

---

# Rechnerstrukturen

Vorlesung im Sommersemester 2007

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik



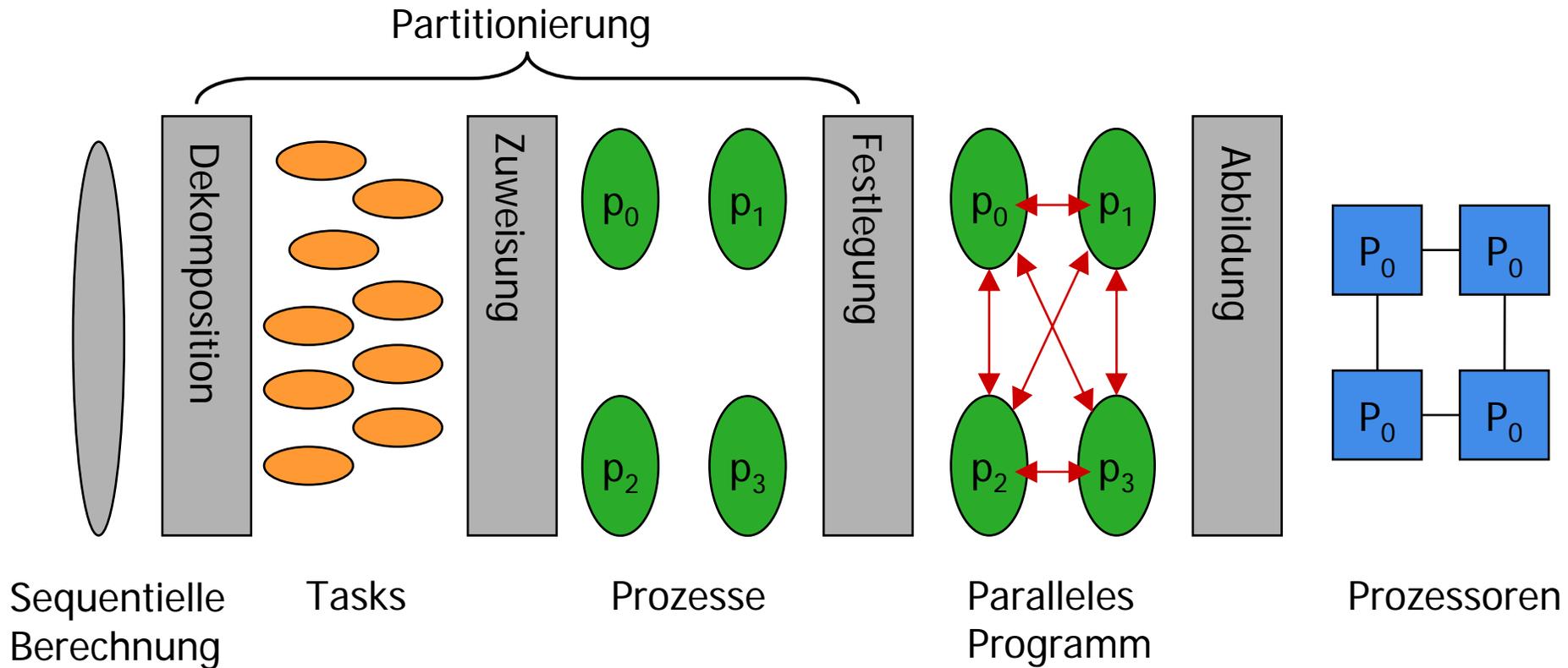
- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess/Thread-Ebene**

## 3.2: Allgemeine Grundlagen



- **Parallelisierungsprozess**

- Schritte bei der Parallelisierung und die Beziehung zwischen Tasks, Prozessen und Prozessoren



- **Parallelisierungsprozess**

- Festlegung

- Architektur und Programmiermodell sowie die Programmiersprache spielen eine Rolle

- Um die zugewiesenen Tasks ausführen zu können, benötigen die Prozesse Mechanismen

- » für den Zugriff auf die Daten,
- » für die Kommunikation (Austausch von Daten)
- » für die Synchronisation untereinander

- Fragen

- » Organisation der Datenstrukturen
- » Ablauf der Tasks
- » Explizite oder implizite Kommunikation

- Ziel:

- Reduzierung des Kommunikations- und Synchronisationsaufwandes (aus der Sicht des Prozessors)
- Erhalten der Lokalität der Datenzugriffe, soweit möglich
- Reduzierung des Parallelisierungsaufwandes
- Rechnerarchitekt: bereitstellen effizienter Mechanismen, die die Festlegung vereinfachen

- Parallelisierungsprozess

- Festlegung

- Programmiermodelle:

- Shared-Memory-Programmiermodell
- Nachrichten-orientiertes Programmiermodell
- Datenparalleles Programmiermodell

- **Parallelisierungsprozess**

- Festlegung

- Shared-Memory Programmiermodell

- Primitive:

Name	Syntax	Funktion
<b>CREATE</b>	<b>CREATE(p,proc,args)</b>	Generiere Prozess, der die Ausführung bei der Prozedur <b>proc</b> mit den Argumenten <b>args</b> startet
<b>G_MALLOC</b>	<b>G_MALLOC(size)</b>	Allokation eines gemeinsamen Datenbereichs der Größe <b>size</b> Bytes
<b>LOCK</b>	<b>LOCK(name)</b>	Fordere wechselseitigen exklusiven Zugriff an
<b>UNLOCK</b>	<b>UNLOCK(name)</b>	Freigeben des Locks

- **Parallelisierungsprozess**

- Festlegung

- Shared-Memory Programmiermodell

- Primitive:

Name	Syntax	Funktion
<b>BARRIER</b>	<b>BARRIER(name, number)</b>	Globale Synchronisation für <b>number</b> Prozesse
<b>WAIT_FOR_END</b>	<b>WAIT_FOR_END(number)</b>	Warten, bis <b>number</b> Prozesse terminieren
wait for flag	<b>while (!flag); or WAIT(flag)</b>	Warte auf gesetztes flag; entweder wiederholte Abfrage (spin) oder blockiere;
set flag	<b>flag=1; or SIGNAL(flag)</b>	Setze flag; weckt Prozess auf, der <b>flag</b> wiederholt abfragt

- **Parallelisierungsprozess**

- Festlegung

- Message Passing

- Primitive:

Name	Syntax	Funktion
<b>CREATE</b>	<b>CREATE</b> ( <i>procedure</i> )	Erzeuge Prozess, der bei <b>procedure</b> startet
<b>SEND</b>	<b>SEND</b> ( <i>src_addr</i> , <i>size</i> , <i>dest</i> , <i>tag</i> )	Sende <b>size</b> Bytes von Adresse <b>src_addr</b> an <b>dest</b> Prozess mit <b>tag</b> Identifier
<b>RECEIVE</b>	<b>RECEIVE</b> ( <i>buffer_addr</i> , <i>size</i> , <i>src</i> , <i>tag</i> )	Empfange eine Nachricht mit der Kennung <b>tag</b> vom <b>src</b> -Prozess und lege <b>size</b> Bytes in Puffer bei <b>buffer_addr</b> ab
<b>BARRIER</b>	<b>BARRIER</b> ( <i>name</i> , <i>number</i> )	Globale Synchronisation von <b>number</b> Prozessen

- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess/Thread-Ebene**

## 3.3: Quantitative Maßzahlen



- **Ausführungszeit**

- Die (Gesamt-)Ausführungszeit  $T$  (Execution Time) eines parallelen Programms ist die Zeit zwischen dem Starten der Programmausführung auf einem der Prozessoren bis zu dem Zeitpunkt, an dem der letzte Prozessor die Arbeit an dem Programm beendet hat.
- Zu beachten:
  - Prozessorzustand:
    - Während der Programmausführung sind alle Prozessoren in einem der drei Zustände:
      - » rechnend
      - » kommunizierend
      - » untätig



- **Ausführungszeit**

- Die Ausführungszeit eines parallelen Programms auf einem dediziert zugeordneten Parallelrechner setzt sich zusammen aus:

- **Berechnungszeit  $T_{\text{comp}}$  (Computation Time)**

- Die Zeit, die für die Rechenoperationen verwendet wird

- **Kommunikationszeit  $T_{\text{comm}}$  (Communication Time)**

- Die Zeit, die für Sende- und Empfangsoperationen verwendet wird

- **Untätigkeitszeit  $T_{\text{idle}}$  (Idle Time)**

- Die Zeit, die mit Warten (auf zu empfangene Nachrichten oder und zu versendende) verbraucht wird

- **Es gilt:**

- $T = T_{\text{comp}} + T_{\text{comm}} + T_{\text{idle}}$

- **Ausführungszeit**

- Übertragungszeit einer Nachricht  $T_{msg}$

- die Zeit, die für das Verschicken einer Nachricht von einer bestimmten Länge zwischen zwei Prozessoren benötigt wird

- Die Übertragungszeit setzt sich zusammen aus:

- der Startzeit  $t_s$  (Message Startup Time):

- » Die Zeit, die benötigt wird, um die Kommunikation zu initiieren

- Transferzeit  $t_w$  pro übertragenem Datenwort:

- » hängt von der physikalischen Bandbreite des Kommunikationsmediums ab.

- Voraussetzung:

- » Verbindungsnetz ist konfliktfrei, da sonst die Übertragungszeit nicht fest berechnet werden kann

- Ausführungszeit

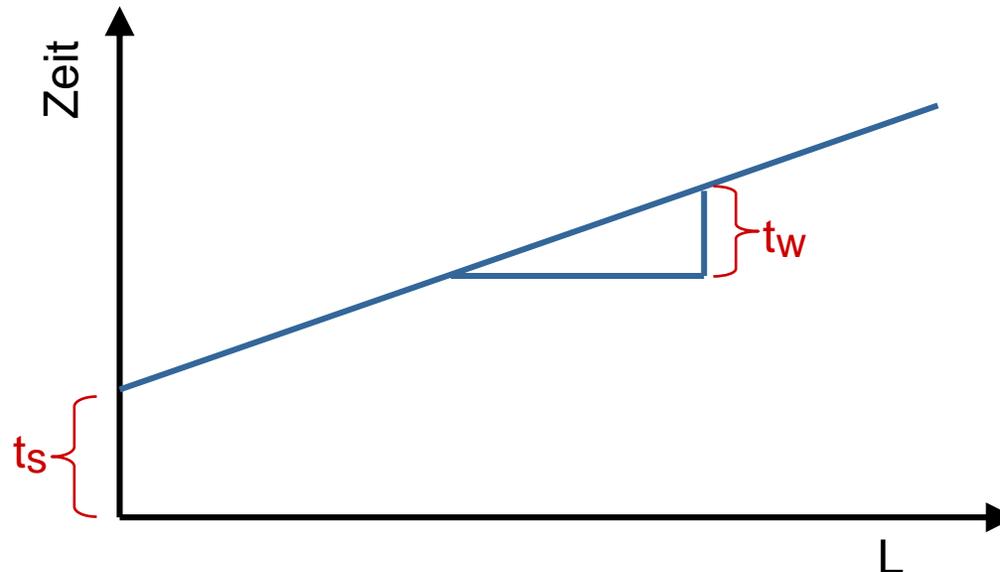
- Übertragungszeit einer Nachricht  $T_{\text{msg}}$

- Formel:

$$- T_{\text{msg}} = t_s + t_w * L,$$

mit L: Anzahl der Datenwörter

- Graphische Darstellung:



- **Parallelitätsprofil**

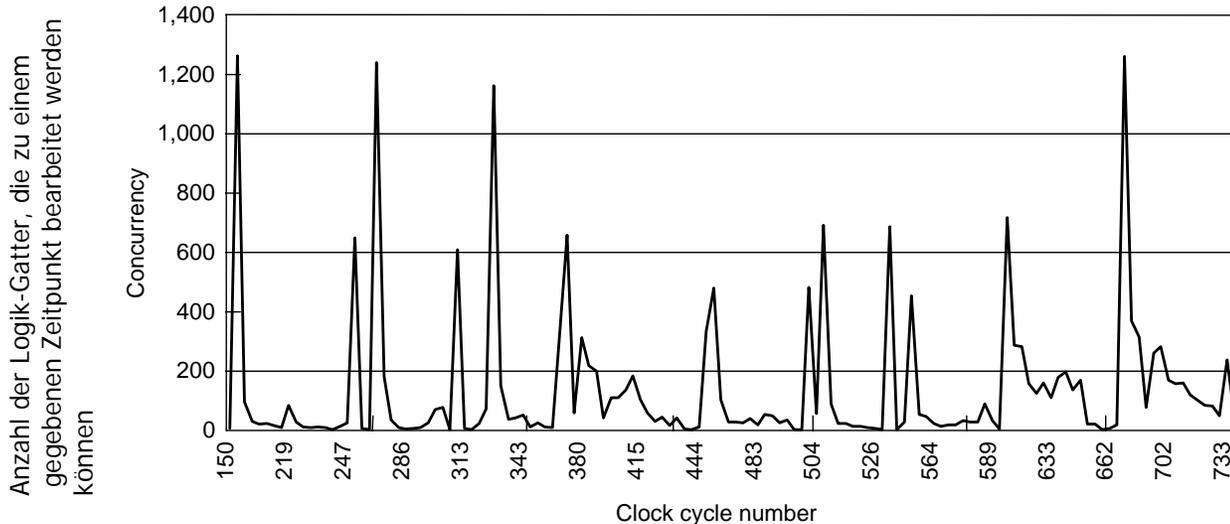
- misst die entstehende Parallelität in einem parallelen Programm bzw. bei der Ausführung auf einem Parallelrechner.
- Gibt eine Vorstellung von der inhärenten Parallelität eines Algorithmus/Programms und deren Nutzung auf einem realen oder ideellen Parallelrechner
- **Grafische Darstellung:**
  - Auf der x-Achse wird die Zeit und auf der y-Achse die Anzahl paralleler Aktivitäten angetragen.
  - Perioden von Berechnungs- Kommunikations- und Untätigkeitszeiten sind erkennbar.



- **Parallelitätsprofil**

- Zeigt an, wie viele Tasks einer Anwendung zu einem Zeitpunkt parallel ausgeführt werden können
- **Parallelitätsgrad  $PG(t)$ :**
  - Anzahl der Tasks, die zu einem Zeitpunkt parallel bearbeitet werden können

## Beispiel: Parallele ereignisgesteuerte Simulation der Logik-Synthese



Quelle: D. Culler: Parallel Computer Architecture.  
Morgan Kaufmann Publishers, 1999, p.87

## • Parallelitätsprofil

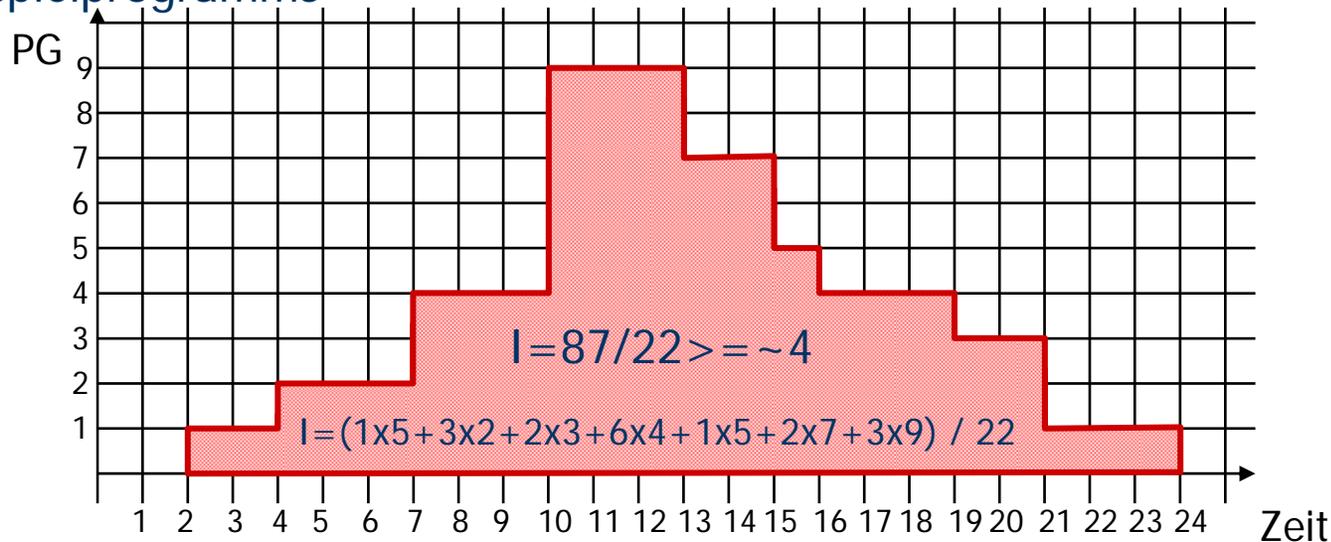
- Parallelindex I (Mittlerer Grad des Parallelismus):

$$I = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} PG(t) dt$$

$$I = \left( \sum_{i=1}^m i * t_i \right) / \left( \sum_{i=1}^m t_i \right)$$

Parallelitätsprofil eines  
Beispielprogramms

PG Bereich      Ausführungszeit



- **Quantitative Maßzahlen**

- Leistungsangaben zu Multiprozessorsystemen werden mit Leistungsangaben zu Einprozessorsystemen in Beziehung gesetzt

- **Notwendig:**

- Programm das auf beiden zu vergleichenden Systemen ablaufen kann

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Definitionen:

- $P(1)$ : Anzahl der auszuführenden (Einheits-) Operationen (Tasks) des Programms auf einem Einprozessorsystem.
- $P(n)$ : Anzahl der auszuführenden (Einheits-) Operationen (Tasks) des Programms auf einem Multiprozessorsystem mit  $n$  Prozessoren.
- $T(1)$ : Ausführungszeit auf einem Einprozessorsystem in Schritten (oder Takten).
- $T(n)$ : Ausführungszeit auf einem Multiprozessorsystem mit  $n$  Prozessoren in Schritten (oder Takten).

- Vereinfachende Voraussetzungen:

- $T(1) = P(1)$ ,
  - da in einem Einprozessorsystem (Annahme: einfacher Prozessor) jede (Einheits-) Operation in genau einem Schritt ausgeführt werden kann.
- $T(n) \leq P(n)$ ,
  - da in einem Multiprozessorsystem mit  $n$  Prozessoren ( $n \geq 2$ ) in einem Schritt mehr als eine (Einheits-)Operation ausgeführt werden kann.

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Beschleunigung  $S(n)$  (Speedup):

$$S(n) = \frac{T(1)}{T(n)}$$

- Gibt die Verbesserung in der Verarbeitungsgeschwindigkeit an
- Wert bezieht sich auf das jeweils bearbeitete Programm oder kann als Mittelwert eine Menge von Programmen angesehen werden
- Üblicherweise gilt:  $1 \leq S(n) \leq n$

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Effizienz  $E(n)$

$$E(n) = \frac{S(n)}{n}$$

- Gibt die relative Verbesserung in der Verarbeitungsgeschwindigkeit an
- Leistungssteigerung wird mit der Anzahl der Prozessoren  $n$  normiert
- Üblicherweise gilt:  $\frac{1}{n} \leq E(n) \leq 1$

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen
  - Beschleunigung (Speed-Up), Effizienz:
    - Algorithmenunabhängige Definition
      - Man setzt den besten bekannten sequentiellen Algorithmus für das Einprozessorsystem in Beziehung zum vergleichbaren parallelen Algorithmus für das Multiprozessorsystem.
    - Absolute Beschleunigung
    - Absolute Effizienz

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen
  - Beschleunigung (Speed-Up), Effizienz:
    - Algorithmenabhängige Definition
      - Man benutzt den parallelen Algorithmus so, als sei er sequentiell, und misst dessen Laufzeit auf einem Einprozessorsystem.
      - Der für die Parallelisierung erforderliche Zusatzaufwand an Kommunikation und Synchronisation kommt „ungerechterweise“ auch für den sequentiellen Algorithmus zum Tragen.
      - ➔ Absolute Beschleunigung
      - ➔ Absolute Effizienz

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen
  - Mehraufwand  $R(n)$  für die Parallelisierung:

$$R(n) = \frac{P(n)}{P(1)}$$

- Beschreibt den bei einem Multiprozessorsystem erforderlichen Mehraufwand für die Organisation, Synchronisation und Kommunikation der Prozessoren
- Es gilt:  $1 \leq R(n)$ 
  - Anzahl der auszuführenden Operationen eines parallelen Programms größer ist als diejenige des vergleichbaren sequentiellen Programms

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Auslastung  $U(n)$ :

$$U(n) = \frac{I(n)}{n} = R(n) \cdot E(n) = \frac{P(n)}{n \cdot T(n)}$$

- Entspricht dem normierten Parallelindex
- Gibt an, wie viele Operationen (Tasks) jeder Prozessor im Durchschnitt pro Zeiteinheit ausgeführt hat

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Folgerungen

- Alle definierten Ausdrücke haben für  $n = 1$  den Wert 1.
- Der Parallelindex gibt eine obere Schranke für die Leistungssteigerung:

$$1 \leq S(n) \leq I(n) \leq n$$

- Die Auslastung ist eine obere Schranke für die Effizienz:

$$\frac{1}{n} \leq E(n) \leq U(n) \leq 1$$

## • Parallelitätsprofil

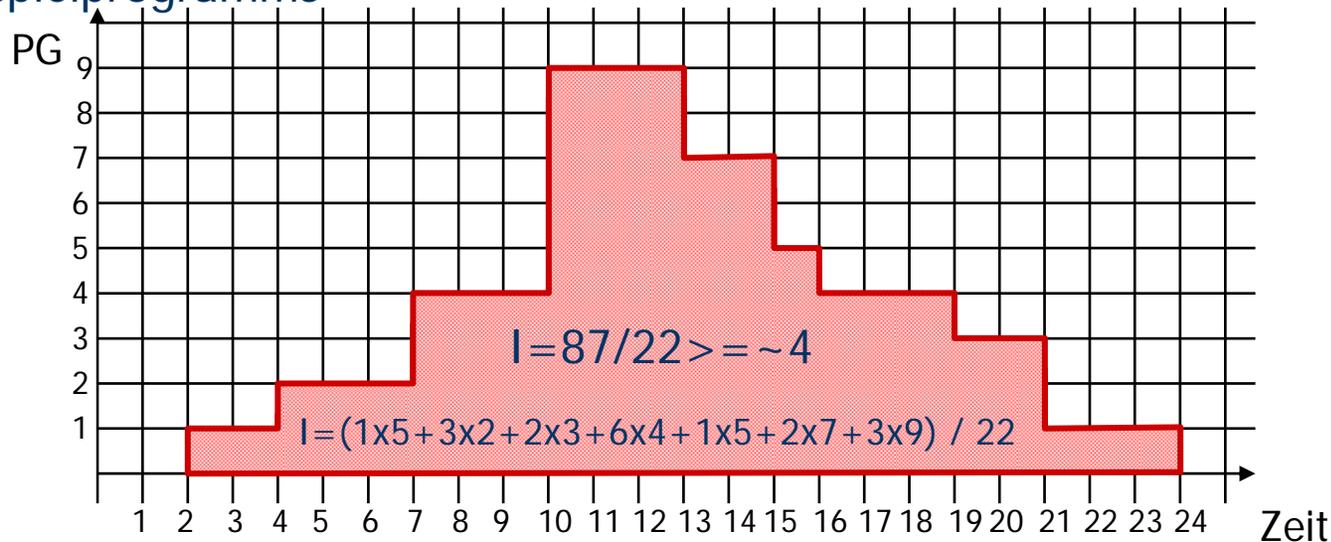
- Parallelindex I (Mittlerer Grad des Parallelismus):

$$I = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} PG(t) dt$$

$$I = \left( \sum_{i=1}^m i * t_i \right) / \left( \sum_{i=1}^m t_i \right)$$

Parallelitätsprofil eines  
Beispielprogramms

PG Bereich      Ausführungszeit



- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Parallelindex  $I(n)$ :

- Unter den vereinfachenden Voraussetzungen (siehe Folie 12-19) gilt:

$$I(n) = \frac{P(n)}{T(n)}$$

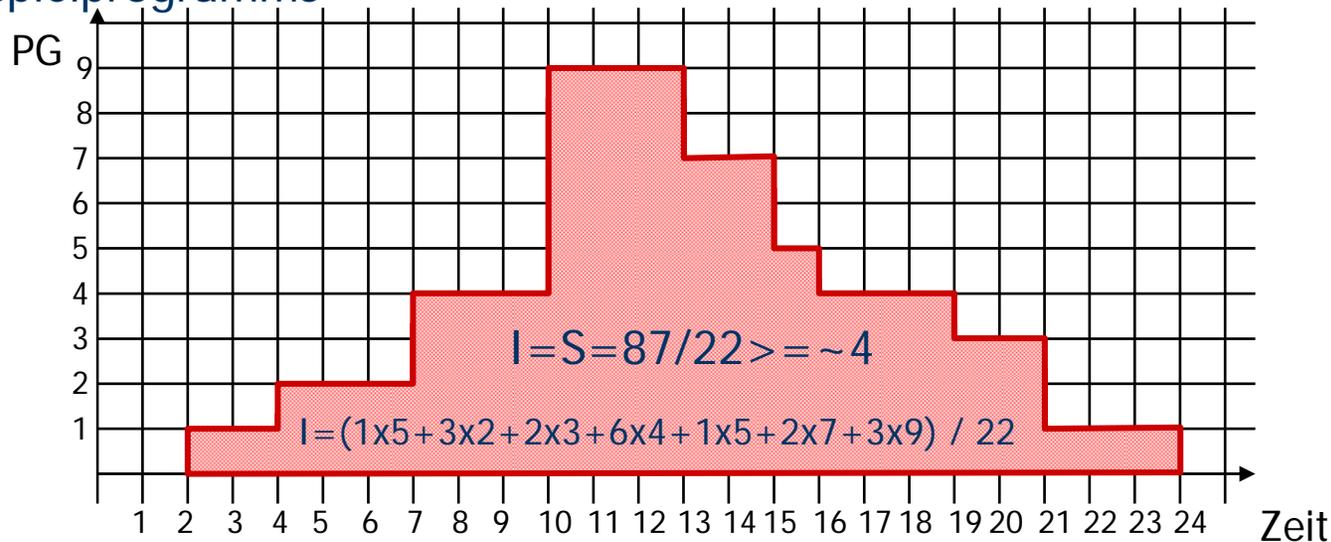
- und damit:  $I(n) = S(n)$

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Beziehung Parallelindex und Speedup

- Unter den vereinfachenden Voraussetzungen (Folie 12-19)

Parallelitätsprofil eines  
Beispielprogramms



- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen

- Zahlenbeispiel:

- Ein Einprozessorsystem benötige für die Ausführung von 1000 Operationen 1000 Schritte.
- Ein Multiprozessorsystem mit 4 Prozessoren benötige dafür 1200 Operationen, die aber in 400 Schritten ausgeführt werden können.

- Damit gilt:

$$P(1) = T(1) = 1000, P(4) = 1200, T(4) = 400$$

- Daraus ergibt sich:

$$S(4) = 2,5 \text{ und } E(4) = 0,625$$

- Die Leistungssteigerung verteilt sich als im Mittel zu 62,5% auf alle Prozessoren

- Vergleich von Multiprozessorsystemen zu Einprozessorsystemen
  - Zahlenbeispiel:
    - Parallelindex und Auslastung:  
 $I(4) = 3$  und  $U(4) = 0,75$ 
      - Es sind im Mittel drei Prozessoren gleichzeitig tätig, d.h., jeder Prozessor ist nur zu 75% der Zeit aktiv.
    - Mehraufwand:  
 $R(4) = 1,2$ 
      - Bei Ausführung auf dem Multiprozessorsystem sind 20% mehr Operationen als bei Ausführung auf einem Einprozessorsystem notwendig.

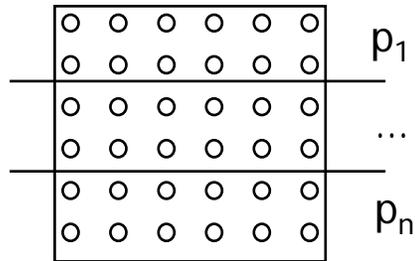
- **Skalierbarkeit eines Parallelrechners**

- das Hinzufügen von weiteren Verarbeitungselementen führt zu einer kürzeren Gesamtausführungszeit, ohne dass das Programm geändert werden muss.
- Insbesondere meint man damit eine lineare Steigerung der Beschleunigung mit einer Effizienz nahe bei Eins.
- Wichtig für die Skalierbarkeit ist eine angemessene Problemgröße.
- Bei fester Problemgröße und steigender Prozessorzahl wird ab einer bestimmten Prozessorzahl eine Sättigung eintreten. Die Skalierbarkeit ist in jedem Fall beschränkt.
- Skaliert man mit der Anzahl der Prozessoren auch die Problemgröße (scaled problem analysis), so tritt dieser Effekt bei gut skalierenden Hardware- oder Software-Systemen nicht auf.

- Gesetz von Amdahl

- Beispiel:

- Gegeben: ein zweidimensionales  $k \times k$ -Gitter
  - 1. Berechnungsphase: Ausführung einer Operation auf allen Gitterpunkten
    - » Annahme: keine Abhängigkeiten zwischen den Gitterpunkten
    - » Parallele Berechnung auf  $n$  Prozessoren



- 2. Berechnungsphase: Berechnung der Summe der  $k^2$  berechneten Werte der Gittersumme
  - » Jeder Prozessor addiert seine  $k^2/n$  berechneten Werte zur globalen Summe

- Gesetz von Amdahl

- Beispiel:

- Problem:

- Akkumulation der globalen Summe muss serialisiert werden!
- 2. Phase benötigt  $k^2$  Zeiteinheiten unabhängig von  $n$
- Ausführungszeit des parallelen Programms:  $k^2/n + k^2$
- Ausführungszeit des sequentiellen Programms:  $2k^2$
- Möglicher Speedup  $S$ :

$$\frac{2k^2}{\frac{k^2}{n} + k^2} = \frac{2n}{n+1}$$

- Selbst bei einer hohen Anzahl Prozessoren nicht mehr als 2!



- Gesetz von Amdahl
  - Gesamtausführungszeit  $T(n)$

$$T(n) = T(1) \cdot \frac{1-a}{n} + T(1) \cdot a$$

Ausführungszeit  
des sequentiell  
ausführbaren  
Programmteils  $a$

Ausführungszeit  
des parallel  
ausführbaren  
Programmteils  $1-a$

$a$ : Anteil des Programmteils,  
der nur sequentiell  
ausgeführt werden kann

- Beschleunigung

$$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1) \cdot \frac{1-a}{n} + T(1) \cdot a} = \frac{1}{\frac{1-a}{n} + a}$$

- Für  $n \rightarrow \infty$  :  $S(n) = \frac{1}{a}$

- Gesetz von Amdahl

- Beispiel:

- Erhöhung der Parallelität

- Aufteilung der 2. Berechnungsphase in zwei weiteren Teilphasen:

- » 1. Teilphase: Jeder Prozessor berechnet die Summe seiner berechneten Werte

- » Kann vollständig parallel abgearbeitet werden

- » 2. Teilphase: Akkumulation der Teilsummen

- » Weiterhin seriell!

- Ausführungszeit  $T(n) = k^2/n + k^2/n + n$

- Beschleunigung  $S(n) = n * 2k^2 / (2k^2 + n^2)$

- Wenn  $n$  groß genug, dann nahezu linear!

- Gesetz von Amdahl

- Diskussion

- Amdahls Gesetz zufolge kann eine kleine Anzahl von sequentiellen Operationen die mit einem Parallelrechner erreichbare Beschleunigung signifikant begrenzen.
- Beispiel:  $a = 1/10$  des parallelen Programms kann nur sequenziell ausgeführt werden, → das gesamte Programm kann maximal zehnmal schneller als ein vergleichbares, rein sequenzielles Programm sein.
- Jedoch: viele parallele Programme haben einen sehr geringen sequenziellen Anteil ( $a \ll 1$ )

- Synergetischer Effekt und superlinearer Speedup
  - Theorie : einen „superlinearen Speedup“ kann es nicht geben:
    - Jeder parallele Algorithmus lässt sich auf einem Einprozessorsystem simulieren, indem in einer Schleife jeweils der nächste Schritt jedes Prozessors der parallelen Maschine emuliert wird.

- Synergetischer Effekt und superlinearer Speedup
  - Ein „superlinearer Speed-up“ kann real beobachtet werden bei
    - parallelem Backtracking (*depth-first search*)
    - Beim Programmlauf auf einem Rechner passen die Daten nicht in den Hauptspeicher des Rechners (häufiger Seitenwechsel), aber: bei Verteilung auf die Knoten des Multiprozessors können die parallelen Programme vollständig in den Cache- und Hauptspeichern der einzelnen Knoten ablaufen.

- Weitere grundsätzliche Probleme bei Multiprozessoren
  - Verwaltungsaufwand (Overhead)
    - Steigt mit der Zahl der zu verwaltenden Prozessoren
  - Möglichkeit von Systemverklemmungen (*deadlocks*)
  - Möglichkeit von Sättigungerscheinungen
    - können durch Systemengpässe (*bottlenecks*) verursacht werden.

---

# Rechnerstrukturen

Vorlesung im Sommersemester 2006

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik



- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess/Thread-Ebene**

## 3.4: Verbindungsstrukturen



- **Verbindungsnetze in Multiprozessoren**
  - Ermöglichen die Kommunikation und Kooperation zwischen den Verarbeitungselementen (Knoten)
    - Zuverlässiger Austausch von Informationen
  - Einsatz eines Verbindungsnetzwerks
    - Multiprozessor mit verteiltem Speicher (nachrichtenorientierter Multiprozessor)
      - Verbinden physikalisch jeden Knoten für das Versenden von Nachrichten
      - Direkte Send/Receive-Kommunikation zwischen den Knoten
    - Multiprozessor mit gemeinsamem Speicher
      - Ermöglicht den Zugriff aller Knoten auf den gemeinsamen Speicher
      - Kommunikation durch Lesen und Schreiben auf gemeinsame Daten

- Charakterisierung von Verbindungsnetzwerken

- Latenz

- Übertragungszeit einer Nachricht  $T_{msg}$ 
  - die Zeit, die für das Verschicken einer Nachricht von einer bestimmten Länge zwischen zwei Prozessoren benötigt wird
- Die Übertragungszeit setzt sich zusammen aus:
  - der Startzeit  $t_s$  (Message Startup Time):
    - » Die Zeit, die benötigt wird, um die Kommunikation zu initiieren
  - Transferzeit  $t_w$  pro übertragenem Datenwort:
    - » hängt von der physikalischen Bandbreite des Kommunikationsmediums ab.
  - Voraussetzung:
    - » Verbindungsnetz ist konfliktfrei, da sonst die Übertragungszeit nicht fest berechnet werden kann

- Charakterisierung von Verbindungsnetzwerken

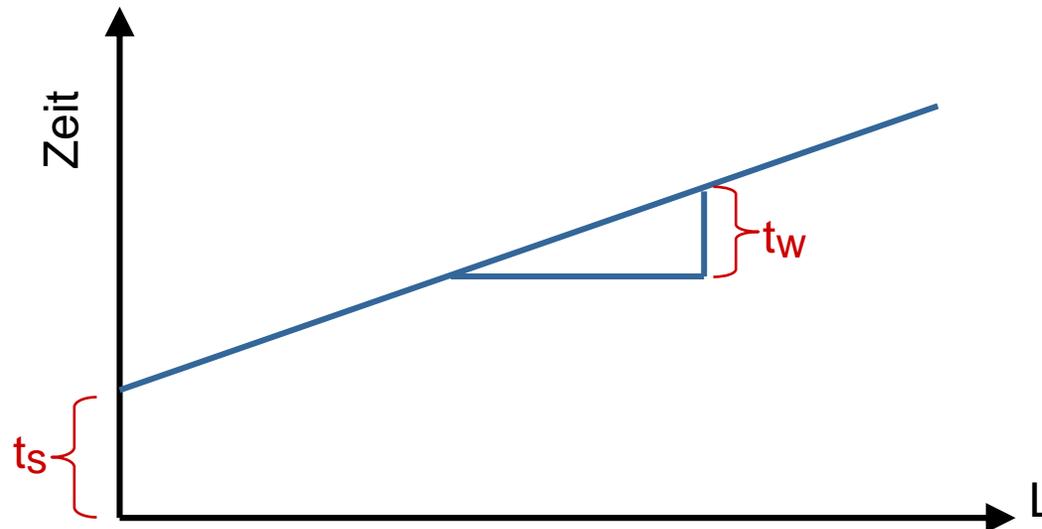
- Latenz

- Formel:

$$- T_{\text{msg}} = t_s + t_w * L,$$

mit L: Anzahl der Datenwörter

- Graphische Darstellung:



- Charakterisierung von Verbindungsnetzwerken
  - Latenz (Übertragungszeit einer Nachricht, latency)
    - Software-Overhead
      - in Verbindung mit dem Senden und Empfangen von Nachrichten

- Charakterisierung von Verbindungsnetzwerken
  - Beispiel für Software-Overhead

Erzeugerprozess:

```
send(proci, processi, @sbuffer, num_bytes)
```

**Sender**

Systemaufruf  
Prüfe Schutzbed.  
DMA Init.  
DMA nach NI

Verbraucherprozess:

```
receive(@rbuffer, max_bytes)
```

**Empfänger**

DMA vom Netzwerk in den Puffer  
BS Interrupt und Dekodierung der Nachr.  
BS kopiert Systempuffer in Userpuffer  
Reschedule Benutzerprozess  
Lesen der Nachricht

Zeit

NI: Netzwerkschnittstelle  
DMA: Direktspeicherzugriff



- Charakterisierung von Verbindungsnetzwerken
  - Latenz (Übertragungszeit einer Nachricht, latency)
    - Kanalverzögerung (channel delay)
      - Dauer Belegung eines Kommunikationskanals durch eine Nachricht
      - Kanal:
        - » Physikalische Verbindung zwischen Schalterelementen oder Knoten mit einem Puffer zum Halten der Daten während ihrer Übertragung
      - Verbindung (link)
        - » Menge von Leitungen

- Charakterisierung von Verbindungsnetzwerken
  - Latenz (Übertragungszeit einer Nachricht, latency)
    - Schaltverzögerung, Routing-Verzögerung (switching delay, routing delay)
      - Zeit, einen Weg zwischen zwei Knoten aufzubauen
      - Pfadberechnung oder Wegefindung (Routing)
        - » die Art, wie der Weg einer Nachricht vom Sender zum Zielknoten berechnet wird
        - » Zu einer Verbindungsstruktur kann es mehrere Wegefindungsalgorithmen geben
        - » einfache Implementierung in Verbindungselementen mit Hilfe eines schnellen Hardware-Algorithmus

- Charakterisierung von Verbindungsnetzwerken
  - Latenz (Übertragungszeit einer Nachricht, latency)
    - Blockierungszeit (contention time)
      - Wird verursacht, wenn zu einem Zeitpunkt mehr als eine Nachricht auf eine Netzwerkressource zugreifen
    - Blockierung (contention)
      - Ein Verbindungsnetzwerk heisst blockierungsfrei, falls jede gewünschte Verbindung zwischen Prozessoren oder zwischen Prozessoren und Speichern unabhängig von schon bestehenden Verbindungen hergestellt werden kann

- Charakterisierung von Verbindungsnetzwerken
  - Durchsatz oder Übertragungsbandbreite (bandwidth)
    - Maximale Übertragungsleistung des Verbindungsnetzwerkes oder einzelner Verbindungen, meist in Megabits pro Sekunde (MBit/s) oder Megabytes pro Sekunde (MB/s)
  - Bisektionsbandbreite (bisection bandwidth)
    - Maximale Anzahl von Megabytes pro Sekunde, die das Netzwerk über die Bisektionslinie, die das Netzwerk in zwei gleiche Hälften teilt, transportieren kann

- Charakterisierung von Verbindungsnetzwerken
  - Diameter oder Durchmesser  $r$  (*diameter*):
    - maximale Distanz für die Kommunikation zweier Prozessoren, also die Anzahl der Verbindungen, die durchlaufen werden müssen. Man spricht auch von der maximalen Pfadlänge zwischen zwei Knoten.
  - Verbindungsgrad eines Knotens  $P$  (node degree, connectivity)
    - ist definiert als die Anzahl der direkten Verbindungen, die von einem Knoten zu anderen Knoten bestehen.
  - Mittlere Distanz  $d_a$  (average distance) zwischen zwei Knoten
    - Anzahl der Links auf dem kürzesten Pfad zwischen zwei Knoten
    - $P/d_a$  ist die maximale Anzahl neuer Nachrichten, die von jedem Knoten in einem Zyklus in das Netzwerk eingebracht werden können



- **Charakterisierung von Verbindungsnetzwerken**
  - **Komplexität oder Kosten:**
    - Kosten für die Implementierung einer Hardware
    - Aufwand für das Verbindungsnetz gemessen in der Anzahl und der Art der Schaltelemente und Verbindungsleitungen.
  - **Erweiterbarkeit:**
    - Multiprozessoren können begrenzt, stufenlos oder nur durch Verdopplung der Anzahl der Prozessoren erweiterbar sein.
  - **Skalierbarkeit:**
    - Fähigkeit, die wesentlichen Eigenschaften des Verbindungsnetzes auch bei beliebiger Erhöhung der Knotenzahl beizubehalten.

- Charakterisierung von Verbindungsnetzwerken

- Ausfalltoleranz oder Redundanz :

- Verbindungen zwischen Knoten sind selbst dann noch zu schalten, wenn einzelne Elemente des Netzes (Schaltelemente, Leitungen) ausfallen.
- Ein fehlertolerantes Netz muss also zwischen jedem Paar von Knoten mindestens einen zweiten, redundanten Weg bereitstellen.

Die Eigenschaft eines Systems, bei Ausfall einzelner Komponenten unter deren Umgehung funktionstüchtig zu bleiben, wenn auch mit verminderter Leistung, wird als **Graceful degradation** bezeichnet.

- **Charakterisierung von Verbindungsnetzwerken**
  - **Art der Adressierung:**
    - **zielbasiert (destination-based routing):**
      - Kopfteil eines Pakets (oder einer Nachricht) wird mit einer systemweit eindeutigen Empfängeradresse versehen, die bei der Wegefindung von jedem Zielknoten zur Auswahl eines Übertragungskanals genutzt wird
    - **quellenbasiert (source-based routing):**
      - Paket wird mit allen Informationen versehen, um über die Zwischenknoten zum Empfänger zu gelangen. Für jeden Zwischenknoten wird im voraus die Abzweigung bestimmt, die das Paket nehmen muss.

- Charakterisierung von Verbindungsnetzwerken

- Art des Datentransfers:

- Durchschalte- oder Leitungsvermittlung (circuit switching):

- Eigenschaft eines Netzes eine direkte Verbindung zwischen zwei oder mehreren Knoten eines Netzes zu schalten. Die physikalische Verbindung bleibt für die gesamte Dauer der Informationsübertragung bestehen.

- » Blockierungsfreie Kommunikation

- » Kurze Latenz

- » Gut geeignet für lange Nachrichten, da die Zeit zum Aufsetzen einer Nachricht im Verhältnis zur Übertragungszeit kurz ist

- » Übertragungszeit einer Nachricht der Länge  $L$  über eine Distanz  $d$  beträgt:  $L/b + d\delta$ , mit individueller Schaltverzögerung  $\delta$  und der Bandbreite  $b$  eines Kanals



- Charakterisierung von Verbindungsnetzwerken
  - Art des Datentransfers:
    - Paketvermittlung (packet switching):
      - Datenpakete fester Länge oder Nachrichten variabler Länge werden entsprechend einem Wegefindungsalgorithmus (routing) vom Absender zum Empfänger geschickt
      - Nachrichten mit Adresse und Daten werden durch das Netzwerk verschickt
        - » Adresse wird in jedem Knoten gelesen und die Nachricht wird zum nächsten Knoten weitergeleitet, bis die Nachricht das Ziel erreicht
        - » Günstig für kurze Nachrichten

- Charakterisierung von Verbindungsnetzwerken

- Art des Datentransfers:

- Paketvermittlung (packet switching):

- Übertragungsmodi: Store-and-forward-Modus

- » Jeder Knoten enthält einen Puffer zum Aufnehmen der vollständigen Nachricht
- » Nachricht wird von jedem Zwischenknoten in Empfang genommen, vollständig zwischengespeichert und dann weiter übertragen
- » Nachfolgende Pakete werden nacheinander verschickt
- » Gegenüber Circuit Switching: höhere Bandbreite, aber auch höhere Latenz
- » Übertragungszeit einer Nachricht der Länge  $L$  über eine Distanz  $d$  von einer Quelle zum Ziel beträgt  $d(L/b + \delta)$

- Charakterisierung von Verbindungsnetzwerken
  - Art des Datentransfers:
    - Paketvermittlung (packet switching):
      - Übertragungsmodi: Cut-through oder wormhole
      - Phit und Flusskontrolle
        - » Eine Nachricht selbst wird in eine Anzahl von Übertragungseinheiten (phits – physical transfer units – oder auch flits – flow control digits – genannt) zerlegt.
        - » Ein Phit ist dabei die Datenportion, die zu einem Zeitpunkt zwischen zwei Knoten übertragen werden kann.
        - » Bei der Nachrichtenübertragung zwischen nicht benachbarten Sender- und Empfängerknoten sind Puffer nötig.

- Charakterisierung von Verbindungsnetzwerken
  - Art des Datentransfers:
    - Paketvermittlung (packet switching):
      - Übertragungsmodi: Virtual-cut-through-Modus:
        - » Nachricht wird aufgeteilt in Zellen (Flits) fester Größe
        - » Der Kopfteil der Nachricht enthält die Empfängeradresse und bestimmt den einzuschlagenden Weg. Flits mit Daten folgen dem Kopf auf dem Pfad von der Quelle zum Ziel
        - » Bei Ankunft des Kopfs einer Nachricht wird dieser dekodiert . Nachfolgende Flits werden automatisch an den nächsten Knoten auf dem ausgewählten Pfad weitergeleitet , ein Flit pro Zeiteinheit gemäß einer Pipeline-Verarbeitung
        - » Kopf-Information wird festgehalten bis letztes Flit angekommen ist.
        - » ankommende Daten werden nur im Konfliktfall im Knoten vollständig zwischengespeichert.
        - » In jedem Knoten werden Puffer bereit gehalten, die auch ein maximal großes Nachrichtenpaket zwischenspeichern können

- Charakterisierung von Verbindungsnetzwerken
  - Art des Datentransfers:
    - Paketvermittlung (packet switching):
      - Übertragungsmodi: Wormhole-routing-Modus:
        - » solange keine Übertragungskanäle blockiert sind, mit den Virtual-cut-through-Modus identisch.
        - » Falls der Kopfteil der Nachricht auf einen Kanal trifft, der gerade belegt ist, wird er abgeblockt. Alle nachfolgenden Übertragungseinheiten der Nachricht verharren dann ebenfalls an ihrer augenblicklichen Position, bis die Blockierung aufgehoben ist. Durch das Verharren werden die Puffer nachfolgender Kanäle auch für weitere Nachrichten blockiert.

- Charakterisierung von Verbindungsnetzwerken
  - Art des Datentransfers:
    - Paketvermittlung (packet switching):
      - Übertragungsmodi: Buffered wormhole routing:
        - » Kompromisslösung zwischen Virtual-cut-through- und Wormhole-routing-Modus eingesetzt,
        - » begrenzter Puffer zur Aufnahme kleinerer Pakete vorhanden
        - » größere Pakete werden im Blockierungsfall – ähnlich dem Wormhole-routing-Modus – in den Puffern mehrerer Knoten zwischengespeichert.

- **Topologie**

- **Statische Verbindungsnetzwerke**

- Die Prozessoren sind mit jedem anderen Prozessor direkt verbunden
- Fest installierte Verbindungen zwischen Paaren von Netzknoten
  - Die Knoten sind mit Punkt-zu-Punkt-Verbindungen verbunden, die sich während der Programmausführung nicht ändern

- **Dynamische Verbindungsnetze**

- Die Knoten sind über Schaltelemente miteinander verbunden, die konfiguriert werden können, um den Kommunikationsanforderungen des ausführenden Programms zu genügen
  - Direkte fest installierte Verbindungen zwischen den Knoten existieren nicht