

PARALLELES RECHNEN

Einführung

GLIEDERUNG

1. Rechnerarchitekturen
2. Leistungsbewertung
3. Parallelisierungskonzepte

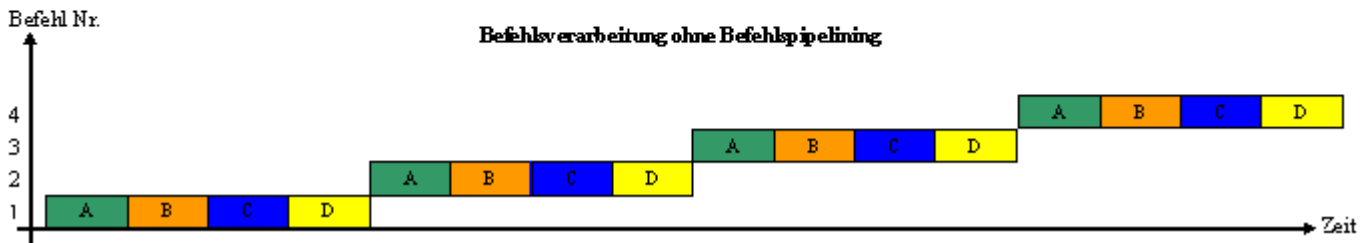
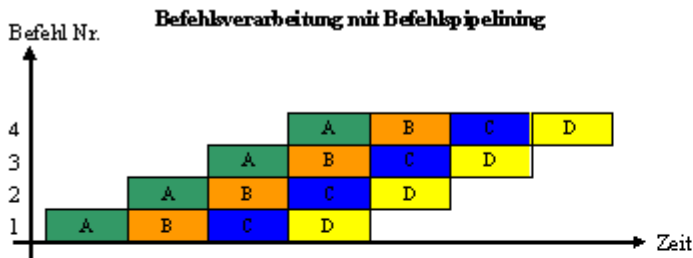
NEBENLÄUFIGKEIT

- Befehle werden parallel auf einem Prozessor ausgeführt (schnelles Umschalten durch Scheduler des OS)
 - Vorteil:
 - Befehle können quasi-parallel abgearbeitet werden
 - Nachteil:
 - Nebenkosten durch häufiges Umschalten
- Befehle werden auf verschiedenen Prozessoren ausgeführt
 - Vorteil:
 - Verarbeitungsgeschwindigkeit steigt
 - Nachteil:
 - temporärer Stillstand bei Datenabhängigkeiten

PIPELINING

- Abarbeitung eines Befehls wird in mehrere Teilaufgaben zerlegt
- Anstatt des gesamten Befehls werden pro Taktzyklus verschiedene Teilaufgaben mehrerer Befehle parallel abgearbeitet
- Teilaufgaben sind einfacher und damit schneller auszuführen

PIPELINING



- A – Befehlscode laden
- B – Instruktion dekodieren und Laden der Daten
- C – Befehl ausführen
- D – Ergebnisse zurückführen

PIPELINING

- Vorteil:
 - Taktfrequenz und Verarbeitungsgeschwindigkeit steigt
- Nachteil:
 - Pipeline Flush (Verwerfen von Ergebnissen abgearbeiteter Teilbefehle wegen Überlagerung) bei Datenabhängigkeiten

NEBENLÄUFIGKEIT VS. PIPELINING

○ Pipelining

- hardwareseitige Erweiterung von Prozessoren mit einem Kern da, um Parallelität zu unterstützen
 - Bsp.: MMX (Multi Media Extension)
- bei Prozessoren mit mehreren Kernen werden die Teilaufgaben der Befehlsabarbeitung verteilt

○ Nebenläufigkeit

- bei Prozessoren mit einem Kern erlaubt softwareseitiger Scheduler quasi-parallele Abarbeitung von Befehlen
- bei Prozessoren mit mehreren Kernen oder Mehrprozessor-Rechnern erlaubt vollständige Parallelität



VORBEMERKUNGEN

- Konzentration auf numerische Problemstellungen
- Konzentration auf Nebenläufigkeit anstatt Pipelining, da es immer stärker an Bedeutung verliert
 - Parallelität meint im folgenden Nebenläufigkeit

1. RECHNERARCHITEKTUR

- Definition
- Klassifizierung
 - SISD-Rechner
 - MISD-Rechner
 - SIMD-Rechner
 - MIMD-Rechner

DEFINITION

- „Unter dem Begriff Rechnerarchitektur versteht man: die interne Struktur des Rechners, d.h. seinen Aufbau aus verschiedenen Komponenten und die Organisation der Arbeitsabläufe.“
(Stahlknecht, Hasenkamp [2005])

KLASSIFIZIERUNG

- nach Michael J. Flynn

	Single Instruction	Multiple Instruction
Single Data	SISD	[MISD]
Multiple Data	SIMD	MIMD

- Nachteil: Keine globale Sicht auf parallele Rechnerarchitektur
 - Softwareaspekte
 - Topologie der Prozessoren und des Speichers bleiben unberücksichtigt
- Vorteil: Ausschnitt einer parallelen Rechnerarchitektur geht auf Eigenschaften ein, die für paralleles Rechnen wichtig sind

SISD-RECHNER

- traditionelle Einprozessor-Rechner
- sequentielle Abarbeitung von Instruktionen und Daten



- nach Von-Neumann- oder Harvard-Architektur aufgebaut
- Vertreter:
 - PC (Personal Computer)
 - Workstation (leistungsfähiger)

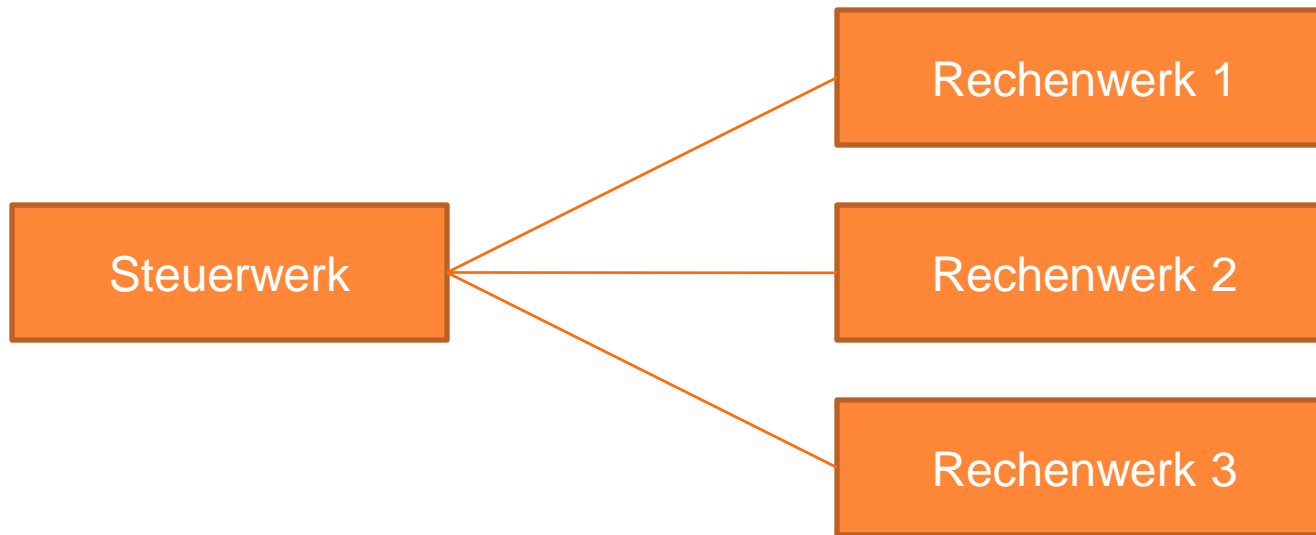


VON-NEUMANN- VS. HARVARD-ARCHITEKTUR

- Von-Neumann-Architektur verwendet für Instruktionen und Daten selben Speicher
- Harvard-Architektur verwendet für Instruktionen und Daten getrennten Speicher
 - Vorteil: Instruktionen und Daten können gleichzeitig geladen werden (Parallelität auf sehr niedrigen Ebene)

SIMD-RECHNER

- ein Befehlsstrom, mehrere Datenströme
- Befehl wird synchron auf große Menge an unterschiedlichen Daten angewendet



SIMD-RECHNER

- Rechenwerke entsprechen Prozessoren mit lokalem Speicher (für operierende Daten)
- System hat einen Befehlszeiger, der auf einen Befehl im Steuerwerk zeigt
- Prozessoren können über gemeinsamen Speicher oder Verbindungsnetzwerk Daten austauschen
- Zur Steuerung des Befehlsflusses existieren globale Befehle und Flags in jedem Prozessor

SIMD-RECHNER



- Vertreter:
 - CM-2 (Connection Machine)
 - MasPar-2
- Vorteil: kostengünstig auf Grund einfacher Rechenwerke
- Nachteil: Leistungseinbruch bei Datenabhängigkeiten, z.B. „if-then-else-Problem“
 - Teil der Prozessoren pausiert

MIMD-RECHNER

- Multiprozessorrechner (wenige bis mehrere tausende Prozessoren)
- Prozessoren voneinander unabhängig
- SPMD-Modus: Auf allen verfügbaren Prozessoren läuft gleichzeitig der identische Objektcode ab
- Zentrale Frage: Anordnung der Prozessoren (Topologie)
- größte Bedeutung heutzutage
- Vertreter:
 - Intel Core Duo
 - Intel Core Quad

TOPOLOGIE – EIGENSCHAFTEN

- Anzahl der gesamten Verbindungen
 - Einfluss auf die Komplexität der Entwicklung
- max. Anzahl der Anschlüsse / Prozessor
 - Grad der effizienten Implementierung
- max. Pfadlänge
 - Aufwand für die Kommunikation

TOPOLOGIE – BEISPIELE

- Ringanordnung
- Sternanordnung
- Binärbaumanordnung (sehr günstig)
- Kettenanordnung
- Gitteranordnungen
- vollständige Vernetzung

MIMD-RECHNER

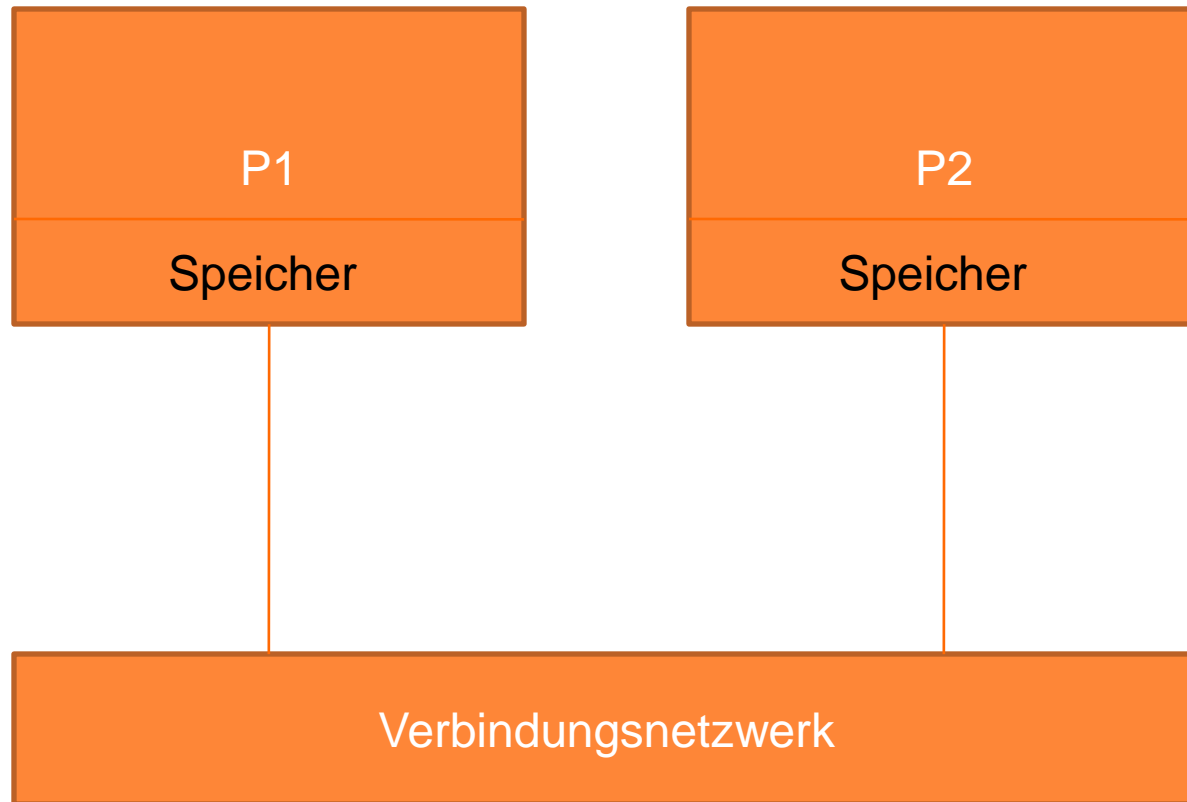
- bisher: allgemeine Eigenschaften
- jetzt: verschiedene Ansätze
 - verteilter Speicher
 - gemeinsamer Speicher
 - virtuell gemeinsamer Speicher

VERTEILTER SPEICHER

- Jeder Prozessor verfügt über lokalen Speicher inkl. assoziiertem Speicher
- Kommunikation findet über Nachrichtenaustausch statt
- Beispiel:
 - MPI Programmierstandard (message-passing Programmiermodell)
- Nachteil: Verbindungsaufbau mit anschließender Kommunikation zwischen Prozessoren ist zeitaufwendig
 - Abhilfe durch Kommunikationsprozessoren



VERTEILTER SPEICHER



GEMEINSAMER SPEICHER

- Kommunikation erfolgt über gemeinsamen Speicher
- Beispiel:
 - OpenMP Programmierstandard (shared-memory Programmiermodell)
- Nachteile:
 - schreibende Zugriffe mehrerer Prozessoren auf gleichen Speicherbereich erfordert eine Synchronisierung
 - Verzögerungen
 - synchrones Laden von Daten ins Register
 - begrenzte Übertragungsgeschwindigkeit des Bussystems

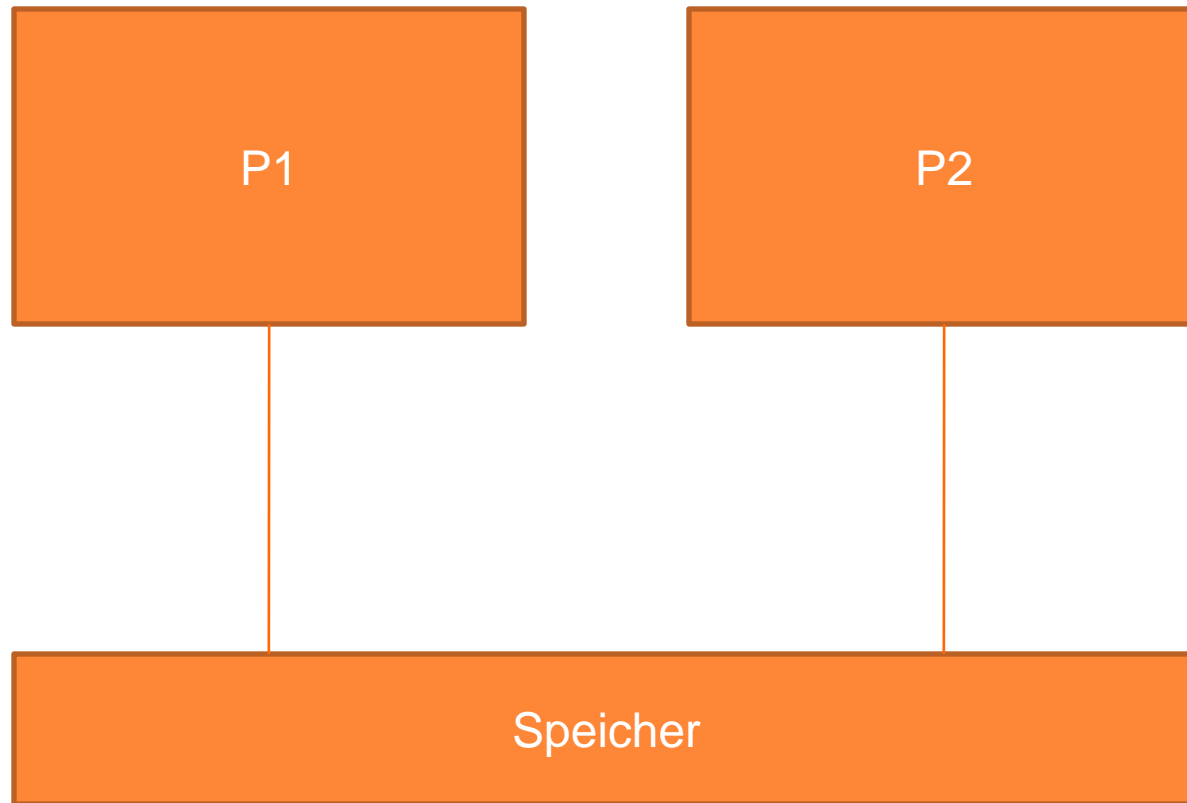


GEMEINSAMER SPEICHER

○ Vorteile:

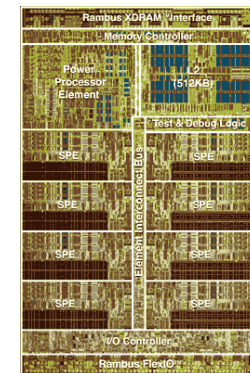
- Kommunikation bei kleiner Prozessorzahl
programmiertechnisch sehr einfach realisierbar

GEMEINSAMER SPEICHER



VIRTUELL GEMEINSAMER SPEICHER

- Versuch die Vorteile bisheriger Ansätze zu vereinigen
- Arbeitsspeicher ist physikalisch auf die einzelnen Prozessoren verteilt
- Betriebssystem verwendet diesen Speicher allerdings als gemeinsam organisierten Speicher
- Beispiel:
 - Cell-Prozessor (Ringanordnung und Element Interface Bus – Verbindungsnetzwerk)



2. LEISTUNGSBEWERTUNG

- Kriterien für parallele Systeme
 - Rechnerarchitektur – Kriterien
 - Methode und Implementierung - Kriterien

LEISTUNGSBEWERTUNG PARALLELER SYSTEME

- Paralleles System:
 - Hardwarekomponenten (Rechnerarchitektur)
 - verwendete Algorithmen (Methode)
 - verwendete Sprache, Datenstruktur (Implementierung)
- Konzentration auf numerische Problemstellungen

RECHNERARCHITEKTUR - KRITERIEN

○ Leistungsmaße:

- Mips (Million instructions per second)
- Mflops (Million floating point operations per second)

$$\text{Mips-Rate} = \frac{\text{\#Instruktionen in einem Programm}}{\text{Ausführungszeit} \cdot 10^6}$$

$$\text{Mflops-Rate} = \frac{\text{\#Gleitpunktoperationen in einem Programm}}{\text{Ausführungszeit} \cdot 10^6}$$

RECHNERARCHITEKTUR - KRITERIEN

- Nachteil: nur sehr ungenaue Bestimmung der Leistungsfähigkeit möglich
 - Parallelität der Hardware (z.B. Prozessorzahl)
 - Systemsoftware (z.B. Betriebssystem)
 - Lastparameter (z.B. Vektorlänge bei Pipelining)
 - maximaler Speicherausbau pro Prozessor und Speicherhierarchiegehen nicht in Leistungsmaße ein
- Komplexere Leistungsmaße sind erforderlich
 - CPU Benchmarks

RECHNERARCHITEKTUR – HEUTE

- ständiger Anstieg der Leistung pro Prozessor für die MIMD-Rechner
 - Trend zu Standardprozessoren (kostengünstig)
- SIMD-Rechner verlieren an Bedeutung
- Trend
 - MPP-Systeme (massiv parallele Prozessoren)
 - Clustering (Verbund von Workstations mittels Hochgeschwindigkeitsverbindung zu großem Parallelrechner)

JUGENE-RECHNER

- 2. schnellster Rechner der Welt
- im Forschungszentrum Juelich, Deutschland
- Systemfamilie: IBM BlueGene
- Systemmodell: eServer Blue Gene Solution
- LINPACK Benchmark
 - 65536 Prozessoren
 - Mflops-Rate: 222.822.000



(Quelle: <http://www.top500.org>)

METHODE UND IMPLEMENTIERUNG – KRITERIEN

- entspricht Leistungsbewertung eines parallelen numerischen Algorithmus
- paralleler numerischer Algorithmus
 - numerische Lösungsmethode
 - Implementierung auf leistungsstarker Hardware

EXKURS: ENTWICKLUNG VON PARALLELEN ALGORITHMEN

- Codierung in höhere Programmiersprache
- Zerlegung des Gesamtproblems in gleiche Teile
- Abbildung entspricht Verteilung der einzelnen Teile auf verschiedene Prozessoren

AMDAHL'S GESETZ

- Parallele Algorithmen lassen sich oftmals nicht vollständig parallelisieren
- f : parallelisierbarer Anteil $0 \leq f \leq 1$
 - $f=0$ überhaupt nicht parallelisierbar
 - $f=1$ vollständig parallelisierbar
- P : Anzahl der eingesetzten Prozessoren

max. Geschwindigkeitssteigerung = $f \cdot P$

(bezogen auf $P = 1$)

LAUFZEITBESTIMMUNG

⊙ Laufzeit Prozessor i

$$T_{ges}^i := T_{comp}^i + T_{comm}^i + T_{idle}^i + T_{start}^i + T_{plaz}^i$$

⊙ Gesamtlaufzeit

$$T_{ges} := \max_{i=1, \dots, P} (T_{ges}^i)$$

LAUFZEITBESTIMMUNG

○ Kommunikationszeit

$$T_{\text{comm}}^i := T_{\text{startup}}^i + T_{\text{sr}}^i$$

T_{comm}^i := Kommunikationszeit (hardwareabhängig)

T_{sr}^i := drückt Nachrichtenlänge und Weglänge aus

○ Idle-Zeiten entstehen durch das Warten auf Nachrichten

○ Startzeit

T_{start}^i := Zeit zum Starten des parallelen Algorithmus

(hardwareabhängig und vernachlässigbar)

LAUFZEITBESTIMMUNG

- Platzierungszeit

$T_{\text{plaz}}^i :=$ Zeit für Zerlegung und Abbildung des Gesamtproblems

FOLGEN FÜR ENTWICKLUNG EINES ALGORITHMUS

- Reduzierung der Kommunikationszeit
 - hohe Datenlokalität
 - Kommunikation am Besten nur zwischen benachbarten Prozessoren
- Aufwand für Zerlegung und Abbildung sollte \ll Rechenaufwand sein
- dynamische Lastenverteilung, da Berechnungsaufwand pro Prozessor über die Zeit variiert

SPEEDUP

- Geschwindigkeitszuwachs
 - bisher: Amdahl's Gesetz (geeignet für einfache Problemstellung)
 - jetzt: für komplexe Problemstellung
- 3 Modelle:
 - feste Problemgröße
 - feste Laufzeit
 - heterogene Umgebung

SPEEDUP - FESTE PROBLEMGRÖÖE

- vollständig parallele Algorithmen

$$S_{par}(P) := \frac{T_{ges}(1)}{T_{ges}(P)} \quad S_{par}(P) \leq P$$

- teilweise parallele Algorithmen

$$S_{par}(P) := \frac{T_{ges}(1)}{T_{ges}(P)} = \frac{T_{ges}(1)}{[(f/P) + (1-f)]T_{ges}(1)}$$

$$S_{par}(P) := \frac{P}{f + (1-f)P} \quad \text{Ware - Gesetz}$$

- erhebliche Reduzierung des Speedup erkennbar

SPEEDUP – FESTE LAUFZEIT

- vollständig paralleler Algorithmus

$$S_{\text{skal}}(P) := \frac{\text{Problemgröße auf } P \text{ Prozessoren bei fester Laufzeit } T}{\text{Problemgröße auf 1 Prozessor bei fester Laufzeit } T}$$

- teilweise paralleler Algorithmus

$$S_{\text{skal}}(P) := P + (1 - P) \cdot (1 - f)$$

(Gesetz von Gustafson)

SPEEDUP – NACHTEILE

- Speedup abhängig von Rechnerarchitekturen
 - Überinterpretation von Speedup Zahlen möglich
 - Bsp.: Ineffizienter paralleler Algorithmus kann sehr gute Speedup Werte haben im Vergleich zu einer effizienten sequentiellen Implementierung mit deutlich besserer Laufzeit

SPEEDUP - ALTERNATIVE DEFINITION

- deshalb: Normierung auf schnellsten bekannten sequentiellen Algorithmus

$$S_{\text{ges}}(P) := \frac{\text{Laufzeit des schnellsten sequ. Algorithmus}}{\text{Laufzeit des parallelen Algorithmus auf } P \text{ Prozessoren}}$$

SPEEDUP – HETEROGENE UMGEBUNG

- Umgebung mit unterschiedlichen Prozessoren
 - Bsp.: Cluster
- Lastverteilung muss unterschiedliche Leistungsfähigkeit berücksichtigen
- Laufzeitgewinn
- Normierung auf den langsamsten Rechner bzw. schnellsten Rechner

3. PARALLELISIERUNGSANSÄTZE

- Ebenen der Parallelität
- Probleme

EBENEN DER PARALLELITÄT

Prozesse	Ebene 3
Daten	Ebene 2
Anweisungen	Ebene 1
Maschinenbefehle	Ebene 0

EXKURS: IMPLIZITE VS. EXPLIZITE PARALLELITÄT

- implizite Parallelität
 - vollständige Unterstützung durch Compiler
 - automatische Platzierung von Daten auf verschiedenen Prozessoren
- explizite Parallelität
 - geringe Unterstützung durch Compiler
 - Platzierung von Daten durch Programmierer mittels entsprechender Sprachkonstrukte
- Mischformen existieren

EBENE 3

- Voraussetzung: Unterstützung vom Betriebssystem
- mehrere parallel ablaufende, unabhängige Benutzerprozesse
 - Rechner mit einem Prozessor
 - Multitasking
 - Rechner mit mehreren Prozessoren
 - virtuelles Multitasking
- steigert den Durchsatz an Benutzerprozessen
- hohes Parallelisierungspotenzial

EBENE 2

- Voraussetzung: Multiprozessorsystem
- Datenparallelität mittels Datenstrukturen
 - i.A. Vektoren und Matrizen
 - Bsp.: Lösen linearer Gleichungssysteme durch Zerlegungsverfahren
- Unterstützung durch Compiler (explizite Parallelität)
- Effizienz von Implementierung abhängig
- hohes Parallelisierungspotenzial

EBENE 1 UND 0

- Parallelität von einzelnen (Maschinen-)Instruktionen
- effiziente Unterstützung von Compilern (implizite Parallelität)
- niedriges Parallelisierungspotenzial

PROBLEME

- bezogen auf Entwicklung und Implementierung paralleler Algorithmen
 - Synchronisierung
 - Verklemmung

SYNCHRONISIERUNG

- Synchronisierung in Form von Kommunikation
 - Signalisierung der Beendigung der eigenen Berechnung
 - Austausch notwendiger Daten untereinander
- Problem: intensive Kommunikation erhöht Gesamtlaufzeit und erniedrigt parallele Effizienz

VERKLEMMUNG

- Verklemmung eng verbunden mit Synchronisation
 - mehrere parallele Prozesse warten auf Daten
 - temporärer Stillstand
- Problem: Verklemmung erhöht Gesamtlaufzeit und erniedrigt parallele Effizienz

BIBLIOGRAPHIE

- W. Huber: Paralleles Rechnen. Eine Einführung. Oldenbourg, München 2002
- Artikel Flynnsche Klassifikation. In: Wikipedia, Die freie Ezyklopädie. Bearbeitungsstand: 30. April 2008, 11:23 UTC.
 - URL:
http://de.wikipedia.org/w/index.php?title=Flynnische_Klassifikation&oldid=45480120

BIBLIOGRAPHIE

- Artikel Nebenläufigkeit. In: Wikipedia, Die freie Ezyklopädie. Bearbeitungsstand: 24. April 2008, 16:26 UTC.
 - URL:
<http://de.wikipedia.org/w/index.php?title=Nebenl%C3%A4ufigkeit&oldid=45265059>
- Artikel Pipeline (Prozessor). In: Wikipedia, Die freie Ezyklopädie. Bearbeitungsstand: 12. März 2008, 23:44 UTC.
 - URL:
http://de.wikipedia.org/w/index.php?title=Pipeline_%28Prozessor%29&oldid=43635189

BIBLIOGRAPHIE

- Artikel Intel Pentium. In: Wikipedia, Die freie Ezyklopädie. Bearbeitungsstand: 29. April 2008, 14:14 UTC.
 - URL:
http://de.wikipedia.org/w/index.php?title=Intel_Pentium&oldid=45447772
- Artikel Cell (Prozessor). In: Wikipedia, Die freie Ezyklopädie. Bearbeitungsstand: 23. April 2008, 11:15 UTC.
 - URL:
http://de.wikipedia.org/w/index.php?title=Cell_%28Prozessor%29&oldid=45212818

ENDE

Vielen Dank für Ihre Aufmerksamkeit