to take effect in the effort to increase the operational speed of a computer ... Electronic circuits are ultimately limited in their speed of operation by the speed of light ... and many ... were already operating in the nanosecond range.”

W. Jack Bouknight et al.: The Illiac IV System (1972)

“... sequential computers are approaching a fundamental physical limit on their potential power. Such a limit is the speed of light ...”

Angel L. DeCegama: The Technology of Parallel Processing (1989)

Mehrprozessorsysteme: Einführung

Beobachtung: Einschätzung, dass Fortschritte bei Uni-Prozessor-Architekturen sich ihrem Ende nähern, wurde verschiedentlich und zu verschiedenen Zeiten vertreten.

Allerdings: Allerdings: 1985-2000 höchste Leistungssteigerung bei Uni-Prozessoren zu beobachten (seit d. Einführung transistorbasierter Rechner Ende der 1950er)

⇒ Triebkraft: Verwendung/Technologie von Mikroprozessoren

Trotzdem: Bedeutung paralleler Prozessoren wächst und wird in der Zukunft weiter steigen

Gründe für wachsende Bedeutung von MPs

- Mikroprozessoren werden vorherrschende Technologie für Uni-Prozessoren bleiben
 - ⇒ Weg zur Leistungssteigerung: Kombination mehrerer Prozessoren (seit einiger Zeit auch on-chip, z.B. Dual/Quad-Core-Prozessoren, MPSoCs)
- **Annahme:** Kosteneffektiver als Spezialprozessoren
- Stetiger Fortschritt bei größter Hürde für MP-Einsatz: *Software*
 - Groß bei Server-Anwendungen (haben häufig “natürlichen” Parallelismus)
 - Problematischer bei Desktop-Systemen/Anwendungen
 - Ex. erste weiter verbreitete Programmiersysteme (z.B. OpenMP)

Motivation der Themenbehandlung

- Bereits seit 2000 Abschwächung der Leistungssteigerung bei Uni-Prozessoren bemerkbar!

Trotzdem: Steigerung der Integrationsdichte folgt (vorerst?) ungebrochen dem Moore'schen Gesetz ... betrifft aber mehr und mehr Integration *weiterer Komponenten* on-chip!

- Vorstoss von MPs in breitere Märkte vollzogen (im Desktop/Server-Bereich: Multi-Core-Prozessoren)
- Parallelisierung *immer schon* wichtigste algorithmische Triebkraft bei der Leistungssteigerung: MPs wenden Prinzip nur auf anderer Granularitätsstufe an (i. Vgl. zu Pipelining)

⇒ Mehrprozessorsysteme werden in Zukunft weiter an Attraktivität gewinnen!

Taxonomie von Mehrprozessorsystemen

Idee der Leistungssteigerung durch Paralleles Rechnen bereits i.d. Anfängen der Rechnerentwicklung verfolgt

Taxonomie möglicher Architekturen nach Flynn (1966)
(betrachtet Parallelismus im Befehls- und Datenstrom)

Single instruction, single data (SISD)

- Entspricht "klassischem/normalem" Uni-Prozessor-System

Single instruction, multiple data (SIMD)

- Gleicher Befehl(-sstrom) wird von mehreren Prozessoren auf unterschiedlichen Daten(strömen) ausgeführt
- Eigene Datenspeicher pro Prozessor, aber gemeinsamer Befehlsspeicher und Kontrolleinheit (zum Holen/Ausgeben der Befehle)

Beispiel: Vektorrechner

Vergleichbar: Multimedia-Befehle $\hat{=}$ eingeschränkte Variante

Single instruction, single data (SISD) ...

Single instruction, multiple data (SIMD) ...

Multiple instruction, single data (MISD)

- Ex. Mehrere Verarbeitungsknoten, in Matrix angeordnet, bearbeiten verschiedene Befehlsströme
- Ein Datenstrom durchläuft Berechnungsmatrix
- Bezeichnet als systolisches Array

Multiple instruction, multiple data (MIMD)

- Jeder Prozessor bearbeitet eigenen Befehls- und eigenen Datenstrom
- Verwendung "normaler" Mikroprozessoren möglich!
- Verbreitetstes Architektur-Schema paralleler Rechner

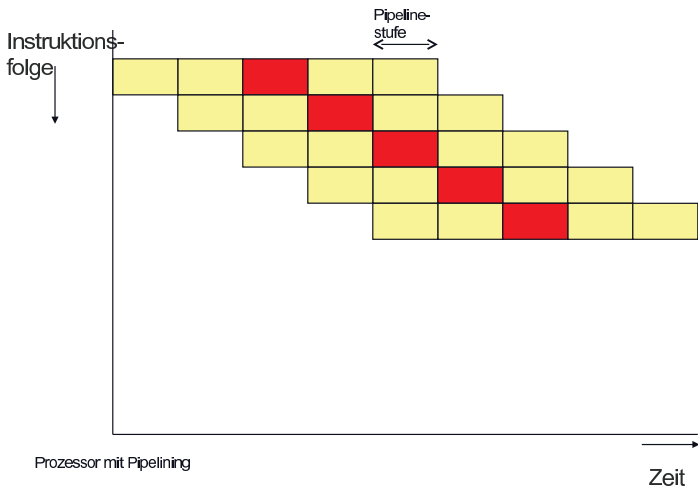
Taxonomie von Mehrprozessorsystemen: Diskussion

- Grobe Kategorisierung \Rightarrow Ex. "hybride" Formen
- SIMD und MISD nur für spezielle Berechnungen geeignet (\Rightarrow keine kommerziellen MISD-Systeme)
- Viele frühe MPs nach SIMD-Prinzip, seit 1990er praktisch verschwunden (Ausnahme: Vektorrechner)
- MIMD ist vorherrschende Architektur für "*general purpose*" Mehrprozessorsysteme
 - Große Flexibilität (gesamtes MP für eine(n) Anwendung/Benutzer bis parallele Bearbeitung vieler unterschiedlicher Tasks/Prozesse)
 - Können von Kosten-Leistungs-Vorteilen gängiger Mikroprozessoren profitieren
 - \Rightarrow praktisch alle heutigen MIMD-Systeme mit gleichen Mikroprozessoren wie Server/Desktops realisiert!

Welche Arten der Parallelität kennen wir?

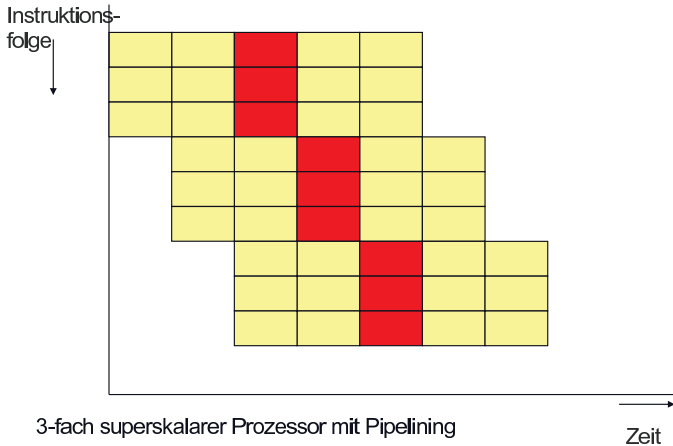
Parallelität: Pipelining

Quelle: R. Yahyapour, ehem. TU Dortmund



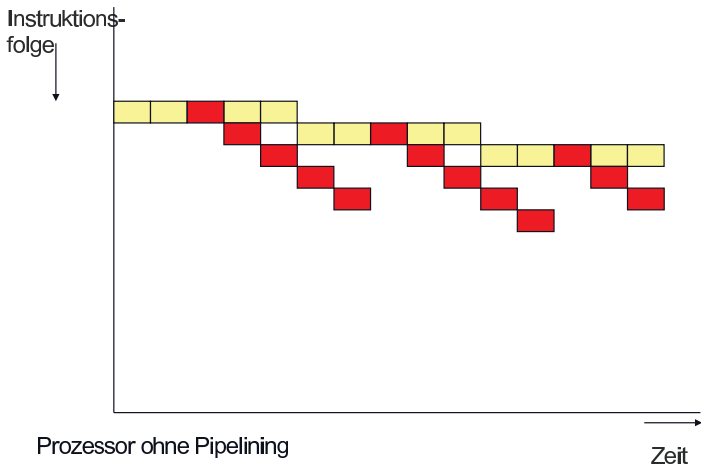
Parallelität: Superskalarität

Quelle: R. Yahyapour, ehem. TU Dortmund



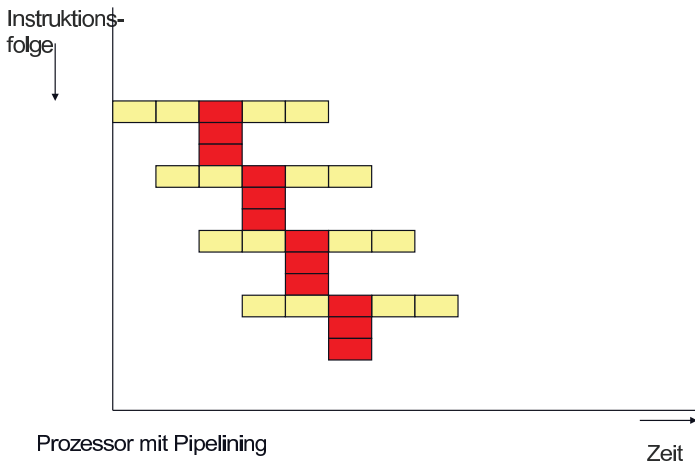
Parallelität: Vektorprozessoren

Quelle: R. Yahyapour, ehem. TU Dortmund



Parallelität: Very Long Instruction Word (VLIW)

Quelle: R. Yahyapour, ehem. TU Dortmund



Mehrprozessorsysteme

Amdahl'sche Gesetz

Herausforderung der Parallelverarbeitung

Zwei wesentliche Herausforderungen ex. beim Versuch, Parallelverarbeitung zur Leistungssteigerung einzusetzen:

1. Leistungssteigerung durch zusätzliche Prozessoren
2. Kosten der notwendigen Kommunikation

Fragen:

- *Welchen Leistungsgewinn erzielt ein N-Prozessor-System gegenüber einem Uni-Prozessor?*

... im allgemeinen?? ... für eine best. Aufgabe?

- Wieviel Overhead entsteht durch Datenaustausch?

... wieso eigentlich Datenaustausch?

Gesetze von Amdahl und Gustafson machen darüber Aussagen!

Das Amdahl'sche Gesetz

Welchen Leistungsgewinn erzielt ein N-Prozessor-System gegenüber einem Uni-Prozessor?

Speedup $S_{\text{tot}} = \frac{\text{Laufzeit ohne Parallelisierung}}{\text{Laufzeit mit Parallelisierung}}$

Nach Amdahl'schem Gesetz abhängig von:

- Anteil P des parallelisierbaren Codes des betreffenden Programms / der Anwendung
- Speedup S , der durch paralleles Rechnen erreicht wird (vereinfachende Annahme: Parallelisierung auf diesem Code [Anteil P] perfekt \Rightarrow Speedup $S = N = \text{Anz. der Prozessoren}$)

$$S_{\text{tot}} = \frac{1}{(1 - P) + \frac{P}{S}} = \frac{1}{(1 - P) + \frac{P}{N}}$$

Das Amdahl'sche Gesetz II

$$S_{\text{tot}} = \frac{1}{(1 - P) + \frac{P}{S}} = \frac{1}{(1 - P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
 - 80% parallelisierbarer Code \Rightarrow Speedup ca. 4,7
 - 50% parallelisierbarer Code \Rightarrow Speedup ca. 2
 - 20% parallelisierbarer Code \Rightarrow Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \rightarrow \infty} S_{\text{tot}} = \frac{1}{(1 - P)}$$

Das Amdahl'sche Gesetz II

$$S_{\text{tot}} = \frac{1}{(1 - P) + \frac{P}{S}} = \frac{1}{(1 - P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
 - 80% parallelisierbarer Code \Rightarrow Speedup ca. 4,7
 - 50% parallelisierbarer Code \Rightarrow Speedup ca. 2
 - 20% parallelisierbarer Code \Rightarrow Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \rightarrow \infty} S_{\text{tot}} = \frac{1}{(1 - P)}$$

Allein vom sequentiellen Programmanteil abhängig!

$$S_{\text{tot}} = \frac{1}{(1 - P) + \frac{P}{S}} = \frac{1}{(1 - P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
 - 80% parallelisierbarer Code \Rightarrow Speedup ca. 4,7
 - 50% parallelisierbarer Code \Rightarrow Speedup ca. 2
 - 20% parallelisierbarer Code \Rightarrow Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \rightarrow \infty} S_{\text{tot}} = \frac{1}{(1 - P)}$$

Allein vom sequentiellen Programmanteil abhängig!



Max. Speedup bei 80% parallel. Code: 5!

(d.h. 128 statt 64 Prozessoren kaum Gewinn [4,8 statt 4,7])

Das Gesetz von Gustafson (& Barsis)

... stellt quasi Gegenstandspunkt zum Amdahl'schen Gesetz dar:

$$S'_{\text{tot}} = N - (1 - P) \cdot (N - 1) = N - N + 1 + P \cdot (N - 1) = (1 - P) + P \cdot N$$

- Betrachten Beispiel (wieder jeweils $N = 64$ Prozessoren):
 - 80% parallelisierbarer Code \Rightarrow Speedup ca. 51,4
 - 50% parallelisierbarer Code \Rightarrow Speedup ca. 32,5
 - 20% parallelisierbarer Code \Rightarrow Speedup ca. 13,6

Wieso dieser (scheinbare) Widerspruch zum Amdahl'schen Gesetz?

Das Gesetz von Gustafson (& Barsis)

... stellt quasi Gegenstandspunkt zum Amdahl'schen Gesetz dar:

$$S'_{\text{tot}} = N - (1 - P) \cdot (N - 1) = N - N + 1 + P \cdot (N - 1) = (1 - P) + P \cdot N$$

- Betrachten Beispiel (wieder jeweils $N = 64$ Prozessoren):
 - 80% parallelisierbarer Code \Rightarrow Speedup ca. 51,4
 - 50% parallelisierbarer Code \Rightarrow Speedup ca. 32,5
 - 20% parallelisierbarer Code \Rightarrow Speedup ca. 13,6

Wieso dieser (scheinbare) Widerspruch zum Amdahl'schen Gesetz?

Annahme: Problemgröße wächst linear mit der Anzahl d. Prozessoren!

$$S'_{\text{tot}} = \frac{(1 - P) + P \cdot N}{(1 - P) + P}$$

Amdahl vs. Gustafson

Eine automobilistische Metapher:

(Quelle: Wikipedia)

Amdahl's Law approximately suggests:

"Suppose a car is traveling between two cities 60 miles apart, and has already spent one hour traveling half the distance at 30 mph. No matter how fast you drive the last half, it is impossible to achieve 90 mph average before reaching the second city. Since it has already taken you 1 hour and you only have a distance of 60 miles total; going infinitely fast you would only achieve 60 mph."

Gustafson's Law approximately states:

"Suppose a car has already been traveling for some time at less than 90mph. Given enough time and distance to travel, the car's average speed can always eventually reach 90mph, no matter how long or how slowly it has already traveled. For example, if the car spent one hour at 30 mph, it could achieve this by driving at 120 mph for two additional hours, or at 150 mph for an hour, and so on."

Problem:

- Ermittlung des Speedup erfordert immer die Betrachtung eines Programms/einer Anwendung
- ⚡ Die theoretische “Peak Performance” ist häufig weit entfernt von der realistischen Leistung eines Systems

Lösungsansatz: Benchmarks für die Bewertung der Rechenleistung

Beispiele:

- *NAS Parallel Benchmarks* (<http://www.nas.nasa.gov/>)
“... set of programs designed to help evaluate the performance of parallel supercomputers. ... derived from computational fluid dynamics (CFD) applications and consist of five kernels ...”
- LINPACK (<http://www.netlib.org/benchmark/hpl/>)
Ursprünglich numerische Programmbibliothek zum Lösen von lin. Gleichungssystemen



Achtung: Einfluss von Programmierung, Compiler, ...