Karlsruher Institut für Technologie

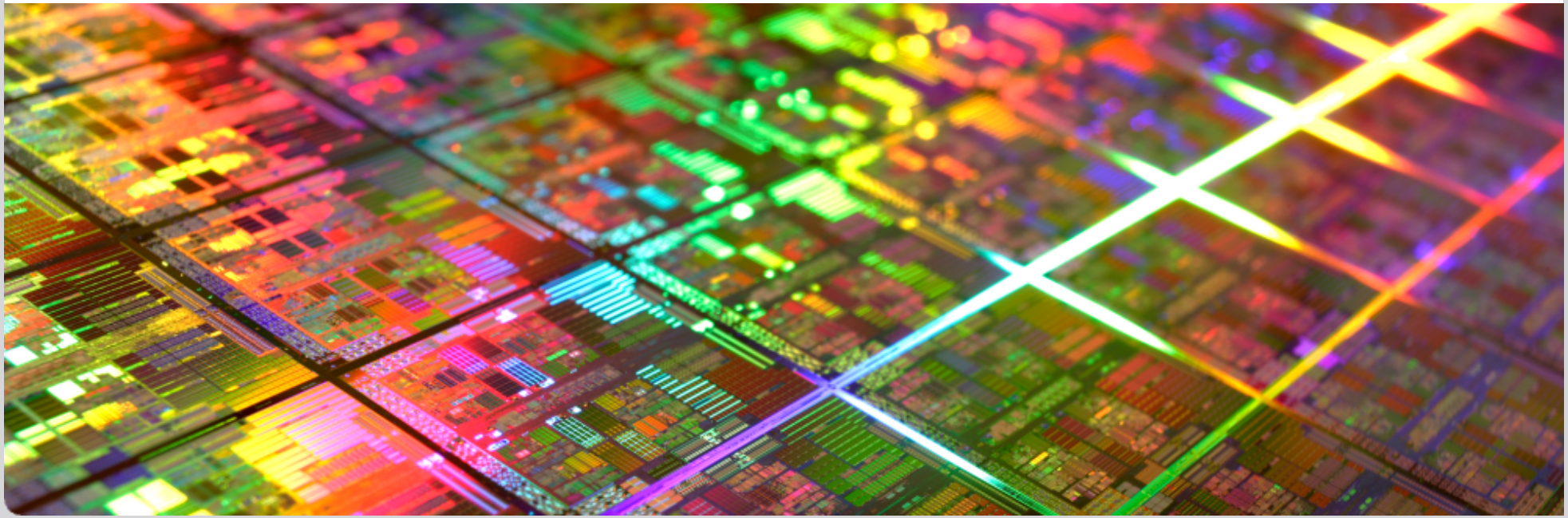
Zentralübung Rechnerstrukturen im SS 2014

Parallelismus und Parallele Programmierung

Oliver Mattes, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

24. Juni 2014



Überblick

Inhalt der Übung:

- Wiederholung: Tomasulo
- Parallelismus:
  - Parallelrechner und Quantitative Maßzahlen
  - Aufgabe 1
- Parallele Programmierung:
  - Parallelisierungsprozess
  - Programmiermodelle
  - Aufgabe 2
- Sonstiges:
  - Klausuraufgabe 2013/14-WS und 2013-SS
  - **Hinweise zur Klausur**

Tomasulo: Motivation

Karlsruher Institut für Technologie

Motivation

- Superskalarer Prozessor
  $\Rightarrow$ mehrere Funktionseinheiten (FU) in der Pipeline
- Mehrere Befehle in der Pipeline
  $\Rightarrow$ Abhängigkeiten (Hazards) zwischen den Befehlen

$\Rightarrow$ Dynamische Zuordnung und Ausführung

Ziel

- Optimale Auslastung der vorhandenen Funktionseinheiten
- Möglichst frühzeitiges Anstoßen der Befehle
- Überbrückung unterschliedlicher Bearbeitungszeiten (# Takte) der verschiedene Funktionseinheiten

Tomasulo: Komponenten und Tabellen

Registerstatustabelle/-datei – weitere Schreibweisen

Feld	R0	R1	R2	F0	F2	...
Value	42	-	23	47,11	0	...
Valid	1	0	1	1	1	
RS	0	1	0	0	0	

- Welches Register ist gültig, hat welchen Wert oder wird von welcher Einheit gerade berechnet?
- Verschiedene Arten und Anzahl an Registern
- Direkte Zahlenwerte oder Verweis auf Register
- Gibt bei Verwendung von Register-Renaming auch Auskunft über Abbildung von Architekturregistern auf physische Register
- Zeile RS mit Zahlen oder mit Namen der Einheiten
- **Nur eine Definition/Schreibweise gleichzeitig verwenden!**

Tomasulo: Komponenten und Tabellen

Registerstatustabelle/-datei – weitere Schreibweisen

Feld	R1	R2	R3	R4	R5	...
Value	(R1)	-	-	(R1)*(R2)	(R5)	...
Valid	1	0	0	1	1	
RS	0	1	Mul 1	0	0	

- Welches Register ist gültig, hat welchen Wert oder wird von welcher Einheit gerade berechnet?
- Verschiedene Arten und Anzahl an Registern
- Direkte Zahlenwerte oder Verweis auf Register
- Gibt bei Verwendung von Register-Renaming auch Auskunft über Abbildung von Architekturregistern auf physische Register
- Zeile RS mit Zahlen oder mit Namen der Einheiten
- **Nur eine Definition/Schreibweise gleichzeitig verwenden!**

Tomasulo: Komponenten und Tabellen

Register-Renaming – Beispiel

```
add   r4, r2, r1          add   r4, r2, r1
div   r2, r3, r4    ⇒     div   r2, r3, r4
store (r5), r2            store (r5), r2
add   r2, r1, r4          add   r6, r1, r4
```

- (Ausgabe-)Abhängigkeit (WAW) zwischen Befehl 2 und 4

Nach Registerumbenennung:

- keine Ausgabe- oder Gegen-Abhängigkeit mehr vorhanden

$\Rightarrow$ Befehle 2 und 4 können parallel ausgeführt werden

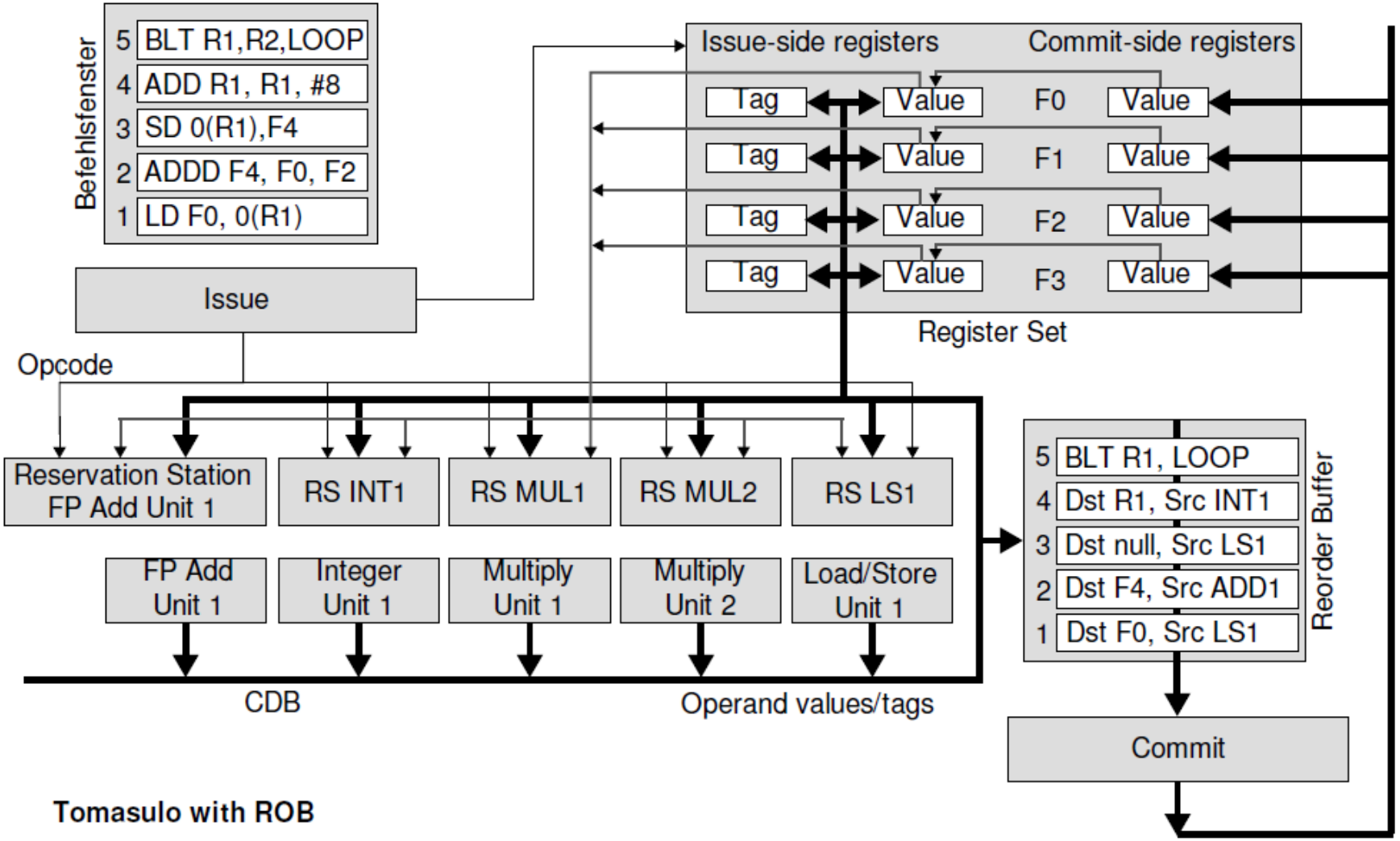
Tomasulo: Komponenten und Tabellen

Reservation Stations (Reservierungstabelle)

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2
L/S										
Add/Sub1 Add/Sub2										
Mul/Div1 Mul/Div2										

- Welche Befehle können mit ihren Operanden demnächst ausgeführt werden?
- Je Ausführungseinheit mit einem oder mehreren Einträgen
- pro Operand: Wert, Gültigkeit, Quelle (Herkunft)
- Dezentral an jeder Einheit oder Familie von Einheiten
- **Heute:** Tendenz zur zentralen Zuteilung
  Bsp.: Intel Westmere - 36 Entry Unified Scheduler

Tomasulo: Funktionseinheiten



Tomasulo with ROB

Tomasulo: Typische Aufgabenstellungen

Karlsruher Institut für Technologie

Aufgaben in Klausuraufgaben

- Bestimmung von Zuteilung (IS), Ausführung (EX) und Ergebnisbus (WB) je Befehl
- Verlauf der Pipeline (Wann welcher Befehl in welcher Stufe)
- Zustand der Reservierungstabelle und Registerdatei x Takte später angeben
- Füllstand der Reservierungstabelle
- ...

Tomasulo: Klausur 2013/14-WS

Karlsruher Institut für Technologie

Aufgabe 6b)

Untenstehend finden Sie den Zustand der Reservierungstabelle und der Registerdatei eines Superskalarprozessors nach Abarbeitung des ersten Taktes der in Listing 1 dargestellten Befehlsfolge. Geben Sie den Zustand der Reservierungstabelle, sowie der Registerdatei nach Ablauf von Takt 4, d.h. nach drei weiteren Takten, unter Berücksichtigung der in Listing 1 dargestellten Befehlsfolge wieder.

Pro Takt kann ein Befehl in die Reservierungstabelle eingetragen werden. Eine Subtraktion benötigt 2 Takte, eine Multiplikation 6 Takte und eine Division 9 Takte.

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b)

- Takt: 1

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	(R1)	(R2)	(R3)	-
Valid	1	1	1	0
RS	-	-	-	Add/Sub 1

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2
Add/Sub 1	0	0	sub	R4	(R1)	1	-	(R3)	1	-
Add/Sub 2	1									
Mul 1	1									
Div 1	1									

Remaining cycles in FU

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b)

- Takt: 2

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	(R1)	(R2)	-	-
Valid	1	1	0	0
RS	-	-	Div 1	Add/Sub 1

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2	
Add/Sub 1	0	1	sub	R4	(R1)	1	-	(R3)	1	-	1
Add/Sub 2	1										
Mul 1	1										
Div 1	0	0	div	R3	(R2)	1	-		0	A/S1	

Remaining cycles in FU

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b)

- Takt: 3

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	(R1)	-	-	-
Valid	1	0	0	0
RS	-	Add/Sub 2	Div 1	Add/Sub 1

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2	
Add/Sub 1	0	1	sub	R4	(R1)	1	-	(R3)	1	-	0
Add/Sub 2	0	0	sub	R2	(R1)	1	-	(R2)	1	-	
Mul 1	1										
Div 1	0	0	div	R3	(R2)	1	-		0	A/S1	

Remaining cycles in FU

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b)

- Takt: 4

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	-	-	-	(R1)-(R3)
Valid	0	0	0	1
RS	Mul 1	Add/Sub 2	Div 1	-

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2	Remaining cycles in FU
Add/Sub 1	1	0	sub	R4	(R1)	1	-	(R3)	1	-	
Add/Sub 2	0	1	sub	R2	(R1)	1	-	(R2)	1	-	1
Mul 1	0	0	mul	R1		0	A/S2		0	Div 1	
Div 1	0	1	div	R3	(R2)	1	-	(R1)-(R3)	1	-	8

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Tomasulo: Klausur 2013/14-WS

Achtung: Verschiedene Angaben in der Aufgabenstellung möglich!

- Wann stehen Ergebnisse dem nächsten Befehl zur Verfügung?
- Folgt dann noch die IS Stufe?

- Volles Bypassing: EX $\Rightarrow$ EX

$\Rightarrow$ IF von zweitem Befehl war schon und Ausführung startet direkt, sobald Ergebnis von erstem Befehl berechnet ist

- Das Ergebnis kann erst im Takt nach dem Ende der Berechnung auf den Ergebnisbus gelegt werden.
- Befehle, die von anderen abhängen, können mit ihrer Berechnung erst im Takt nachdem der letzte Operand auf den Ergebnisbus gelegt wurde beginnen.

$\Rightarrow$ EX $\Rightarrow$ WB/IS $\Rightarrow$ EX

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b) – Korrektur

- Takt: 4 – Wenn Berechnung erst 1 Takt nach letztem Operand

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	-	-	-	(R1)-(R3)
Valid	0	0	0	1
RS	Mul 1	Add/Sub 2	Div 1	-

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2
Add/Sub 1	1	0	sub	R4	(R1)	1	-	(R3)	1	-
Add/Sub 2	0	1	sub	R2	(R1)	1	-	(R2)	1	-
Mul 1	0	0	mul	R1		0	A/S2		0	Div 1
Div 1	0	0	div	R3	(R2)	1	-	(R1)-(R3)	1	-

1

Remaining cycles in FU

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Tomasulo: Klausur 2013/14-WS

Aufgabe 6b) – Korrektur

- Takt: 5 – Wenn Berechnung erst 1 Takt nach letztem Operand

Registerdatei (Register Status):

Feld	R1	R2	R3	R4
Value	-	-	-	(R1)-(R3)
Valid	0	0	0	1
RS	Mul 1	Add/Sub 2	Div 1	-

Programmcode:

Takt	Befehlsfolge
1	`sub R4, R1, R3`
2	`div R3, R2, R4`
3	`sub R2, R1, R2`
4	`mul R1, R2, R3`

Format: `Op Ziel, Q1, Q2`

Reservierungstabelle (Reservation Stations):

Unit	Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2	
Add/Sub 1	1	0	sub	R4	(R1)	1	-	(R3)	1	-	
Add/Sub 2	0	1	sub	R2	(R1)	1	-	(R2)	1	-	0
Mul 1	0	0	mul	R1		0	A/S2		0	Div 1	
Div 1	0	1	div	R3	(R2)	1	-	(R1)-(R3)	1	-	8

Remaining cycles in FU

Bearbeitungszeiten:

Befehl	sub	mul	div
Dauer (Takte)	2	6	9

Karlsruher Institut für Technologie

Fragen zu Tomasulo?

Parallelismus: Parallelrechner

Karlsruher Institut für Technologie

Klassifikation nach Flynn

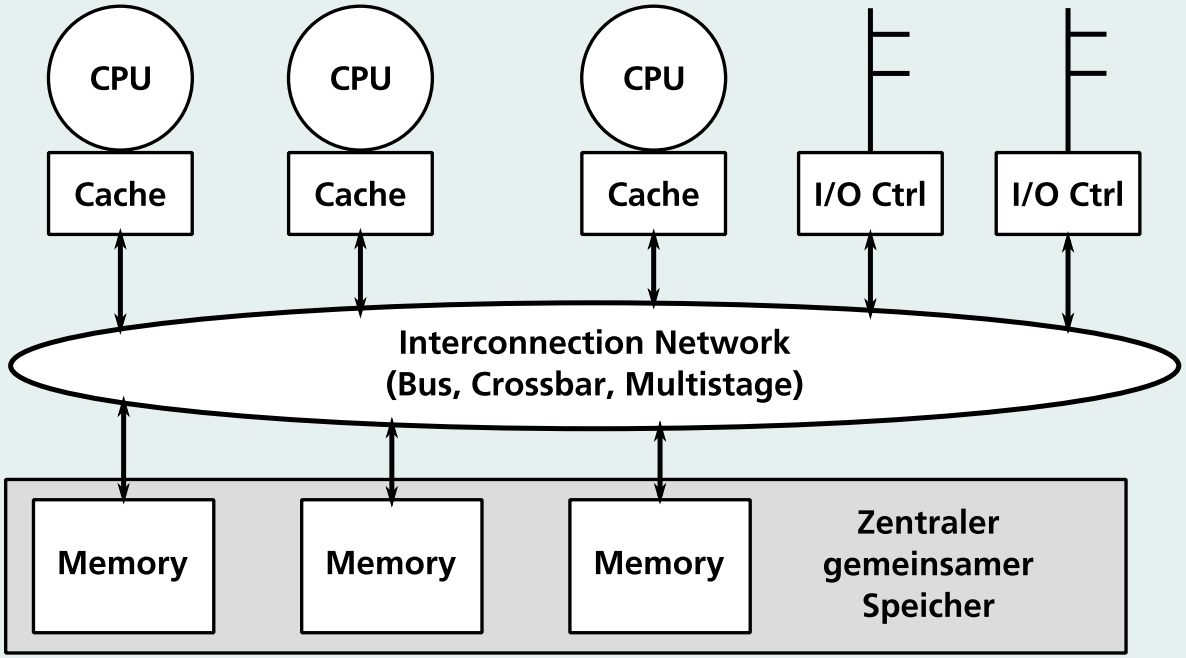
Vier Klassen von Rechnerarchitekturen:

- SISD Single Instruction, Single Data
  - Uniprozessor
- SIMD Single Instruction, Multiple Data
  - Vektorrechner, Feldrechner
- MISD Multiple Instructions, Single Data
  - ?
- MIMD **Multiple Instructions, Multiple Data**
  - Multiprozessor

Parallelismus: Parallele Architekturen

Multiprozessoren mit gemeinsamem Speicher

- **Globaler Speicher $\Leftrightarrow$ Gemeinsamer Adressraum**
- Beispiel: Symmetrischer Multiprozessor (SMP)

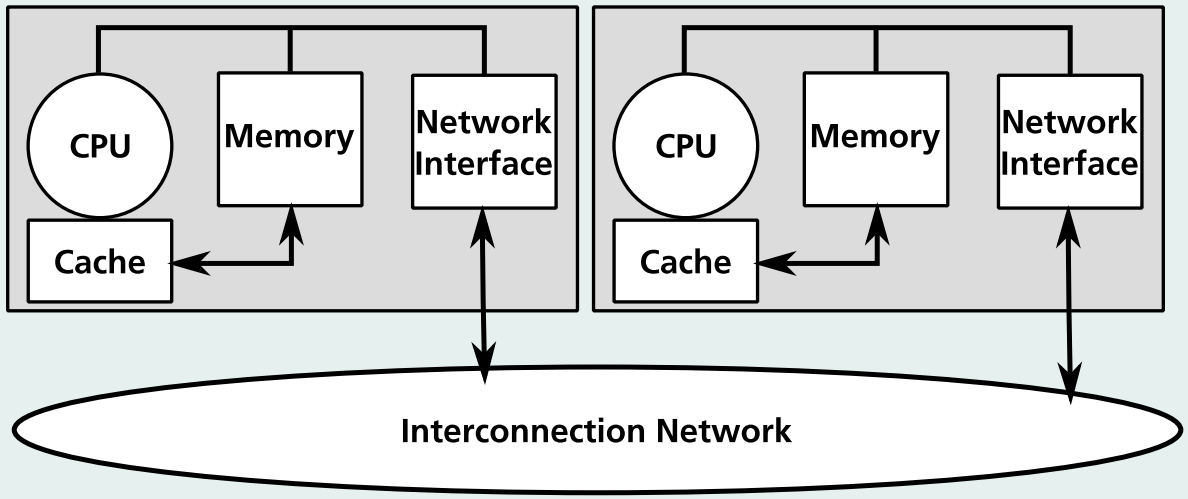


- UMA: Uniform Memory Access

Parallelismus: Parallele Architekturen

Multiprozessoren mit verteiltem Speicher

- **Verteilter Speicher $\Leftrightarrow$ Verteilter Adressraum**
- Beispiel: Cluster
- Nachrichtengekoppelter (Shared-nothing-) Multiprozessor

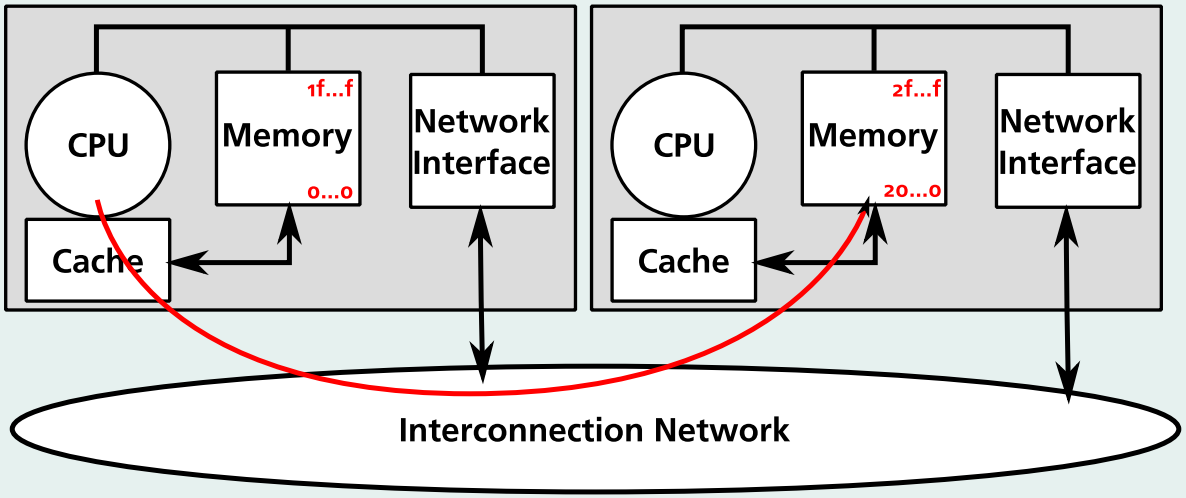


- NORMA: No Remote Memory Access

Parallelismus: Parallele Architekturen

Multiprozessoren mit verteiltem gemeinsamen Speicher

- **Verteilter Speicher $\Leftrightarrow$ Gemeinsamer Adressraum**
- Beispiel: Distributed-shared-memory Multiprozessor (DSM)



- NUMA: Non-Uniform Memory Access
- CC-NUMA: Cache-Coherent Non-Uniform Memory Access

Parallelismus: Parallele Architekturen

Konfigurationen von Multiprozessoren

	Globaler Speicher	Physikalisch verteilter Speicher
gemeinsamer Adreßraum	SMP Symmetrischer Multiprozessor	DSM Distributed-shared-memory Multiprozessor
	UMA: Uniform Memory Access	NUMA: Non-Uniform Memory Access
verteilte Adreßräume	leer	Nachrichtengekoppelter (Shared-nothing) Multiprozessor / Cluster
		NORMA: No Remote Memory Access

Parallelismus: Parallele Architekturen

Karlsruher Institut für Technologie

Konfigurationen von Multiprozessoren

	Globaler Speicher	Physikalisch verteilter Speicher
gemeinsamer Adreßraum	SMP Symmetrischer Multiprozessor	DSM Distributed-shared-memory Multiprozessor
	UMA: Uniform Memory Access	NUMA: Non-Uniform Memory Access
verteilte Adreßräume	leer (?) (Manycore-Prozessoren ?)	Nachrichtengekoppelter (Shared-Nothing) Multiprozessor / Cluster (Manycore-Prozessoren ?)
		NORMA: No Remote Memory Access

Parallelismus: Parallele Architekturen

Karlsruher Institut für Technologie

Programmiermodelle

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

$\Rightarrow$ Einführung erfolgt später in der Übung!

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Definitionen:

- $P(1)$ Anzahl der Einheitsoperationen auf einem Einprozessorsystem
- $P(n)$ Anzahl der Einheitsoperationen auf einem Multiprozessorsystem mit n Prozessoren
- $T(1)$ Ausführungszeit auf einem Einprozessorsystem in Schritten
- $T(n)$ Ausführungszeit auf einem Multiprozessorsystem mit n Prozessoren in Schritten

Vereinfachende Voraussetzungen:

- $T(1) = P(1)$
- $T(n) \leq P(n)$

Parallelismus: Quantitative Maßzahlen

Verbesserung der Verarbeitungsgeschwindigkeit

Beschleunigung (Speed-Up)

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

üblicherweise gilt:

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

Effizienz

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

daraus folgt:

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

Parallelismus: Quantitative Maßzahlen

Verbesserung der Verarbeitungsgeschwindigkeit

Beschleunigung (Speed-Up)

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

üblicherweise gilt:

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

Effizienz

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

daraus folgt:

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

Parallelismus: Quantitative Maßzahlen

Verbesserung der Verarbeitungsgeschwindigkeit

Beschleunigung (Speed-Up)

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

üblicherweise gilt:

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

Effizienz

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

daraus folgt:

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

Algorithmenunabhängige vs. Algorithmenabhängige Definition

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Algorithmenunabhängige Definition

- **Absolute Beschleunigung** und **absolute Effizienz** erhält man, indem der beste sequentielle Algorithmus mit dem besten parallelen Algorithmus verglichen wird.

Algorithmenabhängige Definition

- **Relative Beschleunigung** und **relative Effizienz** erhält man, wenn ein paralleler Algorithmus auf einem Einprozessorsystem ausgeführt wird.
- Zusatzaufwand für Parallelisierung durch Kommunikation und Synchronisation „verfälscht“ bei der sequentiellen Ausführung

Parallelismus: Quantitative Maßzahlen

Mehraufwand für die Parallelisierung

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

$$1 \leq R(n)$$

Mehraufwand für die Organisation, Synchronisation und Kommunikation der Prozessoren in Multiprozessorsystemen.

Parallelindex

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

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

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Mehraufwand für die Parallelisierung

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

$$1 \leq R(n)$$

Mehraufwand für die Organisation, Synchronisation und Kommunikation der Prozessoren in Multiprozessorsystemen.

Parallelindex

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

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

Mittlerer Grad der Parallelität. Anzahl der parallelen Operationen pro Zeiteinheit.

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Auslastung (Utilization)

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

Gibt an wieviel Operationen jeder Prozessor im Durchschnitt pro Zeiteinheit ausführt. Entspricht dem normierten Parallelindex.

Folgerungen

- 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:

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

Parallelismus: Quantitative Maßzahlen

Auslastung (Utilization)

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

Gibt an wieviel Operationen jeder Prozessor im Durchschnitt pro Zeiteinheit ausführt. Entspricht dem normierten Parallelindex.

Folgerungen

- 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:

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

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Superlinearer Speed-Up

$$S(n) > n \qquad T(n) < \frac{T(1)}{n}$$

Beispiele:

- **Paralleles Backtracking (depth-first search)**
  $\Rightarrow$ Verwendung verschiedener Algorithmen
- **Cache- und Hauptspeicherausnutzung**
  Nach der Aufteilung eines Problems auf verschiedene Teilsysteme, können eventuell jeweils die Teilprogramme und -daten komplett im Hauptspeicher bzw. Cache der einzelnen Knoten gehalten werden.
  $\Rightarrow$ kein Seitenwechsel mehr nötig

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Skalierbarkeit eines Parallelrechners

- Von Skalierbarkeit spricht man, wenn das Hinzufügen von weiteren Verarbeitungselementen zu einer kürzeren Gesamtausführungszeit führt, ohne dass das Programm geändert werden muß.
- lineare Steigerung der Beschleunigung, Effizienz nahe 1
- Wichtig für die Skalierbarkeit ist eine angemessene Problemgröße, da sonst ab einer bestimmten Prozessorenzahl eine Sättigung auftritt

- **Achtung:** Nicht verwechseln mit der Skalierbarkeit eines Verbindungsnetzes!

Parallelismus: Quantitative Maßzahlen

Amdahls Gesetz (Amdahl's Law)

$$T(n) = \underbrace{\frac{1}{n} * T(1) * (1 - a)}_{1} + \underbrace{T(1) * a}_{2}$$

- ***a*** mit $0 \leq a \leq 1$ ist der Anteil eines Programms, der nur sequentiell ausgeführt werden kann.

1. Ausführungszeit des parallel ausführbaren Programmteils $(1 - a)$
2. Ausführungszeit des nur sequentiell ausführbaren Programmteils a.

Parallelismus: Quantitative Maßzahlen

Amdahls Gesetz (Amdahl's Law)

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

Durch Einsetzen in die Formel für die Beschleunigung erhält man:

$$\begin{aligned} S(n) &= \frac{T(1)}{T(n)} = \frac{T(1)}{\frac{T(1)}{n} * (1 - a) + T(1) * a} \\ &= \frac{1}{(1 - a) * \frac{1}{n} + a} \\ S(n) &\leq \frac{1}{a} \end{aligned}$$

$\Rightarrow$ Die erreichbare Beschleunigung wird durch den sequentiellen Programmteil begrenzt.

Parallelismus: Quantitative Maßzahlen

Karlsruher Institut für Technologie

Achtung!

Alle auch hier nicht wiederholten Inhalte der Vorlesung sind für die Klausur relevant!

Aufgabe 1.1 – Leistungsbewertung

Karlsruher Institut für Technologie

Gegeben sei ein Multiprozessorsystem mit 16 Prozessoren. Die Leistungssteigerung gegenüber einem Einprozessorsystem sei $S(16) = 8$. Die Ausführungszeit auf dem Einprozessorsystem sei $T(1) = 800$ und die Anzahl der auszuführenden Einheitsoperationen auf dem Multiprozessorsystem sei $P(16) = 1200$.

a) Berechnen Sie die Effizienz $E(16)$, die parallele Ausführungszeit $T(16)$ und den Parallelindex $I(16)$.

b) Interpretieren Sie den berechneten Parallelindex.

c) Ermitteln Sie anhand von Amdahls Gesetz den Bruchteil des Programms, der nur sequentiell ausführbar ist.

Aufgabe 1.1 – Leistungsbewertung

a) **Berechnen Sie die Effizienz $E(16)$, die parallele Ausführungszeit $T(16)$ und den Parallelindex $I(16)$**

$$E(n) = \frac{S(n)}{n} \quad \Rightarrow \quad E(16) = \frac{S(16)}{16} = \frac{8}{16} = 0,5$$

$$S(n) = \frac{T(1)}{T(n)} \quad \Rightarrow \quad T(16) = \frac{T(1)}{S(16)} = \frac{800}{8} = 100$$

$$I(16) = \frac{P(n)}{T(n)} \quad \Rightarrow \quad I(16) = \frac{P(16)}{T(16)} = \frac{1200}{100} = 12$$

Aufgabe 1.1 – Leistungsbewertung

Karlsruher Institut für Technologie

b) Interpretieren Sie den berechneten Parallelindex

- Der Parallelindex gibt den mittleren Grad der Parallelität an, d.h. die Anzahl der parallelen Operationen pro Zeiteinheit.
- Der berechnete Wert ist kleiner als die Anzahl der Prozessoren. Das bedeutet, dass zeitweise einige Prozessoren keinen Befehl ausführen.
- Besser verdeutlicht wird dies durch Berechnung der Auslastung U, die angibt wieviel Operationen jeder Prozessor im Durchschnitt pro Zeiteinheit ausführt.

$$U(n) = \frac{I(n)}{n} \quad \Rightarrow \quad U(16) = \frac{12}{16} = \frac{3}{4} = 75\,\%$$

Aufgabe 1.1 – Leistungsbewertung

Karlsruher Institut für Technologie

b) Interpretieren Sie den berechneten Parallelindex

- Der Parallelindex gibt den mittleren Grad der Parallelität an, d.h. die Anzahl der parallelen Operationen pro Zeiteinheit.
- Der berechnete Wert ist kleiner als die Anzahl der Prozessoren. Das bedeutet, dass zeitweise einige Prozessoren keinen Befehl ausführen.
- Besser verdeutlicht wird dies durch Berechnung der Auslastung U, die angibt wieviel Operationen jeder Prozessor im Durchschnitt pro Zeiteinheit ausführt.

$$U(n) = \frac{I(n)}{n} \quad \Rightarrow \quad U(16) = \frac{12}{16} = \frac{3}{4} = 75\,\%$$

Aufgabe 1.1 – Leistungsbewertung

c) **Ermitteln Sie anhand von Amdahls Gesetz den Bruchteil des Programms, der nur sequentiell ausführbar ist.**

Ahmdahls Gesetz:

$$T(n) = T(1) * \left(\frac{(1-a)}{n} + a \right) = T(1) * \frac{1 - a + na}{n}$$

$\approx 6,7\,\%$ des Programmcodes sind nur sequentiell ausführbar.

Aufgabe 1.1 – Leistungsbewertung

c) **Ermitteln Sie anhand von Amdahls Gesetz den Bruchteil des Programms, der nur sequentiell ausführbar ist.**

Ahmdahls Gesetz:

$$T(n) = T(1) * \left(\frac{(1-a)}{n} + a \right) = T(1) * \frac{1 - a + na}{n}$$

$$\Rightarrow \quad 100 = 800 * \frac{1 - a + 16a}{16} = 800 * \frac{1 + 15a}{16}$$

$$= 50 * (1 + 15a)$$

$\approx 6,7\,\%$ des Programmcodes sind nur sequentiell ausführbar.

Aufgabe 1.1 – Leistungsbewertung

c) **Ermitteln Sie anhand von Amdahls Gesetz den Bruchteil des Programms, der nur sequentiell ausführbar ist.**

Ahmdahls Gesetz:

$$T(n) = T(1) * \left(\frac{(1-a)}{n} + a \right) = T(1) * \frac{1 - a + na}{n}$$

$$\Rightarrow \quad 100 = 800 * \frac{1 - a + 16a}{16} = 800 * \frac{1 + 15a}{16}$$

$$= 50 * (1 + 15a)$$

$$\Rightarrow \quad 15a = 1 \quad \Rightarrow \quad a = \frac{1}{15} \approx 6,7\,\%$$

$\approx 6,7\,\%$ des Programmcodes sind nur sequentiell ausführbar.

Aufgabe 1.2 – Leistungsbewertung

Karlsruher Institut für Technologie

Die Ausführungszeit einer sequentiellen Anwendung betrage T_{seq}. Von dieser Anwendung lassen sich 20 % nicht parallelisieren. Die verbleibenden 80 % werden zwischen den Prozessoren gleichmäßig verteilt. Das bedeutet, dass jeder Prozessor ungefähr den gleichen Anteil der zu parallelisierenden Aufgabe bearbeitet und jeder gleichviel Zeit benötigt.

Beispielsweise betrage die Ausführungszeit der parallelen Anteile der Anwendung 20 % der sequentiellen Ausführungszeit, wenn vier Prozessoren sie ausführen.

a) Setzen Sie voraus, dass die Parallelisierung keinen Aufwand verursacht. Berechnen Sie die Beschleunigung und die Effizienz bei einer unterschiedlichen Anzahl von Prozessoren. Fügen Sie die Ergebnisse in die vorgegebene Tabelle ein.

Aufgabe 1.2 – Leistungsbewertung

Karlsruher Institut für Technologie

a) **Setzen Sie voraus, dass die Parallelisierung keinen Aufwand verursacht. Berechnen Sie die Beschleunigung und die Effizienz bei einer unterschiedlichen Anzahl von Prozessoren. Fügen Sie die Ergebnisse in die vorgegebene Tabelle ein.**

Par. Ausführungszeit: $T(n) = \frac{80\,\%}{n} * T_{seq} + 20\,\% * T_{seq}$

Beschleunigung: $S(n) = \frac{T_{seq}}{T(n)}$

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

Prozessoren	n=2	n=4	n=8	n=16	n=32
Beschleunigung	5/3=1,67	5/2=2,50	10/3=3,33	4	40/9=4,44
Effizienz	5/6=0,83	5/8=0,625	10/24=0,42	1/4=0,25	5/36=0,14

Achtung: In der Klausur bei Berechnungen wenn möglich Brüche statt gerundete Zahlen als Ergebnisse angeben!

Aufgabe 1.2 – Leistungsbewertung

Karlsruher Institut für Technologie

a) **Setzen Sie voraus, dass die Parallelisierung keinen Aufwand verursacht. Berechnen Sie die Beschleunigung und die Effizienz bei einer unterschiedlichen Anzahl von Prozessoren. Fügen Sie die Ergebnisse in die vorgegebene Tabelle ein.**

Par. Ausführungszeit: $T(n) = \frac{80\,\%}{n} * T_{seq} + 20\,\% * T_{seq}$

Beschleunigung: $S(n) = \frac{T_{seq}}{T(n)}$

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

Prozessoren	n=2	n=4	n=8	n=16	n=32
Beschleunigung	5/3=1,67	5/2=2,50	10/3=3,33	4	40/9=4,44
Effizienz	5/6=0,83	5/8=0,625	10/24=0,42	1/4=0,25	5/36=0,14

Achtung: In der Klausur bei Berechnungen wenn möglich Brüche statt gerundete Zahlen als Ergebnisse angeben!

Aufgabe 1.2 – Leistungsbewertung

Prozessoren	n=2	n=4	n=8	n=16	n=32
Beschleunigung	5/3=1,67	5/2=2,50	10/3=3,33	4	40/9=4,44
Effizienz	5/6=0,83	5/8=0,625	10/24=0,42	1/4=0,25	5/36=0,14

b) **Evaluieren Sie zusätzlich die Skalierbarkeit der einzelnen Lösungen**

Die Skalierbarkeit ist sehr schlecht. Grund dafür ist, dass ein großer Anteil des Programms nicht parallelisierbar ist. Die Ausführungszeit dieses Anteils bestimmt mit steigender Anzahl von Prozessoren einen zunehmenden Anteil der gesamten Ausführungszeit.

Beschleunigung: $S(n) = \frac{T_{seq}}{(20\,\% + \frac{80\,\%}{n}) * T_{seq}} \xrightarrow{n \text{ sehr groß}} \frac{1}{20\,\%} = 5$

Die Effizienz strebt damit für große n gegen 0.

Aufgabe 1.2 – Leistungsbewertung

Prozessoren	n=2	n=4	n=8	n=16	n=32
Beschleunigung	5/3=1,67	5/2=2,50	10/3=3,33	4	40/9=4,44
Effizienz	5/6=0,83	5/8=0,625	10/24=0,42	1/4=0,25	5/36=0,14

b) **Evaluieren Sie zusätzlich die Skalierbarkeit der einzelnen Lösungen**

Die Skalierbarkeit ist sehr schlecht. Grund dafür ist, dass ein großer Anteil des Programms nicht parallelisierbar ist. Die Ausführungszeit dieses Anteils bestimmt mit steigender Anzahl von Prozessoren einen zunehmenden Anteil der gesamten Ausführungszeit.

Beschleunigung: $S(n) = \frac{T_{seq}}{\left(20\,\% + \frac{80\,\%}{n}\right) * T_{seq}} \xrightarrow{n \text{ sehr groß}} \frac{1}{20\,\%} = 5$

Die Effizienz strebt damit für große n gegen 0.

Aufgabe 1.2 – Leistungsbewertung

c) **Zur Steigerung der Genauigkeit der Berechnung nehmen Sie nun an, dass durch jeden verwendeten Prozessor eine zusätzliche Bearbeitungszeit von 1 % der ursprünglichen sequentiellen Ausführungszeit benötigt wird. Berechnen Sie Beschleunigung und Effizienz für 64 Prozessoren.**

Beschleunigung:

$$S(n) = \frac{T_{seq}}{\left(20\,\% + \frac{80\,\%}{n} + 1\,\% * n\right) * T_{seq}}$$

$$\Rightarrow \quad S(64) \approx \quad 1,17$$

Effizienz:

$$E(64) = \frac{S(64)}{64} = \frac{1,17}{64} = 0,018$$

Aufgabe 1.2 – Leistungsbewertung

Karlsruher Institut für Technologie

c) **Zur Steigerung der Genauigkeit der Berechnung nehmen Sie nun an, dass durch jeden verwendeten Prozessor eine zusätzliche Bearbeitungszeit von 1 % der ursprünglichen sequentiellen Ausführungszeit benötigt wird. Berechnen Sie Beschleunigung und Effizienz für 64 Prozessoren.**

Beschleunigung:

$$S(n) = \frac{T_{seq}}{\left(20\,\% + \frac{80\,\%}{n} + 1\,\% * n\right) * T_{seq}}$$

$$\Rightarrow \quad S(64) \approx \quad 1,17$$

Effizienz:

$$E(64) = \frac{S(64)}{64} = \frac{1,17}{64} = 0,018$$

Aufgabe 1.3 – Leistungsbewertung

Karlsruher Institut für Technologie

Ein Einprozessorsystem soll erweitert werden. Dabei existieren folgende, in der Anschaffung gleich teure Alternativen:

- Ausbau zu einem 2-fach SMP-System
  - Installation eines zweiten identischen Hauptprozessors
  - Zugriff auf einen gemeinsamen Hauptspeicher
  - Synchronisationsoverhead: Ausführung des Programms wird um 2 % der unparallelisierten Ausführungszeit erhöht.
- Einsetzen eines mathematischen Koprozessors
  - 10x schnellere Ausführung der Gleitkommaarithmetik
  - Keine parallele Verarbeitung, d.h. der gleichzeitige Einsatz von Haupt- und Koprozessor, möglich.

Das zu bearbeitende Problem ist zu 74 % parallelisierbar. Der Anteil der Gleitkommaarithmetik am Gesamtprogramm beträgt 40 %. Bestimmen Sie, welche der beiden Möglichkeiten unter dem Gesichtspunkt der Ausführungszeit zu bevorzugen ist.

Aufgabe 1.3 – Leistungsbewertung

Karlsruher Institut für Technologie

Die Ausführungszeiten lassen sich in beiden Fällen wiederum mit Hilfe von Ahmdahls Gesetz ausrechnen und damit auch die erreichte Beschleunigung vergleichen. T_{seq} ist hierbei die Zeit ohne Parallelisierung oder Koprozessorunterstützung.

- SMP-System:

$$T_{SMP} = 74\,\% * \frac{1}{2} * T_{seq} + 26\,\% * T_{seq} + 2\,\% * T_{seq} = 65\,\% * T_{seq}$$

$$S_{SMP} = \frac{T_{seq}}{T_{SMP}} = \frac{1}{65\,\%} \approx 1,5385$$

- System mit Koprozessor:

$$T_{CP} = 40\,\% * \frac{1}{10} * T_{seq} + 60\,\% * T_{seq} = 64\,\% * T_{seq}$$

$$S_{CP} = \frac{T_{seq}}{T_{CP}} = \frac{1}{64\,\%} \approx 1,5625$$

Aufgabe 1.3 – Leistungsbewertung

Karlsruher Institut für Technologie

Die Ausführungszeiten lassen sich in beiden Fällen wiederum mit Hilfe von Ahmdahls Gesetz ausrechnen und damit auch die erreichte Beschleunigung vergleichen. T_{seq} ist hierbei die Zeit ohne Parallelisierung oder Koprozessorunterstützung.

- SMP-System:

$$T_{SMP} = 74\,\% * \frac{1}{2} * T_{seq} + 26\,\% * T_{seq} + 2\,\% * T_{seq} = 65\,\% * T_{seq}$$

$$S_{SMP} = \frac{T_{seq}}{T_{SMP}} = \frac{1}{65\,\%} \approx 1,5385$$

- System mit Koprozessor:

$$T_{CP} = 40\,\% * \frac{1}{10} * T_{seq} + 60\,\% * T_{seq} = 64\,\% * T_{seq}$$

$$S_{CP} = \frac{T_{seq}}{T_{CP}} = \frac{1}{64\,\%} \approx 1,5625$$

Aufgabe 1.3 – Leistungsbewertung

Karlsruher Institut für Technologie

Die Ausführungszeiten lassen sich in beiden Fällen wiederum mit Hilfe von Ahmdahls Gesetz ausrechnen und damit auch die erreichte Beschleunigung vergleichen. T_{seq} ist hierbei die Zeit ohne Parallelisierung oder Koprozessorunterstützung.

- SMP-System:

$$T_{SMP} = 74\,\% * \frac{1}{2} * T_{seq} + 26\,\% * T_{seq} + 2\,\% * T_{seq} = 65\,\% * T_{seq}$$

$$S_{SMP} = \frac{T_{seq}}{T_{SMP}} = \frac{1}{65\,\%} \approx 1,5385$$

- System mit Koprozessor:

$$T_{CP} = 40\,\% * \frac{1}{10} * T_{seq} + 60\,\% * T_{seq} = 64\,\% * T_{seq}$$

$$S_{CP} = \frac{T_{seq}}{T_{CP}} = \frac{1}{64\,\%} \approx 1,5625$$

Aufgabe 1.3 – Leistungsbewertung

- SMP-System:

$$S_{SMP} \approx 1,5385$$

- System mit Koprozessor:

$$S_{CP} \approx 1,5625$$

$\Rightarrow$ Die Koprozessorlösung ist für diese Anwendung deshalb dem SMP-System vorzuziehen.

$\Rightarrow$ Ohne Beachtung der benötigten Synchronisationszeit würde die Berechnung ein fehlerhaftes Ergebnis liefern.

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

- **Ziel: Schnellere Lösung der parallelen Version gegenüber der sequentiellen Version**

- Festlegen der Aufgaben, die parallel ausgeführt werden können
- Aufteilen der Aufgaben und der Daten auf Verarbeitungsknoten
  - Berechnung
  - Datenzugriff
  - Ein-/Ausgabe
- Verwalten des Datenzugriffs, der Kommunikation und Synchronisation

- Programmierer oder Automatische Parallelisierung

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess – Definitionen

- **Tasks**
  - Kleinste Parallelisierungseinheit
  - Beispiel: Berechnung eines Gitterpunkts (oder einer Teilmenge von Punkten)
  - grobkörnig vs. feinkörnig
- **Prozess** oder Thread
  - Paralleles Programm setzt sich aus mehreren kooperierenden Prozessen zusammen, von denen jeder eine Teilmenge der Tasks ausführt
  - Kommunikation und Synchronisation der Prozesse untereinander
  - Virtualisierung von einem Multiprozessor, Abstraktion

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess – Definitionen

- **Tasks**
  - Kleinste Parallelisierungseinheit
  - Beispiel: Berechnung eines Gitterpunkts (oder einer Teilmenge von Punkten)
  - grobkörnig vs. feinkörnig
- **Prozess** oder Thread
  - Paralleles Programm setzt sich aus mehreren kooperierenden Prozessen zusammen, von denen jeder eine Teilmenge der Tasks ausführt
  - Kommunikation und Synchronisation der Prozesse untereinander
  - Virtualisierung von einem Multiprozessor, Abstraktion

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess – Definitionen

- **Prozessor**
  - Ausführung eines oder mehrerer Prozesse
  - Physikalische Ressource

- **Unterscheidung zwischen Prozess und Prozessor!**

$\Rightarrow$ Anzahl der Prozesse muss nicht gleich der Anzahl der Prozessoren eines Multiprozessorsystems sein

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess – Definitionen

- **Prozessor**
  - Ausführung eines oder mehrerer Prozesse
  - Physikalische Ressource

- **Unterscheidung zwischen Prozess und Prozessor!**

$\Rightarrow$ Anzahl der Prozesse muss nicht gleich der Anzahl der Prozessoren eines Multiprozessorsystems sein

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

Schritte bei der Parallelisierung:

- Ausgangspunkt ist ein sequentielles Programm

1. **Dekomposition** (oder Aufteilung) der Berechnung in Tasks
   - Granularität, möglicherweise dynamische Anzahl an Tasks
2. **Zuweisung** der Tasks zu Prozessen
   - Lastverteilung, geringe Kommunikation
3. **Zusammenführung (Orchestration)** Festlegung des notwendigen Datenzugriffs, der Kommunikation und der Synchronisation zwischen den Prozessen
   - Auswahl des passenden Programmiermodells
4. **Abbildung** der Prozesse auf die Prozessoren

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

Schritte bei der Parallelisierung:

- Ausgangspunkt ist ein sequentielles Programm

1. **Dekomposition** (oder Aufteilung) der Berechnung in Tasks
   - Granularität, möglicherweise dynamische Anzahl an Tasks
2. **Zuweisung** der Tasks zu Prozessen
   - Lastverteilung, geringe Kommunikation
3. **Zusammenführung (Orchestration)** Festlegung des notwendigen Datenzugriffs, der Kommunikation und der Synchronisation zwischen den Prozessen
   - Auswahl des passenden Programmiermodells
4. **Abbildung** der Prozesse auf die Prozessoren

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

Schritte bei der Parallelisierung:

- Ausgangspunkt ist ein sequentielles Programm

1. **Dekomposition** (oder Aufteilung) der Berechnung in Tasks
   - Granularität, möglicherweise dynamische Anzahl an Tasks
2. **Zuweisung** der Tasks zu Prozessen
   - Lastverteilung, geringe Kommunikation
3. **Zusammenführung (Orchestration)** Festlegung des notwendigen Datenzugriffs, der Kommunikation und der Synchronisation zwischen den Prozessen
   - Auswahl des passenden Programmiermodells
4. **Abbildung** der Prozesse auf die Prozessoren

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

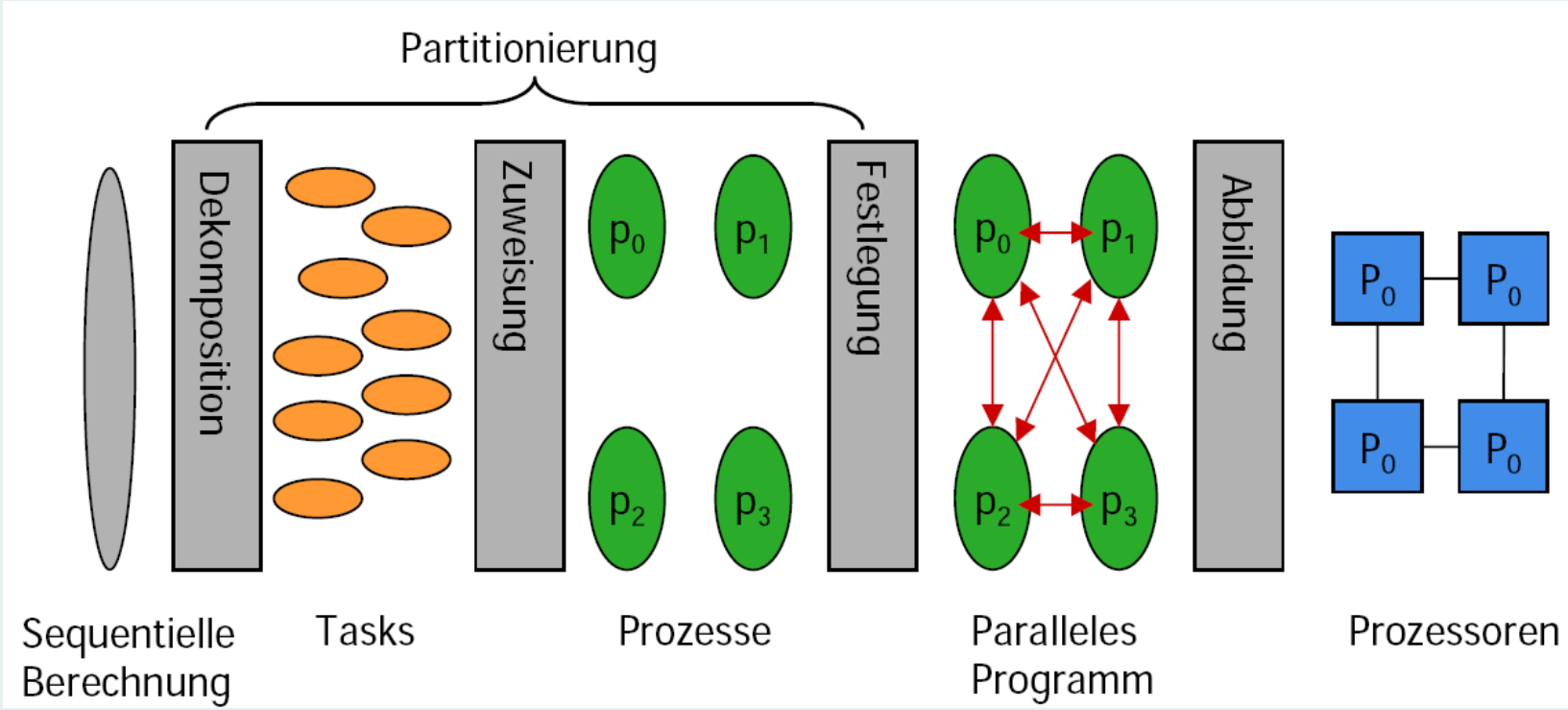
Schritte bei der Parallelisierung:

- Ausgangspunkt ist ein sequentielles Programm

1. **Dekomposition** (oder Aufteilung) der Berechnung in Tasks
   - Granularität, möglicherweise dynamische Anzahl an Tasks
2. **Zuweisung** der Tasks zu Prozessen
   - Lastverteilung, geringe Kommunikation
3. **Zusammenführung (Orchestration)** Festlegung des notwendigen Datenzugriffs, der Kommunikation und der Synchronisation zwischen den Prozessen
   - Auswahl des passenden Programmiermodells
4. **Abbildung** der Prozesse auf die Prozessoren

Parallele Programmierung – Einführung

Parallelisierungsprozess



Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Parallelisierungsprozess

- Dekomposition, Zuweisung und Festlegung werden zusammen auch als **Partitionierung** bezeichnet
- Granularität beachten!

$\Rightarrow$ Verwaltungsaufwand (Overhead) gering halten

$\Rightarrow$ Vgl. Amdahls Gesetz

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Programmiermodelle

- Definition einer abstrakten parallelen Maschine
- Spezifikationen
  - Parallele Abarbeitung von Teilen des Programms
  - Informationsaustausch
  - Synchronisationsoperationen zur Koordination
- Anwendungen werden auf der Grundlage eines parallelen Programmiermodells formuliert

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Multiprogramming

- Menge von unabhängigen sequentiellen Programmen
- Keine Kommunikation oder Koordination
- **Aber: Mögliche gegenseitige Beeinflussung beim gleichzeitigen Zugriff auf den Speicher!**

Verwendung:

- Auf allen Rechnern, die „gleichzeitig“ verschiedene Programme ausführen können (multitasking-fähiges Betriebssystem)

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Gemeinsamer Speicher (Shared Memory)

- Kommunikation und Koordination von Prozessen über gemeinsame Variablen
- Atomare Synchronisationsoperationen
- Semaphoren, Mutex, Monitore, Transactional Memory,...

Verwendung:

- Symmetrischer Multiprozessor (SMP)
- Distributed-shared-memory Multiprozessor (DSM)

Speicherzugriff:

- Uniform Memory Access (UMA)
- Non-Uniform Memory Access (NUMA), CC-NUMA

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Message Passing

- Nachrichtenorientiertes Programmiermodell
- Kein gemeinsamer Adressraum
- Kommunikation der Prozesse mit Hilfe von Nachrichten
- Verwendung von korrespondierenden Send- und Receive-Operationen

Verwendung:

- Cluster
- Nachrichtengekoppelter (shared-nothing-) Multiprozessor

Speicherzugriff:

- No Remote Memory Access (NORMA)

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Shared Memory und Message Passing

- Mischung der Programmiermodelle
- **Cluster mit SMP-/DSM-Knoten** (Multicore-CPUs oder Multiprozessor-System mit gemeinsamem Speicher)
- Shared-Memory-Programmiermodell innerhalb eines Knotens
- Nachrichtenorientiertes Programmiermodell zwischen den Knoten

Parall. Programm. – Programmiermodelle

Karlsruher Institut für Technologie

Datenparallelismus

- Gleichzeitige Ausführung von Operationen auf getrennten Elementen einer Datenmenge (Feld, Vektor, Matrix)

Verwendung:

- Typischerweise in Vektorrechnern

$\Rightarrow$ Übung 8

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

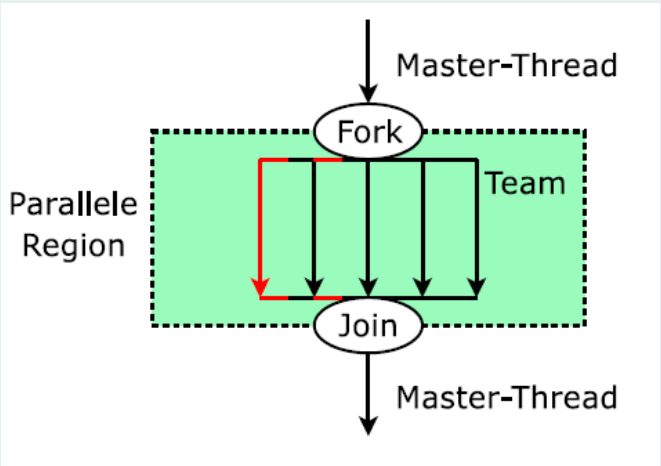
Thread-Programmierung

- Parallele Programme für Shared-Memory-Systeme bestehen aus mehreren Threads
- Alle Threads eines Prozesses teilen sich Adressraum, Daten, Filehandler,. . .
- Threads werden meist vom Betriebsystem verwaltet
- Unterstützung vom Betriebsystem notwendig, z.B. für die Thread-Generierung, Synchronisation,. . .
- Explizite Synchronisation notwendig!
- **Vorteil**: durch die Thread-Bibliothek erhält man eine detailierte Kontrolle über die Threads
- **Nachteil**: die Thread-Bibliothek erzwingt, dass man die Kontrolle über die Threads übernimmt

Parallele Programmierung – Einführung

OpenMP

- OpenMP ist eine offene Spezifikation von Übersetzerdirektiven, Bibliotheken und Umgebungsvariablen, spezifiziert für die Parallelisierung von Programmen auf gemeinsamen Speicher
- Verwendet das Join-Fork-Modell



- Mehrere (auch verschaltelte) parallele Regionen pro Programm sind zugelassen

Karlsruher Institut für Technologie

OpenMP – Beispiele

- Bibliothek und Funktionen:
  - `#include <omp.h>`
  - `omp_set_num_threads(NUM_THREADS)`
  - `omp_get_num_threads()`
  - `omp_get_max_threads()`
- Pragmas:
  - `#pragma omp parallel`
  - `#pragma omp parallel for`
  - `#pragma omp parallel for private(x) shared(delta_x) reduction(+:sum)`
  - `#pragma omp critical`
  - `#pragma omp atomic`

Parallele Programmierung – π-Berechnung

Karlsruher Institut für Technologie

OpenMP

```
#include <omp.h>
static long num_steps = 100000;        double step;
#define NUM_THREADS 2
void main ()
{       int i;   double x, pi, sum = 0.0;
        step = 1.0/(double) num_steps;
        omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for reduction(+:sum) private(x)
        for (i=1;i<= num_steps; i++){
                x = (i-0.5)*step;
                sum = sum + 4.0/(1.0+x*x);
        }
        pi = step * sum;
}
```

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Message Passing Interface (MPI)

- MPI ist ein Standard für die nachrichtenbasierte Kommunikation in einem Multiprozessorsystem
- Nachrichtenbasierter Ansatz gewährleistet eine gute Skalierbarkeit
- Bibliotheksfunktionen koordinieren die Ausführung von mehreren Prozessen, sowie Verteilung von Daten, per Default keine gemeinsamen Daten
- Single Program Multiple Data (SPMD) Ansatz

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Message Passing Interface (MPI)

- Bibliothek und Funktionen:
  - `#include <mpi.h>`
  - Initialisierung

    `MPI_Init(&argc, &argv)`
  - `MPI_COMM_WORLD`

    beinhaltet alle am Programm beteiligten Prozesse
  - Prozessnummer abfragen:

    `MPI_Comm_rank(MPI_Comm comm, int *rank)`

    `MPI_Comm_rank(MPI_COMM_WORLD, &rank)`
  - Anzahl an Prozesse:

    `MPI_Comm_size(MPI_Comm comm, int *size)`

    `MPI_Comm_size(MPI_COMM_WORLD, &size)`
  - `MPI_Finalize()`

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Message Passing Interface (MPI)

- Kommunikation und Synchronisation:
  - Senden:
    `MPI_Send`
  - Senden (nicht blockierend):
    `MPI_Isend`
  - Gepuffert:
    `MPI_Bsend, MPI_Ibsend`
  - ready/synchron:
    `MPI_Rsend, MPI_Ssend, MPI_Irsend, MPI_Issend`
  - Empfangen:
    `MPI_Recv, MPI_Irecv`
  - `MPI_Sendrecv`

Parallele Programmierung – Einführung

Karlsruher Institut für Technologie

Message Passing Interface (MPI)

- Kommunikation und Synchronisation:
  - Warten auf alle Prozesse in `comm`:
    `MPI_Barrier(comm)`
  - Verteilen von Daten:
    `MPI_Bcast, MPI_Scatter`
  - Sammeln von Daten:
    `MPI_Gather, MPI_Reduce`
  - Empfangstests einer Sende- oder Empfangsoperation:
    `MPI_Probe, MPI_Test, MPI_Wait`

Parallele Programmierung – π-Berechnung

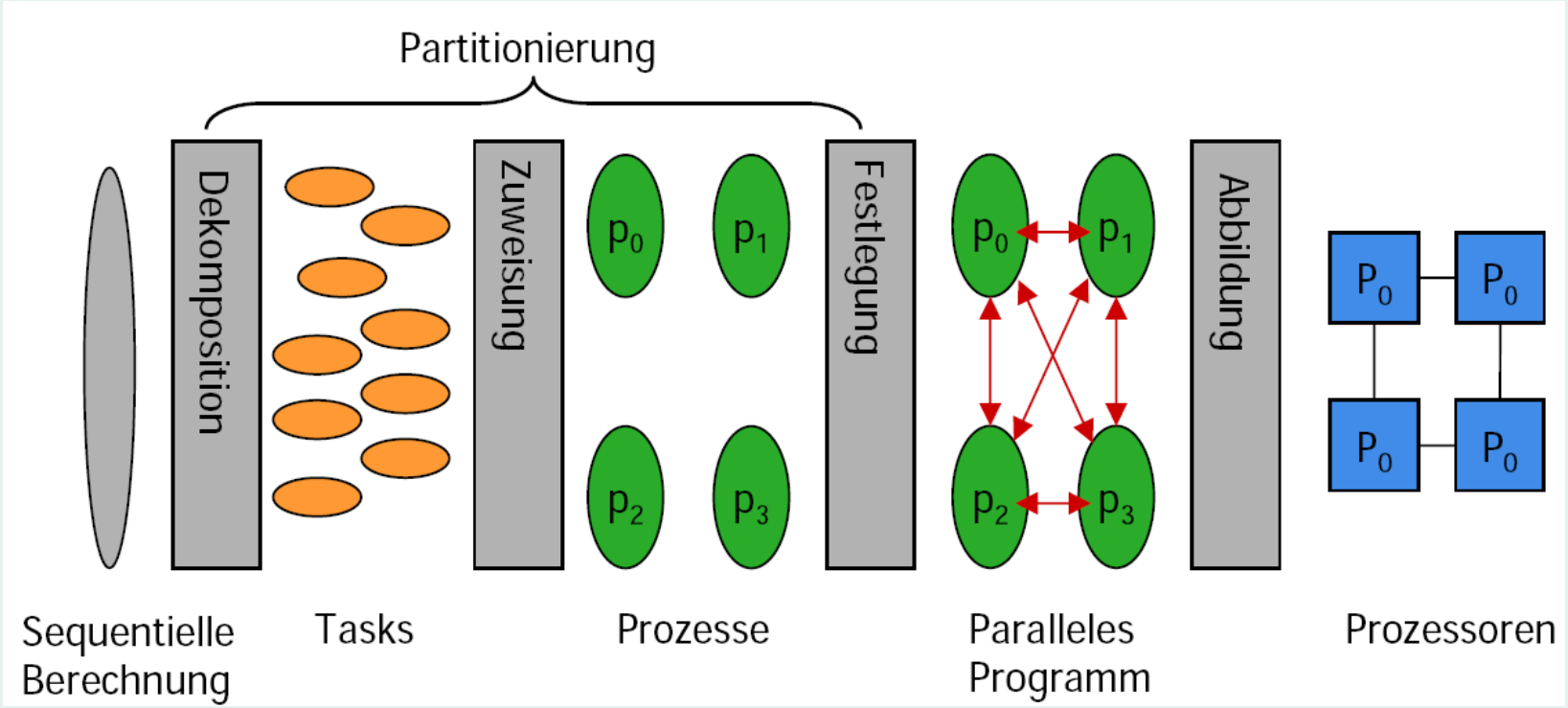
Karlsruher Institut für Technologie

MPI

```
#include <mpi.h>
void main (int argc, char *argv[])
{
        int i, my_id, numprocs;  double x, pi, step, sum = 0.0 ;
        step = 1.0/(double) num_steps ;
        MPI_Init(&argc, &argv) ;
        MPI_Comm_Rank(MPI_COMM_WORLD, &my_id) ;
        MPI_Comm_Size(MPI_COMM_WORLD, &numprocs) ;
        my_steps = num_steps/numprocs ;
        for (i=my_id*my_steps; i<(my_id+1)*my_steps ; i++)
        {
                x = (i+0.5)*step;
                sum += 4.0/(1.0+x*x);
        }
        sum *= step ;
        MPI_Reduce(&sum, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
                        MPI_COMM_WORLD) ;
}
```

Aufgabe 2.1 – Parallelisierungsprozess

Parallelisierungsprozess



Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

Eine Methode aus der Numerischen Mathematik arbeitet auf einem 2D-Torus mit $n * n$ Knoten.

In jeder Iteration werden drei Schritte durchgeführt:

1. Zustände aus dem Speicher lesen
2. Berechnung der neuen Zustände der Knoten, basierend auf ihrem aktuellen Zustand und den Zuständen ihrer Nachbarknoten
3. Zurückschreiben der neuen Zustände in den Speicher

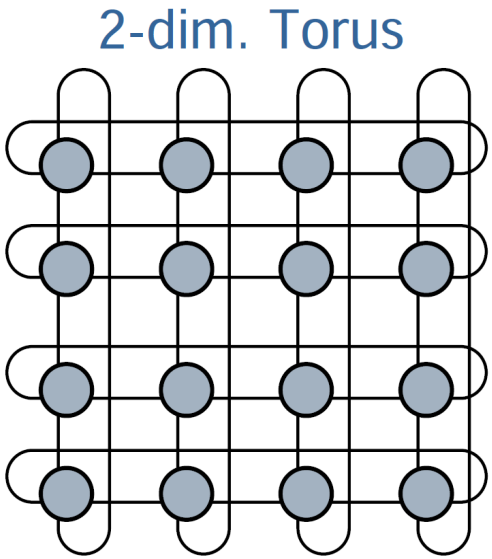
Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

Gegeben sei ein SMP-Knoten mit P Prozessoren.

a) Beachten Sie, dass in diesem System nur immer ein Lese- oder Schreibzugriff ausgeführt werden kann. Setzen Sie zusätzlich voraus, dass sich die Lese- und Schreibzugriffe nicht mit den Berechnungen überlappen (z.B. wegen einer vorherigen Synchronisation).
Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?

Aufgabe 2.2 – Parallelisierung



SMP gemeinsamer Adressraum, globaler Speicher

$\Rightarrow$ meist **UMA** (Uniform Memory Access), d.h. gleiche Zugriffszeit von allen Knoten

DSM gemeinsamer Adressraum, physikalisch verteilter Speicher

$\Rightarrow$ meist **NUMA** (Non-Uniform Memory Access), d.h. unterschiedliche Zugriffszeiten

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

a) **Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?**

- Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

- Parallele Zeit $T(P)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\text{par. Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 6n^2 + \frac{n^2}{P}$$

- Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{6n^2 + \frac{n^2}{P}} = \frac{7}{6 + \frac{1}{P}} < \frac{7}{6} = 1,1\overline{6}$$

Aufgabe 2.2 – Parallelisierung

a) **Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?**

- Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

- Parallele Zeit $T(P)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\text{par. Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 6n^2 + \frac{n^2}{P}$$

- Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{6n^2 + \frac{n^2}{P}} = \frac{7}{6 + \frac{1}{P}} < \frac{7}{6} = 1,1\overline{6}$$

Aufgabe 2.2 – Parallelisierung

a) **Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?**

- Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

- Parallele Zeit $T(P)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\text{par. Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 6n^2 + \frac{n^2}{P}$$

- Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{6n^2 + \frac{n^2}{P}} = \frac{7}{6 + \frac{1}{P}} < \frac{7}{6} = 1,1\overline{6}$$

Aufgabe 2.2 – Parallelisierung

a) **Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?**

Das Problem ist hierbei, dass der Speicher als limitierender Faktor wirkt und damit die Beschleunigung stark beschränkt ist.

In der Realität benötigten Synchronisationen jedoch in SMP-Systemen ebenfalls Speicherzugriffe von allen Prozessoren und damit sehr viel Zeit! Die erreichbare Beschleunigung wäre deshalb noch geringer.

Aufgabe 2.2 – Parallelisierung

a) **Berechnen Sie die Beschleunigung der Anwendung bei der Ausführung auf dem SMP-Knoten. Welches Problem tritt in dem SMP-System auf?**

Das Problem ist hierbei, dass der Speicher als limitierender Faktor wirkt und damit die Beschleunigung stark beschränkt ist.

In der Realität benötigten Synchronisationen jedoch in SMP-Systemen ebenfalls Speicherzugriffe von allen Prozessoren und damit sehr viel Zeit! Die erreichbare Beschleunigung wäre deshalb noch geringer.

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

b) **Was ändert sich, wenn auf eine Synchronisation vor der Berechnung verzichtet wird?**

Durch ein Verzicht auf die Synchronisation kann die Berechnung auf den Prozessoren mit den Datenzugriffen überlappend erfolgen.

Bei der parallelen Ausführungszeit spart man sich dadurch die Zeit für die Berechnung $\frac{n^2}{P}$ ein. Dies gilt aber nur, wenn die Berechnungszeit kürzer ist, als die Zeit für die Datenzugriffe.

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

b) **Was ändert sich, wenn auf eine Synchronisation vor der Berechnung verzichtet wird?**

Es gilt dann:

$$T(P) = 5n^2 + n^2 = 6n^2$$

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{6n^2} = \frac{7}{6} = 1,1\overline{6}$$

Insgesamt überwiegt auch hier die Zeit für die nicht parallelisierbaren Lese- und Schreibzugriffe. Der gemeinsame Speicher des SMP-Systems ist der limitierende Faktor.

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

Um eine Leistungssteigerung zu erhalten, wird der SMP-Knoten durch eine NUMA-Architektur mit P Prozessoren und P Speichern ersetzt.

Beachten Sie dabei folgende vereinfachende Annahme:

- Erfolgt ein Speicherzugriff auf einen entfernten Speicher, so benötigt ein Speichergriff drei Zeiteinheiten.
- Wird nur ein Prozessor verwendet, so befinden sich alle zur Berechnung notwendigen Daten im lokal zum Prozessor gehörenden Speicher.

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

- Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

- Parallele Zeit $T(P)$:

$$\underbrace{\underbrace{\frac{4 * 3 * n^2}{P}}_{\substack{\text{4 Werte der Nachbarknoten aus entferntem Speicher}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{eigener Wert im lokalen Speicher}}}}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{parallele Berechnung}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{Zurückschreiben in lokalen Speicher}}} = \frac{15n^2}{P}$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

- Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

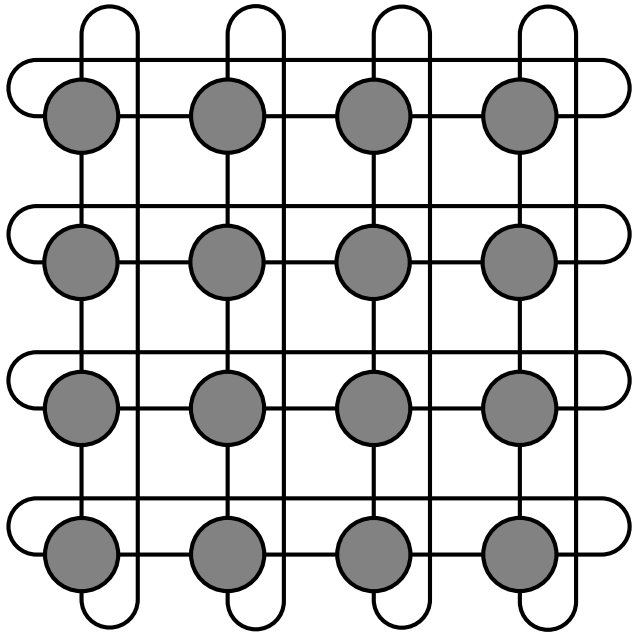
- Parallele Zeit $T(P)$:

$$\underbrace{\underbrace{\frac{4 * 3 * n^2}{P}}_{\substack{\text{4 Werte der Nachbarknoten aus entferntem Speicher}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{eigener Wert im lokalen Speicher}}}}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{parallele Berechnung}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{Zurückschreiben in lokalen Speicher}}} = \frac{15n^2}{P}$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

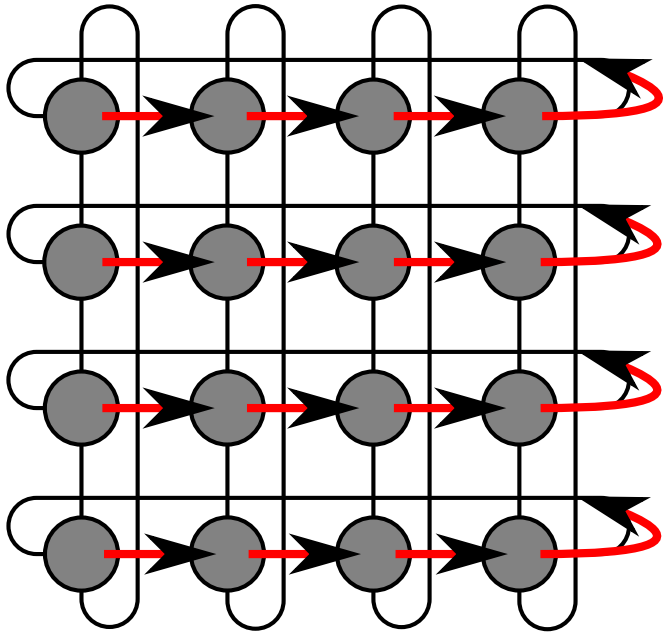


- Parallele Zeit $T(P)$ für $n^2 = P$:

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**



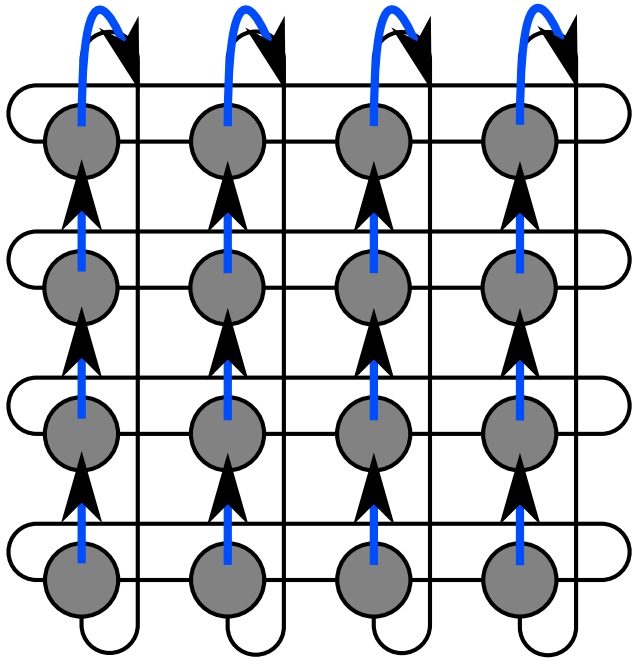
- Parallele Zeit $T(P)$ für $n^2 = P$:

$$3$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

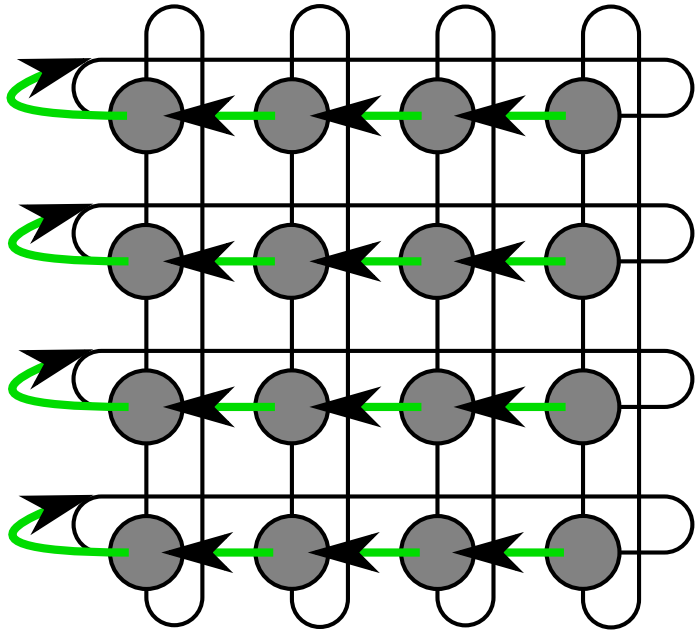


- Parallele Zeit $T(P)$ für $n^2 = P$:

$$3 + 3$$

Aufgabe 2.2 – Parallelisierung

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**



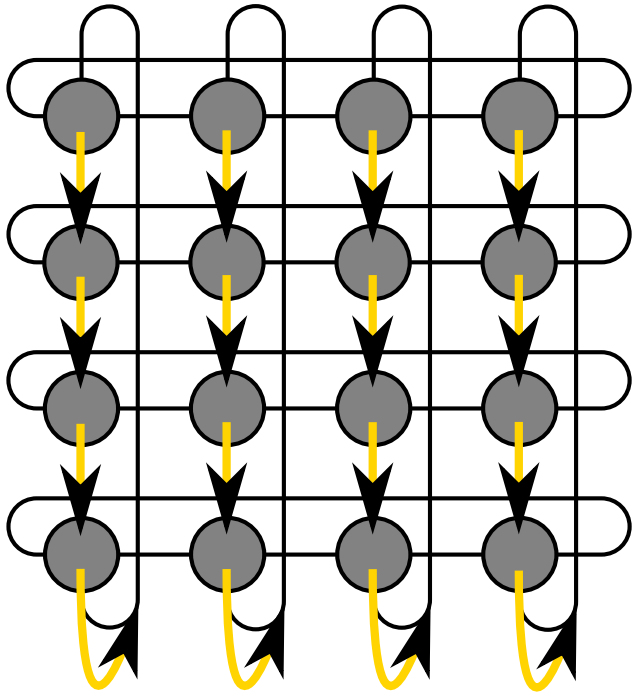
- Parallele Zeit $T(P)$ für $n^2 = P$:

$$3 + 3 + 3$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zuständе der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

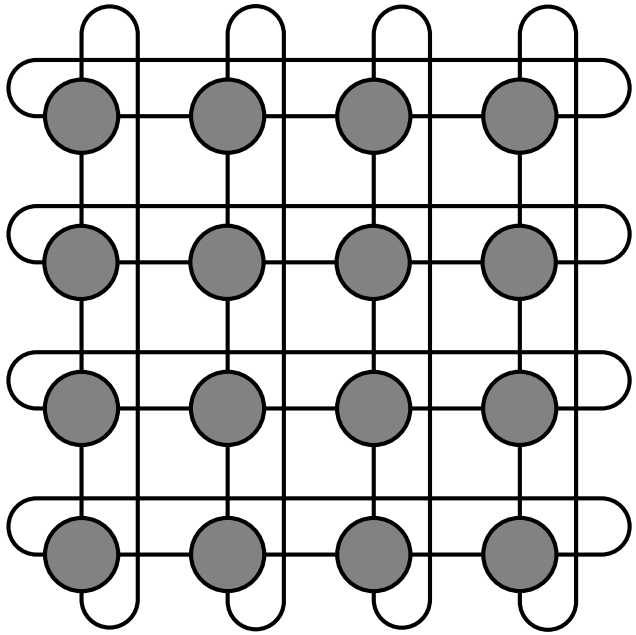


- Parallele Zeit $T(P)$ für $n^2 = P$:

$$3 + 3 + 3 + 3$$

Aufgabe 2.2 – Parallelisierung

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**



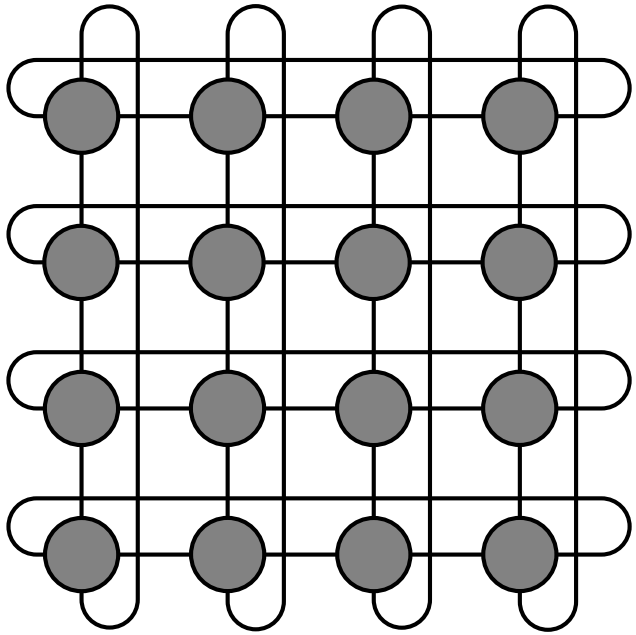
- Parallele Zeit $T(P)$ für $n^2 = P$:

$$3 + 3 + 3 + 3 = 4 * 3$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

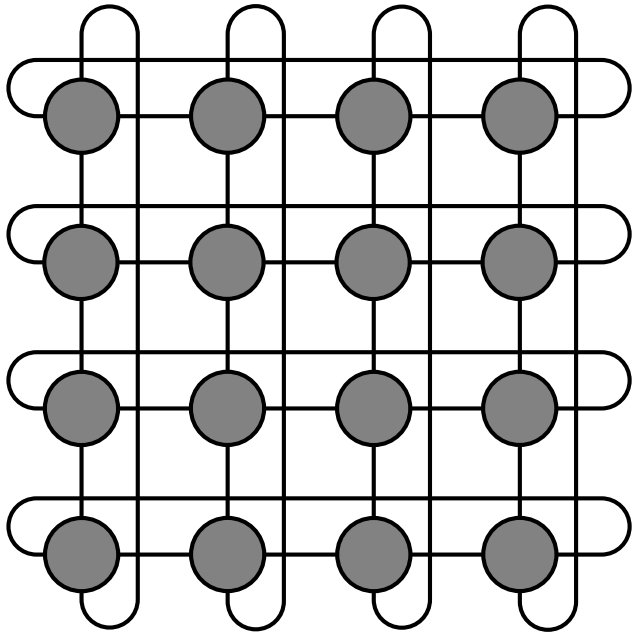


- Parallele Zeit $T(P)$ für $n^2 = P$:

$$T(P) = 4 * 3 + 1 + 1 + 1 = 15$$

Aufgabe 2.2 – Parallelisierung

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

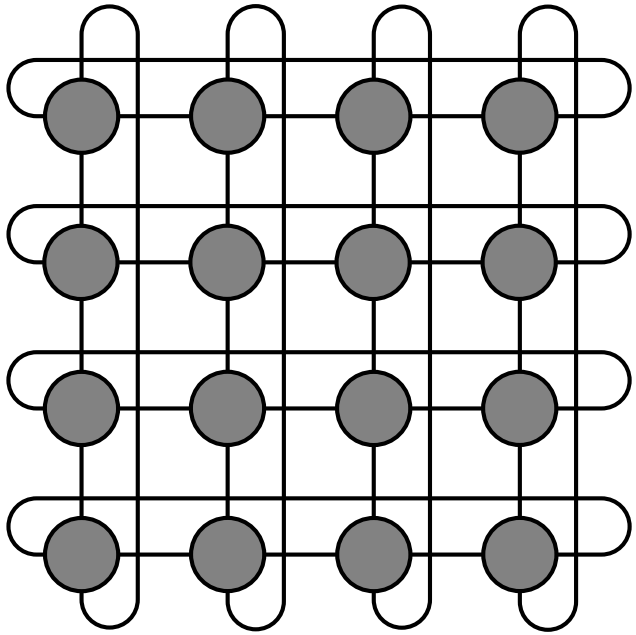


- Parallele Zeit $T(P)$ für $n^2 \neq P$, $n^2 \gg P$:

$$T(P) = (4 * 3 + 1 + 1 + 1) * \frac{n^2}{P}$$

Aufgabe 2.2 – Parallelisierung

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**



- Parallele Zeit $T(P)$ für $n^2 \neq P$, $n^2 \gg P$:

$$T(P) = \frac{4 * 3 * n^2}{P} + \frac{n^2}{P} + \frac{n^2}{P} + \frac{n^2}{P} = \frac{15n^2}{P}$$

Aufgabe 2.2 – Parallelisierung

Karlsruher Institut für Technologie

c) **Welche Beschleunigung läßt sich erzielen, wenn die Zustände der Nachbarknoten immer aus entfernten Speichern abgerufen werden müssen?**

- Sequentielle Zeit $T(1) = 7n^2$
- Parallele Zeit $T(P) = \frac{15n^2}{P}$
- Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{\frac{15n^2}{P}} = \frac{7}{15}P$$

$$\text{für } P = 2: \quad S(2) = \frac{7}{15} * 2 = \frac{14}{15} \approx 0,93$$

$$\text{für } P = 3: \quad S(3) = \frac{7}{15} * 3 = \frac{21}{15} \approx 1,40$$

$$\text{für } P = 4: \quad S(4) = \frac{7}{15} * 4 = \frac{28}{15} \approx 1,87$$

$\Rightarrow$ Die Beschleunigung skaliert mit $\frac{7}{15}$ der Prozessorenzahl (linear)

Klausuraufgaben Wintersemester 2013/14

Karlsruher Institut für Technologie

Aufgabe 5

a) **Vervollständigen Sie die Grafik auf dem Lösungsblatt, indem Sie die beim Parallelisierungsprozess durchzuführenden Schritte in die jeweils vorgesehene Box eintragen.**

⇒ siehe Folie 62!

b) Ein gegebenes paralleles Programm mit dazugehörenden Daten ist bisher auf einem Parallelrechner mit 128 Kernen ausgeführt worden. Nach dem Neukauf eines leistungsfähigeren Rechners mit 64000 Kernen zeigt die Ausführung nicht die gewünschte Beschleunigung.
Was ist für die Skalierbarkeit eines Parallelrechners besonders wichtig?

- Das Problem selbst muss skalieren (Problemgröße, Parallelisierbarkeit).

Klausuraufgaben Wintersemester 2013/14

Karlsruher Institut für Technologie

Aufgabe 5

a) **Vervollständigen Sie die Grafik auf dem Lösungsblatt, indem Sie die beim Parallelisierungsprozess durchzuführenden Schritte in die jeweils vorgesehene Box eintragen.**

   ⇒ siehe Folie 62!

b) Ein gegebenes paralleles Programm mit dazugehörenden Daten ist bisher auf einem Parallelrechner mit 128 Kernen ausgeführt worden. Nach dem Neukauf eines leistungsfähigeren Rechners mit 64000 Kernen zeigt die Ausführung nicht die gewünschte Beschleunigung.
Was ist für die Skalierbarkeit eines Parallelrechners besonders wichtig?

   - Das Problem selbst muss skalieren (Problemgröße, Parallelisierbarkeit).

Klausuraufgaben Wintersemester 2013/14

Karlsruher Institut für Technologie

Aufgabe 5

c) **Tragen Sie die folgenden Begriffe an der richtigen Stelle in die Tabelle auf den Lösungsblättern ein.**

- SMP
- DSM
- Nachrichtengekoppelter Multiprozessor
- UMA
- NUMA
- NORMA

Jede korrekte Antwort gibt einen halben Punkt, für jeden falsch eingetragenen Begriff wird ein halber Punkt abgezogen. Die Aufgabe wird nicht mit weniger als 0 Punkten bewertet.

$\Rightarrow$ siehe Folie 18!

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

Sie sollen ein geeignetes Rechensystem und eine Anwendungsimplementierung wählen, um Sequenzvergleiche möglichst schnell durchführen zu können.
Auf Architektur A sind 80% der Anwendung mit den 16 verfügbaren Kernen parallel ausführbar. Die Anwendungslaufzeit beträgt bei sequentieller Ausführung 2 Minuten.
Auf Architektur B stehen Ihnen 24 Kerne zur Verfügung, von der architekturangepassten Implementierung der Anwendung sind jedoch nur 72% parallelisierbar. Die Anwendungslaufzeit beträgt bei sequentieller Ausführung 1 Minute, 40 Sekunden.

a) **Geben Sie die Formeln für die Beschleunigung $S(n)$, die Effizienz $E(n)$ sowie den Parallelindex $I(n)$ und die Auslastung $U(n)$ an.**

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

a) **Geben Sie die Formeln für die Beschleunigung $S(n)$, die Effizienz $E(n)$ sowie den Parallelindex $I(n)$ und die Auslastung $U(n)$ an.**

- $S(n) = T(1)/T(n)$
- $E(n) = S(n)/n$
- $I(n) = P(n)/T(n)$
- $U(n) = I(n)/n = R(n) * E(n)$

b) **Nach welcher Formel können Sie die maximal erzielbare Beschleunigung eines Programms errechnen? Leiten Sie die Formel her und geben Sie dazu Namen, Formel und Bestandteile an.**

$\Rightarrow$ Amdahls Gesetz erklären, siehe Folie 27!

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

a) **Geben Sie die Formeln für die Beschleunigung $S(n)$, die Effizienz $E(n)$ sowie den Parallelindex $I(n)$ und die Auslastung $U(n)$ an.**

- $S(n) = T(1)/T(n)$
- $E(n) = S(n)/n$
- $I(n) = P(n)/T(n)$
- $U(n) = I(n)/n = R(n) * E(n)$

b) **Nach welcher Formel können Sie die maximal erzielbare Beschleunigung eines Programms errechnen? Leiten Sie die Formel her und geben Sie dazu Namen, Formel und Bestandteile an.**

$\Rightarrow$ Amdahls Gesetz erklären, siehe Folie 27!

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

c) **Welche Architektur wählen Sie aus?**

- $T_A(n) = 0,8 * 120s/16 + 0,2 * 120s = 6s + 24s = 30s$
- $T_B(n) = 0,72 * 100s/24 + 0,28 * 100s = 3s + 28s = 31s$

$\Rightarrow$ Wir wählen Architektur A aus, da sie um eine Sekunde schneller ist.

- Alternativer Rechnungsweg:
  - $S_A(n) <= 1/(0,8/16 + 0,2) = 1/0,25 = 4$
    auf zwei Minuten anzuwenden $\Rightarrow$ 30 Sekunden.
  - $S_B(n) <= 1/(0,72/24 + 0,28) = 1/0,31$
    auf 100 Sekunden anzuwenden $\Rightarrow$ 31 Sekunden.

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

d) **Wofür stehen die Abkürzungen NORMA, NUMA sowie UMA?**

- NORMA: No Remote Memory Access
- NUMA: Non-Uniform Memory Access
- UMA: Uniform Memory Access

e) **Nennen Sie jeweils ein Beispiel für ein Programmiermodell für Systeme mit gemeinsamem Speicher und verteiltem Speicher.**

- OpenMP für gemeinsamen Speicher
- MPI für verteilten Speicher

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

d) **Wofür stehen die Abkürzungen NORMA, NUMA sowie UMA?**

- NORMA: No Remote Memory Access
- NUMA: Non-Uniform Memory Access
- UMA: Uniform Memory Access

e) **Nennen Sie jeweils ein Beispiel für ein Programmiermodell für Systeme mit gemeinsamem Speicher und verteiltem Speicher.**

- OpenMP für gemeinsamen Speicher
- MPI für verteilten Speicher

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

f) **Geben Sie die Klassifizierung von Rechensystemen nach M. Flynn an.**

Vier Klassen von Rechnerarchitekturen:

- SISD: Single Instruction – Single Data
- SIMD: Single Instruction – Multiple Data
- MISD: Multiple Instruction – Single Data
- MIMD: Multiple Instruction – Multiple Data

g) **In welche Klasse bei der Klassifikation von Rechensystemen nach Flynn fallen Systeme, auf denen MPI-Programme ausgeführt werden?**

- MIMD: Multiple Instruction – Multiple Data

$\Rightarrow$ siehe Folie 14!

Klausuraufgaben Sommersemester 2013

Karlsruher Institut für Technologie

Aufgabe 5

f) **Geben Sie die Klassifizierung von Rechensystemen nach M. Flynn an.**

Vier Klassen von Rechnerarchitekturen:

- SISD: Single Instruction – Single Data
- SIMD: Single Instruction – Multiple Data
- MISD: Multiple Instruction – Single Data
- MIMD: Multiple Instruction – Multiple Data

g) **In welche Klasse bei der Klassifikation von Rechensystemen nach Flynn fallen Systeme, auf denen MPI-Programme ausgeführt werden?**

- MIMD: Multiple Instruction – Multiple Data

$\Rightarrow$ siehe Folie 14!

Klausur Rechnerstrukturen

Karlsruher Institut für Technologie

Klausur am 06.08.2014, 11:00 Uhr

- Anmeldung ist online möglich via QISPOS!
- Anmeldeschluss ist der 29.07.2014
- **Danach besteht kein Anrecht mehr auf Klausurteilnahme!**

- Abmeldung online bis 29.07.2014
- **Hinweise auf der RS-Homepage beachten!**

- Erasmus-Studenten,. . .

$\Rightarrow$ Bitte Prüfungzulassung bei mir persönlich im Büro oder nach den Übungen abgeben!

Ausblick

Übung #7 – 10.07.2014

- **Verbindungsstrukturen**
- **Vergleich von Parallelrechnern**
- Klausuraufgaben

- **Evaluation der Rechnerstrukturen-Übung**

Karlsruher Institut für Technologie

Fragen?

Karlsruher Institut für Technologie

Zentralübung Rechnerstrukturen im SS 2014

Parallelismus und Parallele Programmierung

Oliver Mattes, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

24. Juni 2014