

## 6. Analytische Bewertung paralleler Systeme

1. Parallele Systeme
2. Leistungsmaße für parallele Systeme
  - Speedup, Effizienz, Kosten, Isoeffizienz
3. Historisch: Gesetze von Amdahl und Gustafson-Barsis

## 6. Analytische Bewertung paralleler Systeme

1. Parallele Systeme
2. Leistungsmaße für parallele Systeme
  - Speedup, Effizienz, Kosten, Isoeffizienz
3. Historisch: Gesetze von Amdahl und Gustafson-Barsis

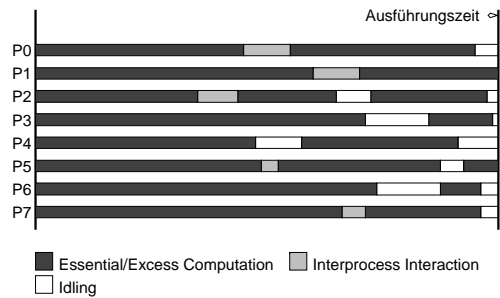
## Parallele Systeme

- Ein **paralleles System** besteht aus einem parallelem Algorithmus und einer Parallelrechnerarchitektur auf der der Algorithmus ausgeführt wird.
  - Die Ausführungszeit eines parallelen Programms hängt wesentlich von der Anzahl der Prozessoren und der Leistungsfähigkeit des Verbindungsnetzwerks ab.
  - Konsequenz: Parallele Programme können nicht isoliert von einer konkreten Parallelrechnerarchitektur quantitativ untersucht werden.
- **Ziel:** Quantitative Beschreibung der Gesamtleistung eines parallelen Systems.

## Overhead in parallelen Systemen

- **Interprocess Interactions**
  - Interaktionen zwischen den Prozessen (Kommunikation, Synchronisation)
- **Excess Computation**
  - Zusätzlicher Programmcode zur Realisierung von Parallelität
  - Paralleler Algorithmus beruht nicht auf dem besten sequentiellen Verfahren
- **Process Idling**
  - Synchronisation zwischen den Prozessen
  - Ungleichmäßige Lastverteilung
  - Sequentielle Abschnitte im parallelen Algorithmus

## Overhead in parallelen Systemen



## 6. Analytische Bewertung paralleler Systeme

1. Parallele Systeme
2. Leistungsmaße für parallele Systeme
  - Speedup, Effizienz, Kosten, Isoeffizienz
3. Historisch: Gesetze von Amdahl und Gustafson-Barsis

## Laufzeit



- Die **sequentielle Laufzeit**  $T_s$  eines Programms ist die Zeit, die zwischen dem Programmstart und dem Programmende bei der Ausführung auf einem sequentiellen Rechner verstreicht.
- Die **parallele Laufzeit**  $T_p$  ist die Zeit zwischen dem Start und dem Ende der parallelen Programmausführung auf  $p$  Prozessoren.
- Es wird grundsätzlich die tatsächlich verstrichene Zeit (**Wall-Clock Time**) betrachtet, nicht die CPU Zeit.

## Absoluter paralleler Overhead



- **Definition absoluter paralleler Overhead:**  $T_o = p T_p - T_s$
- Der absolute parallele Overhead  $T_o$  eines parallelen Systems ist die zusätzliche Zeit gegenüber der Laufzeit des besten sequentiellen Algorithmus, die alle Prozessoren zusammengenommen zur parallelen Berechnung benötigen.

## Speedup (Beschleunigung)

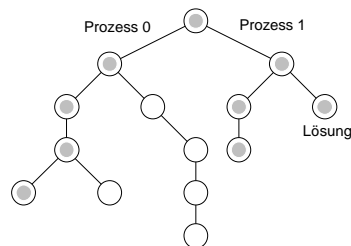
- **Speedup  $S$**  ist ein wichtiges Maß für den Nutzen der Parallelisierung eines Problems gegenüber der sequentiellen Programmausführung.
- **Definition Speedup:  $S := T_S / T_P$** 
  - $T_S$ : Sequentielle Laufzeit des besten bekannten Algorithmus für das Problem auf einem Prozessor.
  - $T_P$ : Parallele Laufzeit des betrachteten parallelen Algorithmus zur Lösung des Problems auf  $p$  Prozessoren.
- Voraussetzungen:
  - Der verwendete Parallelrechner ist aus identischen Prozessoren aufgebaut.
  - Die sequentielle Laufzeit wird auf einem der Prozessoren bestimmt.

## Superlineare Speedups

- In einigen Situationen können **superlineare Speedups** ( $S > p$ ) beobachtet werden.
- Mögliche Ursachen:
  - **Cache-Effekte**
    - Durch Verteilung der Daten auf mehrere Prozesse arbeitet ein einzelner Prozess auf einer kleineren Datenmenge.
    - Bei kleinerer Datenmenge kann sich die Cache-Hit Rate erhöhen; es ergibt sich dann eine schnellere Befehlsabarbeitung.
  - **Algorithmische Effekte bei paralleler Suche**
    - Ein paralleler Such-Task startet seine Suche „sehr dicht“ bei einer Lösung.
    - Die Arbeit zur Berechnung einer Lösung kann dann im parallelen Fall kleiner sein, als im sequentiellen Fall.

## Superlineare Speedups bei paralleler Depth-First Baumsuche

- Sequentielle Laufzeit: 14 Schritte
- Parallele Laufzeit: 5 Schritte, parallele Arbeit: 9 Schritte
- Speedup:  $14/5 = 2.8$



## Effizienz

- Die **Effizienz  $E$**  eines parallelen Systems gibt den Anteil der Laufzeit an, in dem die Prozesse keine Overhead-Instruktionen ausführen.
- **Definition Effizienz:  $E := S / p$** 
  - $S$ : Speedup
  - $p$ : Anzahl der Prozessoren
- In einem idealen parallelen System beträgt die Effizienz 1 (bzw. 100%).

## Kosten und Problemgröße



- Die **Kosten**  $C_p$  eines parallelen Systems beschreiben die gesamte Rechenzeit der einzelnen Prozessoren.  
 $C_p := p T_p$
- Die **Kosten**  $C_s$  zur sequentiellen Lösung des Problems entsprechen der Ausführungszeit des besten sequentiellen Algorithmus auf einem Prozessor.  
 $C_s := T_s$
- Ein paralleles System hat **optimale Kosten**, falls  $C_s(n)$  und  $C_p(n)$  ( $n$  = Größe der Eingabe) asymptotisch gleich sind.
- Die **Problemgröße**  $W$  ist definiert als die sequentiellen Kosten  $C_s$ .  
 $W := C_s = T_s$

## Skalierbarkeit paralleler Systeme



- Ein paralleles System heißt **skalierbar**, falls sich seine Effizienz nicht verschlechtert, wenn die Anzahl der Prozessoren und die Problemgröße (gleichzeitig) erhöht werden.
  - Anzahl der Prozessoren und Problemgröße müssen nicht im selben Verhältnis erhöht werden.
- Ein paralleles System ist **gut skalierbar**, wenn relativ kleine Erhöhungen der Problemgröße ausreichen, um eine größere Anzahl von Prozessoren effizient einsetzen zu können.
  - Die **Isoeffizienzfunktion**  $W(p)$  quantifiziert diese Aussage.

## Motivation für die gegebene Definition der Skalierbarkeit



- Es gilt:  $E = \frac{s}{p} = \frac{T_s}{pT_p} = \frac{1}{1 + \frac{T_o}{W}}$  mit  $T_o = pT_p - T_s$  und  $T_s = W$
- Der absolute parallele Overhead  $T_o$  wächst mit zunehmender Prozessorzahl  $p$  bei konstanter Problemgröße.
  - Typischerweise enthält jedes parallele Programm eine sequentielle Komponente, z.B. Einlesen der Eingabe oder Zugriff auf gemeinsame Variablen.
  - Sei  $t_{ser}$  die Ausführungszeit der sequentiellen Komponente.
  - Dann ist  $(p - 1) t_{ser}$  eine untere Schranke für  $T_o$ .
- **Folgerung:** Die Effizienz  $E$  nimmt bei konstanter Problemgröße  $W$  mit zunehmender Prozessorzahl  $p$  ab.

## Isoeffizienzfunktion



- Es gilt:  $E = \frac{s}{p} = \frac{T_s}{pT_p} = \frac{1}{1 + \frac{T_o(W,p)}{W}}$
- Daraus folgt:  $W = \frac{E}{1-E} T_o(W, p)$
- Bei einem skalierbaren parallelen System ist  $K := \frac{E}{1-E}$  für geeignetes  $W$  und  $p$  konstant.
- Die **Isoeffizienzfunktion**  $W(p)$  eines skalierbaren parallelen Systems ergibt sich dann bei gegebenem  $k$  (bzw.  $E$ ) aus  $W(p) = k T_o(W, p)$ 
  - Die Isoeffizienzfunktion zeigt das Maß der Skalierbarkeit eines parallelen Systems an.
  - Oft kann kein geschlossener Ausdruck angegeben werden.

## 6. Analytische Bewertung paralleler Systeme

1. Parallele Systeme
2. Leistungsmaße für parallele Systeme
  - Speedup, Effizienz, Kosten, Isoeffizienz
3. Historisch: Gesetze von Amdahl und Gustafson-Barsis

## Gesetz von Amdahl (1967)

- **Vereinfachtes Modell:**
  - $T_1$ : Ausführungszeit des parallelen Algorithmus auf einem Prozessor ( $T_1 \ll T_S$ )
  - $T_p$ : Ausführungszeit des parallelen Algorithmus auf  $p$  Prozessoren
  - $S_R = T_1/T_p$  (**relativer Speedup**)
  - $\beta$ : sequentieller Anteil (nicht parallelisierbarer Anteil) des sequentiellen Verfahrens.
- $T_p = \beta T_1 + (1 - \beta) T_1 / p$ 
  - Berechnungsdauer für sequentiellen Anteil:  $\beta T_1$
  - Berechnungsdauer für parallelen Anteil:  $(1 - \beta) T_1 / p$
- **Gesetz von Amdahl:**  $S_R = \frac{p}{\beta p + (1 - \beta)} = \frac{1}{\beta + \frac{1 - \beta}{p}}$

## Maximaler relativer Speedup nach Amdahl

- $S_{\max} := S_R(p \rightarrow \infty) = 1 / \beta$ 
  - $\beta = 50\% \rightarrow S_{\max} = 2$
  - $\beta = 10\% \rightarrow S_{\max} = 10$
  - $\beta = 5\% \rightarrow S_{\max} = 20$
  - $\beta = 1\% \rightarrow S_{\max} = 100$
- **Mögliche Schlussfolgerung:** „Da selbst triviale parallele Programme immer einen sequentiellen Anteil aufweisen, lohnt sich Parallelisierung nicht wirklich.“
- **Einwand:** Das Gesetz von Amdahl betrachtet nicht skalierbare parallele Systeme; hier sind im  $p$  und  $\beta$  nicht mehr unabhängig.

## Gesetz von Gustafson-Barsis (1988)

- **Komplementärer Ansatz:**
  - $\alpha$ : sequentieller Anteil des parallelen Verfahrens.
  - $T_p := 1$  (konstant)
    - „Stehen mehr Prozessoren zur Verfügung können größere Probleme berechnet werden“.
    - $\leftrightarrow$  Amdahl: „Die Laufzeit eines Programms wird durch Parallelisierung bei konstanter Problemgröße verringert“.
- Für  $T_1$  ergibt sich:  $T_1 = \alpha + (1 - \alpha) p$
- **Gesetz von Gustafson-Barsis:**  $S_R = \alpha + (1 - \alpha) p$