



Grundlagen digitaler Systeme

Kapitel 15: Speicherhierarchie

Holger Karl

Was bisher geschah

- Speicher sind im Vergleich zu CPUs sehr langsam
- Schnelle Speicher sehr teuer, also klein
- Caches haben sich als unverzichtbar für akzeptable Leistung eines Rechners erwiesen

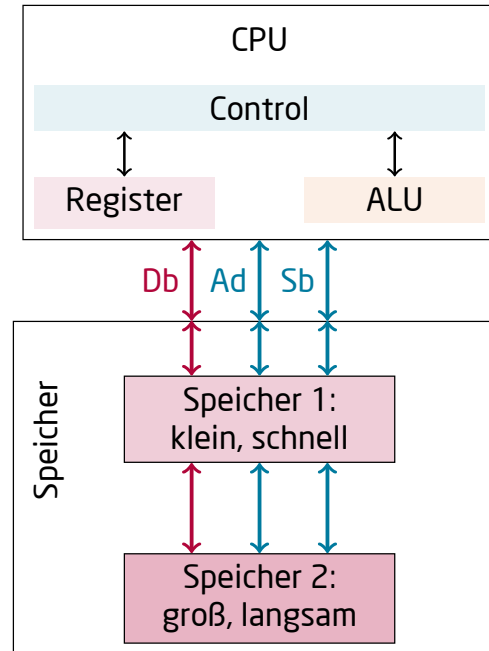


Abbildung 15.1: Systemstruktur aus CPU, Cache und Speicher
(Wiederholung von Abb. 14.9)

Hierarchie
Parallelität
Cache-Kohärenz
Zusammenfassung
Material

Plan für dieses Kapitel

5.1-5.4

- Verallgemeinerung von einem Cache zu mehreren Cache-Stufen
- Weitere Verallgemeinerung zu Hierarchie unterschiedlicher Speicher
- Optionen für Parallelverarbeitung und Zusammenhang mit Speicherhierarchie, insbes. Cache-Kohärenz

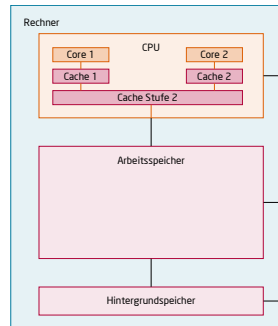


Abbildung 15.2: Einordnung Kapitel 15 in Gesamtbild

Teil 4: Speicher

Cache

Speicherhierarchie

Virtueller Speicher

I/O, stabiler Speicher

Abbildung 15.3: Speicherhierarchie

Hierarchie
Parallelität
Cache-Kohärenz
Zusammenfassung
Material

- Nutzen mehrstufiger Speicherhierarchien begründen, quantitativ einschätzen
- Programmausführung in Multi-Core-Systemen erläutern
- Cache-Kohärenz-Probleme erkennen und Strategien zur Beseitigung begründet auswählen

Hierarchie
Parallelität
Cache-Kohärenz
Zusammenfassung
Material

Inhaltsverzeichnis

1. Hierarchie

1.1 Mehrere Caches

1.2 Vier Aspekte

1.3 Arten von Cache Miss

2. Parallelität

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15: Speicherhierarchie

H. Karl, WS 22/23

Folie 5/67

Erinnerung: Mehrere Teil-Speicher

■ Ideen in Kapitel 14

- Mehrere Teil-Speicher nebeneinander
- Set-assoziativer Cache als mehrere parallel genutzte Teil-Caches

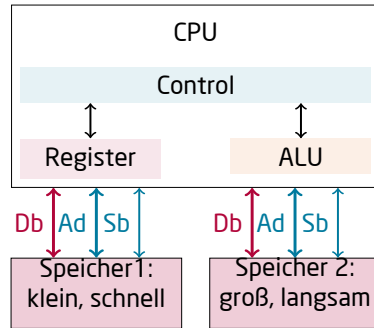


Abbildung 15.4: Mögliche Struktur für separierte heterogene Speicher (Db: Datenbus, Ad: Adressbus, Sb: Steuerbus) - Wiederholung von Abb. 14.8

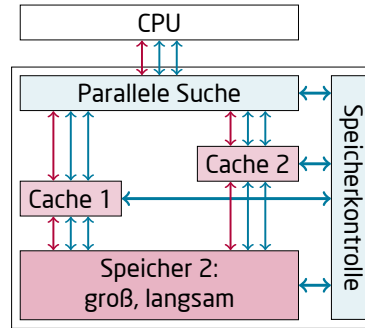


Abbildung 15.5: Aufteilung eines Caches in zwei Teil-Caches, die parallel durchsucht und benutzt werden - Wiederholung von Abb. 14.21

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15: Speicherhierarchie

H. Karl, WS 22/23

Folie 6/67

Register vs. Cache vs. Original-Speicher

- Bis jetzt Reihenfolge:
 - ☐ Original-Speicher/Arbeitsspeicher: groß, langsam
 - ☐ Cache
 - ☐ Register: schnell, wenig
- Cache überbrückt den Abstand in Datenrate/Latenz zwischen Register und Original-Speicher
 - ☐ Kleiner wegen Preisunterschied

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 7/67

Register vs. Cache vs. Original-Speicher

- Bis jetzt Reihenfolge:
 - ☐ Original-Speicher/Arbeitsspeicher: groß, langsam
 - ☐ Cache
 - ☐ Register: schnell, wenig
- Cache überbrückt den Abstand in Datenrate/Latenz zwischen Register und Original-Speicher
 - ☐ Kleiner wegen Preisunterschied
- Verallgemeinerung?
- Was, wenn mehr Speicher-Typen mit unterschiedlichen Tradeoffs Größe-Rate/Latenz-Preis zur Verfügung stünden?

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 7/67

Zwei Cache-Ebenen

- Wir fügen **zwischen Cache und Original-Speicher** einen weiteren Cache hinzu
- Terminologie: Cache Ebene (**Level**) 1, 2:
Kleinere Nummer näher an CPU; kurz: L1, L2
 - Darstellung von oben nach unten
- Je höhere Ebene, desto kleiner, schneller, teurer

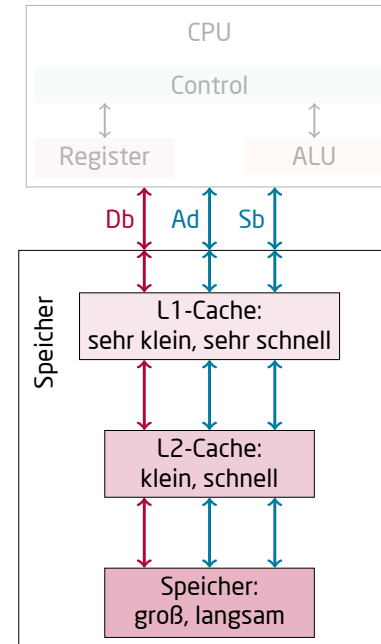


Abbildung 15.6: Systemstruktur aus CPU, Cache mit zwei Ebenen und Speicher

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15: Speicherhierarchie

H. Karl, WS 22/23

Folie 8/67

Verallgemeinerung: Mehrere Cache-Ebenen

- Das geht natürlich auch mit drei, vier, . . . Ebenen

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 9/67

Betrieb mehrerer Cache-Ebenen: Transparenz

- Caches sind **transparent** vor einander
- Cache der Ebene k sieht Cache der Ebene $k + 1$ als Original-Speicher
- Cache der Ebene k sieht Cache der Ebene $k - 1$ als CPU

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 10/67

- Jede Cache-Ebene kann beliebig organisiert werden
 - Z.B. als Write-through oder Write-back Cache
 - Z.B. als direkter, set-assoziativer oder voll assoziativer Cache
 - Z.B. integrierter oder separater Cache für Instruktionen und Daten

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 11/67

- Jede Cache-Ebene kann beliebig organisiert werden
 - Z.B. als Write-through oder Write-back Cache
 - Z.B. als direkter, set-assoziativer oder voll assoziativer Cache
 - Z.B. integrierter oder separater Cache für Instruktionen und Daten
- Optimale Organisation?
 - Großes Optimierungsproblem; viele Kombinationen
 - Und wie üblich: Hängt von Programm-Mix ab

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 11/67

1. Hierarchie

1.1 Mehrere Caches

1.2 Vier Aspekte

1.3 Arten von Cache Miss

2. Parallelität

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Viele Optionen - Gemeinsamkeiten?

- Komplexe, heterogene Speicherhierarchie kann schnell unübersichtlich werden
- Gibt es grundlegende Optionen, um das zu ordnen?

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss


Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Viele Optionen - Gemeinsamkeiten?

 5.8

- Komplexe, heterogene Speicherhierarchie kann schnell unübersichtlich werden
- Gibt es grundlegende Optionen, um das zu ordnen?
- Ja! - Speicherhierarchie hat vier wesentliche Entwurfsaspekte

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 13/67

Vier Aspekte

1. Wo Block ablegen?
2. Wie Block finden?
3. Welchen Block verdrängen?
4. Wie schreiben?

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 14/67

Aspekt 1: Wo kann ein Block abgelegt werden?

- Für gegebenen Block: Unter wie viele, welchen Orten kann ausgewählt werden?

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 15/67

Aspekt 1: Wo kann ein Block abgelegt werden?

- Für gegebenen Block: Unter wie viele, welchen Orten kann ausgewählt werden?
- Beispiele:
 - ☐ Direkter Cache: keine Alternativen; genau ein Ort für Block
 - ☐ Voll-assoziativer Cache: alle Orte im Cache sind Kandidaten
 - ☐ Teil-assoziativer Cache: Grad der Assoziativität

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Aspekt 2: Wie wird Block im Cache gefunden?

- Für gegebene Adresse: Wie wird der zugehörige Block im Cache gefunden?

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Aspekt 2: Wie wird Block im Cache gefunden?

- Für gegebene Adresse: Wie wird der zugehörige Block im Cache gefunden?
- Beispiele:
 - ☐ Direkter Cache: Blocknummer aus Adresse berechnet
 - ☐ Voll-assoziativer Cache: Suche nötig
 - Linear durch alle Kandidaten
 - Oder mit Suchstrukturen (z.B. Hash-Tabelle) unterstützt
 - ☐ Teil-assoziativ: Anzahl Suchschritte = Grad der Assoziativität

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 16/67

Aspekt 3: Welcher Block wird verdrängt?

- Strategie bei Verdrängen eines Blocks?
 - ☐ Unter den Optionen, die Aspekt 1 offen lässt
- Beispiele (vgl. Abschnitt 14.4):
 - ☐ Zufällige Auswahl
 - ☐ LFU, LRU
 - ☐ Approximationen davon (insbes. bei hoher Assoziativität nötig)

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Aspekt 4: Wie werden Schreibzugriffe behandelt?

- Was passiert beim Schreiben in eine Cache-Ebene?
- Beispiele (vgl. 14.2.3):
 - Write-through
 - Write-back

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

1. Hierarchie

1.1 Mehrere Caches

1.2 Vier Aspekte

1.3 Arten von Cache Miss

2. Parallelität

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 19/67

Wie kommt es zu Cache Miss?

- Zur weiteren Einordnung: Wie kommt es zu Cache Miss; welche Sorten gibt es?
- Hier: Das 3C-Modell

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Compulsory Miss: Unvermeidbar

- Wenn auf einen Block erstmals zugegriffen wird **kann** er nicht im Cache liegen
- Der Cache Miss ist unvermeidlich
 - Egal wie man Cache parametriert, organisiert, . . .
- Ein sog **compulsory miss**
 - Auch: **cold-start miss**

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15: Speicherhierarchie

H. Karl, WS 22/23

Folie 21/67

Capacity Miss: Zu kleiner Cache

- Auf einen Block wird zugegriffen; Block ist **nicht mehr** im Cache und der Cache ist voll
 - Im Unterschied zum cold-start miss: Block **war schon** im Cache, wurde aber verdrängt
- Miss durch zu kleinen Cache verursacht
 - Selbst bei einer voll-assoziativen Struktur

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15: Speicherhierarchie

H. Karl, WS 22/23

Folie 22/67

Conflict Miss: Vorhandener Platz nicht nutzbar

- Auf einen Block wird zugegriffen; Block ist nicht mehr im Cache, **obwohl noch Platz im Cache ist**
 - Passiert nur bei nicht voll-assoziativen Caches
- Ist die Erhöhung der Miss Rate bedingt durch direkte oder teil-assoziative Cache-Struktur

Hierarchie

Mehrere Caches

Vier Aspekte

Arten von Cache Miss

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 23/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

2.1 Multiprozessor

2.2 MIMD & Multi-Core

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 24/67

- Bis jetzt: Ein Prozessor arbeitet ein Programm ab; verarbeitet ein Datum nach dem nächsten
 - Sog. Single Instruction, Single Data (SISD)-Struktur
- Wie kann man Leistung steigern?
 - Leistung: Durchsatz oder Programmabarbeitungszeit!
- Optionen bisher:
 - Verbesserte Technologie (Taktfrequenz, . . .)
 - Verbesserte CPU-Architektur: Parallelität auf Instruktionsebene
 - Pipelining
 - Superskalare Ausführung
 - Verbesserte Speicheranbindung: Cache

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 25/67

Weitere Leistungssteigerung?

- Bisherige Optionen sind weitgehend ausgereizt
 - Physikalische Grenzen, Programmstrukturen, . . .
- Wir brauchen mehr Quellen der Parallelität!
 - Auf gröberer Granularität!
 - Innerhalb eines Programms
 - Zwischen mehreren Programmen
- Bei grob-granularer Parallelität: Mehrere Prozessoren einsetzen!
 - Kann Durchsatz wie Abarbeitungszeit steigern

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 26/67

Idee: Mit einer Instruktion mehrere Daten verarbeiten

- Oft: die gleiche Instruktion soll auf unterschiedliche Daten angewandt werden
 - Sog. Single Instruction, Multiple Data (SIMD)
 - Typisches Beispiel: Vektorarithmetik, z.B. in Bild-/Audioverarbeitung
- Unterstützung in CPU durch Integration in Pipeline, ALU-Ansteuerung, Datentransfer, ...
- Erfordert geeignete Code-Erzeugung
- Nicht universell anwendbar!

Hierarchie

Parallelität

Multiprozessor
MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 27/67

Idee: Gleiche Daten mit unterschiedlichen Instruktionen verarbeiten

- Sog. Multiple Instruction, Single Data (MISD)
- Tja . . .
- Dafür gibt es keine sinnvollen Beispiele

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 28/67

Idee: Unterschiedliche Instruktionen auf unterschiedliche Instruktionen

- Sog. Multiple Instruction, Multiple Data (MIMD)
- Unterschiedliche Instruktionen woher?
 - Aus unabhängigen Programmen: Sinnvoll für Durchsatz (viele Programme pro Zeiteinheit)
 - Innerhalb eines Programms: Nebenläufig ablaufbare Programmteile

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 29/67

Nebenläufig vs. Parallelität

Definition 15.1 (Nebenläufig)

Mehrere Programme oder mehrere Teile eines Programms heißen **nebenläufig** wenn sie ohne Beeinträchtigung der Korrektheit gleichzeitig oder in beliebiger Reihenfolge ausgeführt werden können.

Nebenläufigkeit ist eine Eigenschaft des/der Programme und hat nichts mit dem ausführenden System zu tun.

Definition 15.2 (Parallel)

Nebenläufige Programme werden **parallel** ausgeführt, wenn die nebenläufige Teile gleichzeitig durch mehrere Ausführungseinheiten (z.B. CPUs) ausgeführt werden. Parallelität ist eine Eigenschaft des ausführenden Systems und setzt nebenläufige Programme voraus.

Hierarchie

Parallelität

Multiprozessor
MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 30/67

Flynn'sche Taxonomy: Überblick

■ SISD, . . . : Taxonomy nach Flynn [Fly66]

Tabelle 15.1: Klassifikation nach Flynn

		Instruktionen	
		Single	Multiple
Daten	Single	SISD	(MISD)
	Multiple	SIMD	MIMD

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 31/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

2.1 Multiprozessor

2.2 MIMD & Multi-Core

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 32/67

Cluster

- Option 1: Mehrere unabhängige Rechner
 - Prima für unabhängige Programme; Durchsatzrechnen
 - Kein gemeinsames Nutzen irgendwelcher Ressourcen
 - Ggf. Datenaustausch über ein Verbindungsnetz;
– Nachrichtengekoppeltes System

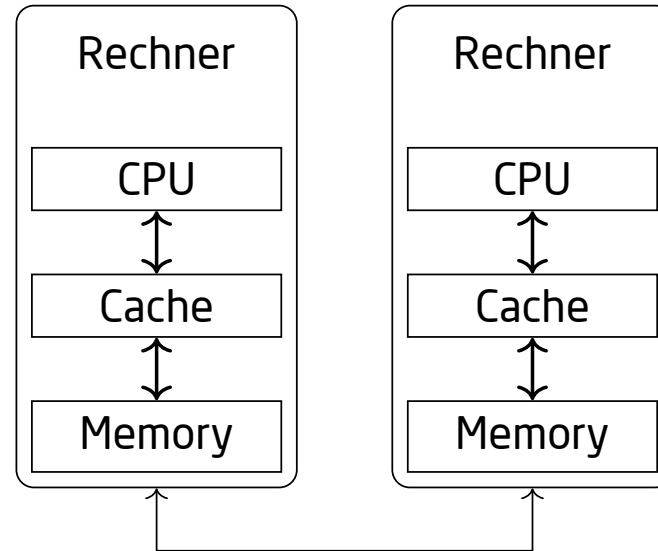


Abbildung 15.7: Unabhängige Rechner, durch Verbindungsnetz zum Datenaustausch befähigt: Ein Cluster

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 33/67

MIMD-Option: Unabhängige Prozessoren

Multiprozessor

- Option 2: Mehrere unabhängige Prozessoren, mit Cache
 - CPUs können über gemeinsamen Speicher Daten teilen; gemeinsam an Problem arbeiten
 - **Speichergekoppeltes System**
 - Müssen Speicher-Datenrate teilen
 - Separate Caches können Vor-/Nachteil sein
 - Z.B. beide CPUs greifen auf gleiche Adresse zu: Kopieren zwischen Caches notwendig

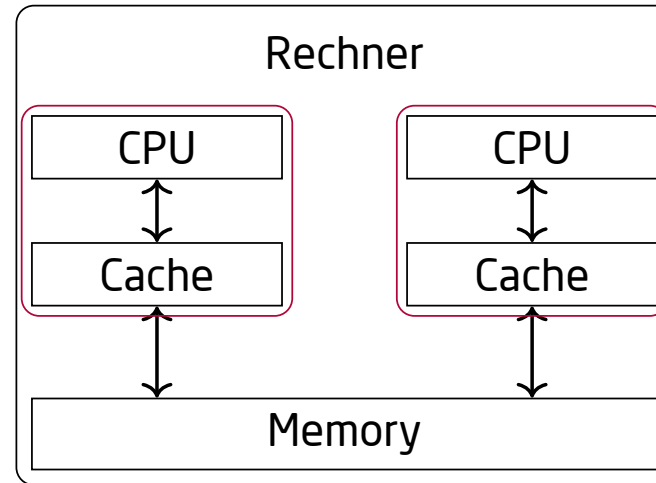


Abbildung 15.8: Unabhängige CPUs, durch Zugriff auf gemeinsamen Speicher zum Datenaustausch befähigt: Ein Multiprozessor (rote Umrandung zeigt Prozessor)

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 34/67

MIMD-Option: Unabhängige Kerne

Multi-Core Prozessor

- Option 3: Mehrere unabhängige CPU-Kernfunktionen, mit **gemeinsamen** Cache
 - Kern: ALU, Steuerwerk, Register
 - Superskalarer Prozessor: nur 1 Steuerwerk!
 - CPU Cores nutzen gemeinsamen Cache; sind **eng gekoppelt**
 - Gemeinsamer Cache ist charakteristisch!
 - Daher als **eine** CPU aufgefasst, mit mehreren unabhängig voneinander arbeitsfähigen Kernen

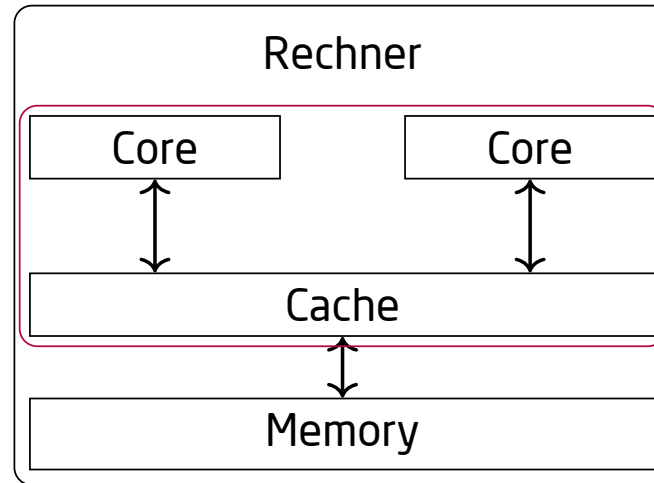


Abbildung 15.9: Unabhängige CPUs, durch gemeinsamen Cache zum Datenaustausch befähigt: Ein Multi-Core-System (rote Umrandung zeigt Prozessor)

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 35/67

Terminologie: Bauform?

- Was ist nun eigentlich ein **Prozessor**? (rote Umrandung in vorherigen Abbildungen?)

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 36/67

Terminologie: Bauform?

- Was ist nun eigentlich ein **Prozessor**? (rote Umrandung in vorherigen Abbildungen?)
- Unterschied zwischen Multiprozessor und Multi-Core kann auch über Bauform / Packaging definiert werden
 - Multiprozessor: Pro Gehäuse/Package genau ein Kern
 - Multi-Core: Pro Gehäuse/Package mehrere Kerne
- Die Zuordnung, Anzahl von Caches zu Kernen ist dazu orthogonal
- Beide Terminologien sind berechtigt; leider nicht präzise unterschieden

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 36/67

Terminologie: Bauform?

- Was ist nun eigentlich ein **Prozessor**? (rote Umrandung in vorherigen Abbildungen?)
- Unterschied zwischen Multiprozessor und Multi-Core kann auch über Bauform / Packaging definiert werden
 - Multiprozessor: Pro Gehäuse/Package genau ein Kern
 - Multi-Core: Pro Gehäuse/Package mehrere Kerne
- Die Zuordnung, Anzahl von Caches zu Kernen ist dazu orthogonal
- Beide Terminologien sind berechtigt; leider nicht präzise unterschieden
- In Praxis: weniger Relevanz, Multiprozessor-Systeme nur noch relevant in Servern, Spezialanwendungen
 - Und Cache sowieso in Package wegen Latenz, Rate

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 36/67

- Beliebige Mischformen zwischen Multiprozessor, Multi-Core, Cache-Platzierung denkbar (und existent)
- Abb. 15.10 zeigt typische Mischform
 - Naheliegend zu Multiprozessor-System aus Multi-Core-Prozessoren zu erweitern
 - Und natürlich zu Cluster

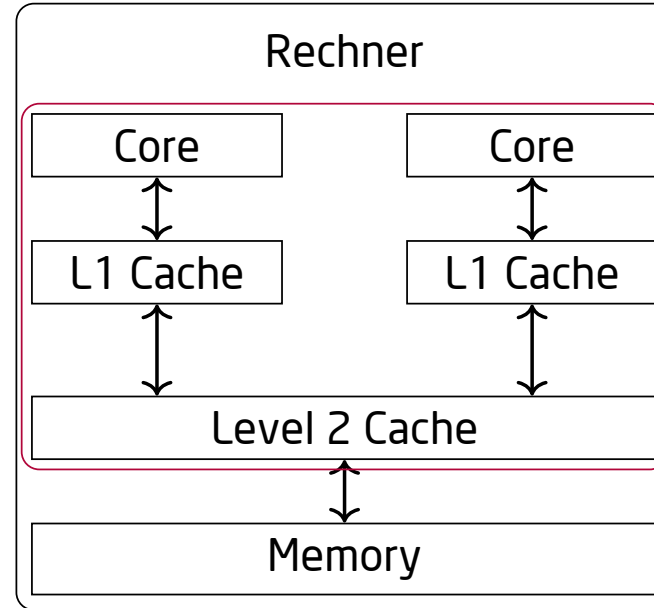


Abbildung 15.10: Mischform: Dual-Core mit zwischengeschalteten, separierten L1 Caches (rote Umrandung zeigt Prozessor)

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 37/67

Mischform: Multiprozessorsystem aus Multi-Core-Prozessoren

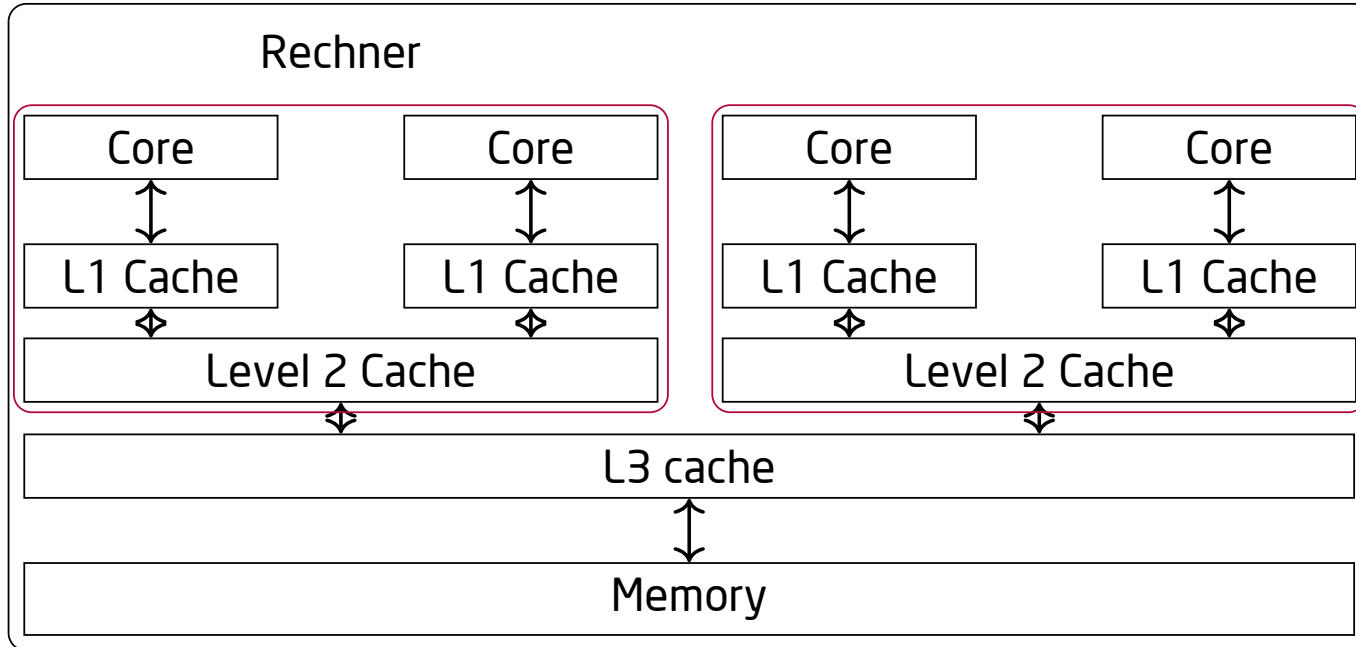


Abbildung 15.11: Mischform: Zwei-Prozessor-System mit jeweils zwei Kernen pro Prozessor; Cache der Stufe 3 wird gemeinsam von allen Prozessoren benutzt (rote Umrandung zeigt Prozessor)

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 38/67

Cache-Aufteilung: Harvard oder von-Neumann?

- Im Prinzip kann auf jeder Cache-Ebene entschieden werden, ob Instruktionen und Daten gemeinsam oder separat gespeichert werden
 - Unterscheidung: Adresse stammt aus PC vs. sonstigem Register/Wert
- Nachteil separate Caches (**split caches**): Schlechtere Hit-Rate
- Vorteil: Separate Caches können separat an CPU angebunden werden
 - Mit zusätzlichen Daten-/Adressbussen; höhere **Rate**
 - Instruktionsregister und Datenregister können **gleichzeitig** befüllt werden
- Split L1 Cache passt i.d.R. besser zu Pipeline/superskalaren CPU-Architekturen
 - L2 Cache und höher i.d.R. gemeinsam

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 39/67

AMD Zen+ Architektur (z.B. Ryzen),

2018 Quelle

- L1:
 - Instruction: 64KiB pro Core; 4-fach set associative
 - Data: 32 KiB pro Core; 8-fach set associative
- L2: 512 KiB pro Core; 8-fach set associative
- L3: 2 MiB pro Core; 16-fach set associative

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 40/67

Praxisbeispiele

AMD Zen+ Architektur (z.B. Ryzen), 2018 **Quelle**

- L1:
 - Instruction: 64KiB pro Core; 4-fach set associative
 - Data: 32 KiB pro Core; 8-fach set associative
- L2: 512 KiB pro Core; 8-fach set associative
- L3: 2 MiB **pro Core**; 16-fach set associative

Alle 64 Bytes Blockgröße; Write back

AMD Zen3 Architektur (z.B. Ryzen), 2020 **Quelle**

- L1:
 - Instruction: 32KiB pro Core; 4-fach set associative
 - Data: 32 KiB pro Core; 8-fach set associative; typisch 4-8 Zyklen Latenz
- L2: 512 KiB pro Core; 8-fach set associative; typisch > 12 Zyklen Latenz
- L3: 16-32 MiB **pro Core-Gruppe** (typisch 4 Cores); 16-fach set ass.; mittlere Latenz 46 Zyklen

Hierarchie

Parallelität

Multiprozessor

MIMD & Multi-Core

Cache-Kohärenz

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 40/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

3.1 Problem

3.2 Strategien

3.3 Mechanismen

3.4 Weitere Beispiele

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 41/67

Datum in mehreren Caches

5.10; vgl. Folie 18

- Schauen wir uns noch einmal Abb. 15.12 an, diesmal mit Inhalt der Adresse #4711 in Speicher beiden Caches
- Gleiches Datum existiert damit an **drei Orten**
 - Alle haben gleichen Wert; alles gut

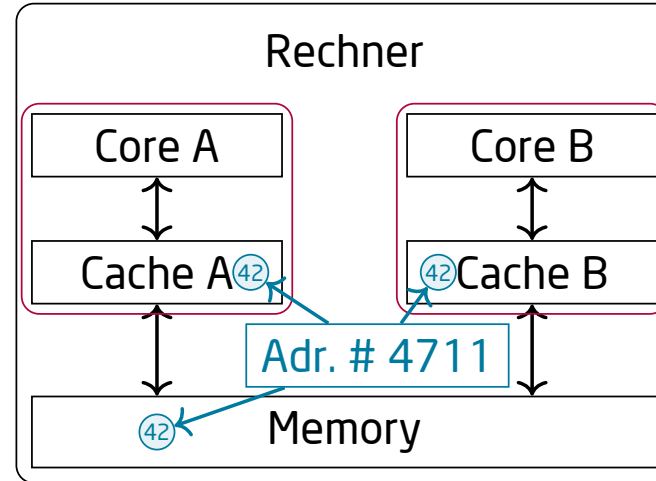


Abbildung 15.12: Inhalt der Adresse 4711 an drei Orten im System gespeichert, mit identischen Werten

Hierarchie
Parallelität
Cache-Kohärenz
Problem
Strategien
Mechanismen
Weitere Beispiele
Zusammenfassung
Material

Datum in mehreren Caches

Was passiert nach Schreib-Instruktion von Core A auf Adresse 4711?

Write-back

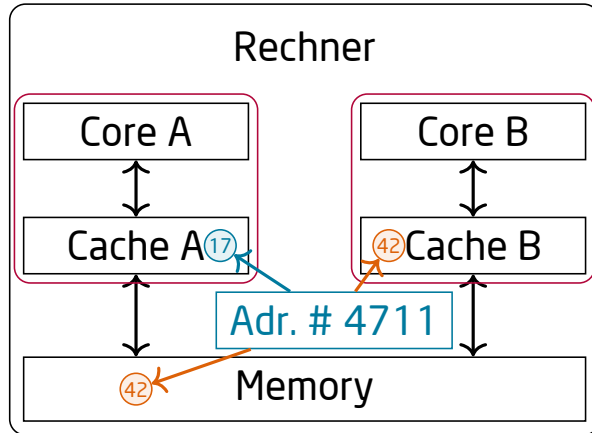


Abbildung 15.13: Unterschiedliche Werte für Adresse 4711; Inhalt von Speicher und Cache B inkonsistent zu Rest des Systems bei Write-back Cache

Write-through

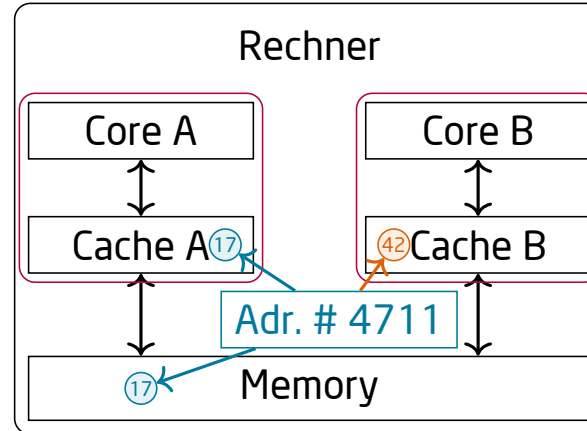


Abbildung 15.14: Unterschiedliche Werte für Adresse 4711; Inhalt von Cache B inkonsistent zu Rest des Systems bei Write-through Cache

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 43/67

Inkonsistenter Speicher

Speicher nicht mehr konsistent

Unterschiedliche Kopien des Inhalts von Adr. 4711 unterscheiden sich!

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 44/67

Inkonsistenter Speicher

Speicher nicht mehr konsistent

Unterschiedliche Kopien des Inhalts von Adr. 4711 unterscheiden sich!

Verletzung des Kohärenz-Modells?

- Ist Inkonsistenz ein Problem?
 - Vgl. Def. 14.5, [AG96]
- Verletzung hängt ab von weiterem Programmverlauf
 - Wenn Core B Adresse 4711 nicht mehr benötigt: Kein Problem
 - Wenn Core B Adresse 4711 unmittelbar liest und den **veralteten Wert** 42 erhält: Erwartungshaltung / Kohärenzmodell verletzt

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 44/67

Inkonsistenter Speicher

Speicher nicht mehr konsistent

Unterschiedliche Kopien des Inhalts von Adr. 4711 unterscheiden sich!

Verletzung des Kohärenz-Modells?

- Ist Inkonsistenz ein Problem?
 - Vgl. Def. 14.5, [AG96]
- Verletzung hängt ab von weiterem Programmverlauf
 - Wenn Core B Adresse 4711 nicht mehr benötigt: Kein Problem
 - Wenn Core B Adresse 4711 unmittelbar liest und den **veralteten Wert** 42 erhält: Erwartungshaltung / Kohärenzmodell verletzt
- Wir wollen System, das für **alle** Programme funktioniert
 - Aber mit **welchem** Kohärenzmodell? Genaue Forderung?

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 44/67

Modell: Sequentielle Kohärenz

1. Read your writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor A liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl auf X dazwischen geschah (von A oder anderem Prozessor), dann ist w das Resultat der Leseoperation

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 45/67

Modell: Sequentielle Kohärenz

1. Read your writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor A liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl auf X dazwischen geschah (von A oder anderem Prozessor), dann ist w das Resultat der Leseoperation

1. Read other writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor B liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl dazwischen geschah **und** genügend viel Zeit zwischen Lesen und Schreiben verging, dann ist w das Resultat

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 45/67

Modell: Sequentielle Kohärenz

1. Read your writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor A liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl auf X dazwischen geschah (von A oder anderem Prozessor), dann ist w das Resultat der Leseoperation

1. Read other writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor B liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl dazwischen geschah **und** genügend viel Zeit zwischen Lesen und Schreiben verging, dann ist w das Resultat

1. Write serialization:

- a) Prozessor A schreibt Wert v auf Adresse X
- b) Prozessor A oder B schreibt Wert **danach** w auf Adresse X
- c) Kein Prozessor wird mit zwei Lesebefehlen zuerst w, dann v erhalten

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 45/67

Modell: Sequentielle Kohärenz

1. Read your writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor A liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl auf X dazwischen geschah (von A oder anderem Prozessor), dann ist w das Resultat der Leseoperation

1. Read other writes:

- a) Prozessor A schreibt Wert w auf Adresse X
- b) Prozessor B liest **danach** von Adresse X
- c) **Wenn** kein anderer Schreibbefehl dazwischen geschah **und** genügend viel Zeit zwischen Lesen und Schreiben verging, dann ist w das Resultat

1. Write serialization:

- a) Prozessor A schreibt Wert v auf Adresse X
- b) Prozessor A oder B schreibt Wert **danach** w auf Adresse X
- c) Kein Prozessor wird mit zwei Lesebefehlen zuerst w, dann v erhalten

Hier: Prozessor, Core egal

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 45/67

Anmerkung: Andere Kohärenzmodelle

- Es gibt viele weitere Kohärenzmodelle
- Meist: Abschwächungen; geringere Erwartungshaltungen des Programmierers an *intuitives* Verhalten des Speichers
- Insbesondere wichtig in Systemen mit langen, heterogenen Latenzen zu unterschiedlichen Adressen

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 46/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

3.1 Problem

3.2 Strategien

3.3 Mechanismen

3.4 Weitere Beispiele

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 47/67

Ursache: Nach oben verzweigende Speicherhierarchie

- Mehrfache Kopien entstehen, weil die Speicherhierarchie **sich nach oben, zu den Cores hin, verzweigt**
 - Mehrere Caches, die die gleiche Adresse beinhalten können
- Kann entsprechend auch bei mehreren Cache-Ebenen, anderen Formen von Speicherhierarchie auftreten

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 48/67

Optionen für Strategien?

Welche Optionen haben wir, um (z.B.) das sequentielle Kohärenzmodell umzusetzen?

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 49/67

Optionen für Strategien?

Welche Optionen haben wir, um (z.B.) das sequentielle Kohärenzmodell umzusetzen?

- **Migration:** Keine Kopien in parallelen Speichern
 - ☐ Keine (horizontale) Kopie, kein Problem
 - ☐ "Vertikale" Kopien sind erlaubt
 - ☐ Bei Zugriff aus parallelem Zweig: Wert wird migriert

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 49/67

Optionen für Strategien?

Welche Optionen haben wir, um (z.B.) das sequentielle Kohärenzmodell umzusetzen?

■ **Migration:** Keine Kopien in parallelen Speichern

- ☐ Keine (horizontale) Kopie, kein Problem
- ☐ "Vertikale" Kopien sind erlaubt
- ☐ Bei Zugriff aus parallelem Zweig: Wert wird migriert

■ **Replikation:** Kopie, wie oben besprochen

- ☐ Was passiert bei Schreibzugriff auf eine Kopie?
 - **Write invalidate:** Andere Kopien werden als ungültig erklärt (aus Cache entfernt)
 - **Write update:** Andere Kopien werden auf neuen Wert aktualisiert

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 49/67

Strategie: Forderung erfüllt?

- Beispiel: Replikation mit write invalidate

Forderungen (vgl. Folie 45):

- Read your writes: Primär Eigenschaft des lokalen Caches; erfüllt

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 50/67

Strategie: Forderung erfüllt?

- Beispiel: Replikation mit write invalidate

Forderungen (vgl. Folie 45):

- Read your writes: Primär Eigenschaft des lokalen Caches; erfüllt
- Read other writes:
 - Schreibinstruktion eines Prozessors A invalidiert alle anderen Caches
 - Leseinstruktion von Prozessor B muss neuen Wert liefern - wo ist der?
 - Bei write-through Cache: Im Speicher; wird von dort geholt; erfüllt
 - Bei write-back Cache: Nur im Cache von A! Lesen muss Wert dort finden! Fragen für Mechanismus! Dann erfüllt.

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 50/67

Strategie: Forderung erfüllt?

- Beispiel: Replikation mit write invalidate

Forderungen (vgl. Folie 45):

- Read your writes: Primär Eigenschaft des lokalen Caches; erfüllt
- Read other writes:
 - ☐ Schreibinstruktion eines Prozessors A invalidiert alle anderen Caches
 - ☐ Leseinstruktion von Prozessor B muss neuen Wert liefern - wo ist der?
 - Bei write-through Cache: Im Speicher; wird von dort geholt; erfüllt
 - Bei write-back Cache: Nur im Cache von A! Lesen muss Wert dort finden! Fragen für Mechanismus! Dann erfüllt.
- Write serialization: Müssen im Mechanismus Reihenfolgen sicherstellen
 - ☐ Herausforderung: **danach** bei unterschiedlichen Prozessoren feststellen!

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 50/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

3.1 Problem

3.2 Strategien

3.3 Mechanismen

3.4 Weitere Beispiele

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 51/67

Wie Strategie realisieren?

- Aufgabe Mechanismus: Strategie realisieren
 - Noch besser: Unterschiedliche Strategien ermöglichen

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 52/67

Mechanismusbeispiel: Snooping Bus

Beispiel 15.1 (Replikation mit write invalidate und write-back Cache)

- Problem: Cache A muss wissen, wenn in Cache B eine Adresse aktualisiert wird, die ebenfalls in Cache A vorhanden ist

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 53/67

Mechanismusbeispiel: Snooping Bus

Beispiel 15.1 (Replikation mit write invalidate und write-back Cache)

- Problem: Cache A muss wissen, wenn in Cache B eine Adresse aktualisiert wird, die ebenfalls in Cache A vorhanden ist
- Idee: Adressen über Adressbus verfügbar!
 - Alle Caches lesen alle Adressinformationen mit, reagieren dann eigenständig
 - Es reicht **nicht**, nur Bus zwischen Cache und Speicher zu beobachten; Bus zwischen Cache und CPU relevant!
 - Sog. **Snooping Bus**

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 53/67

Mechanismusbeispiel: Snooping Bus

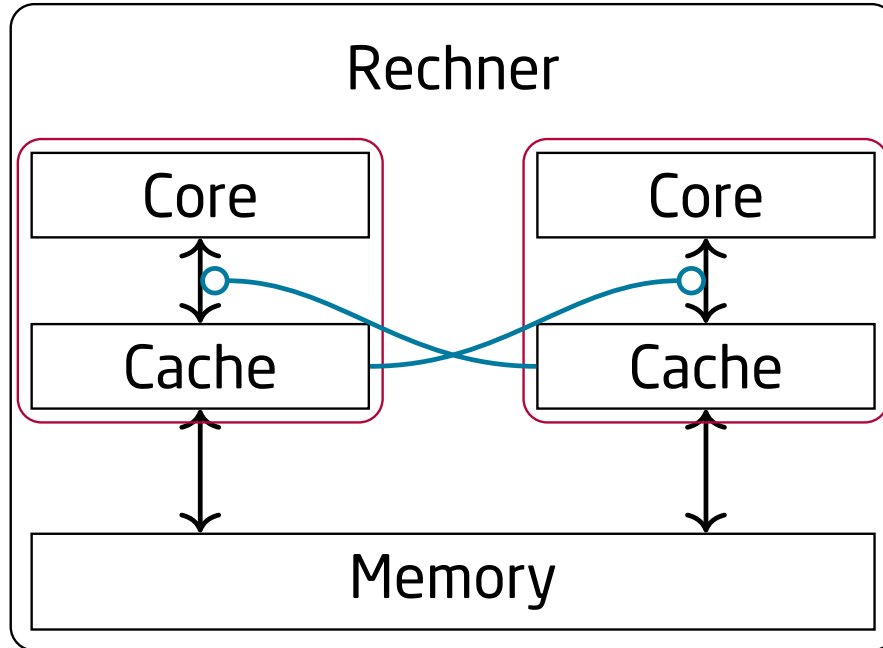


Abbildung 15.15: Snooping Bus Struktur: Caches beobachten Adressbus der jeweils anderen CPU-Cache-Verbindung, invalidieren ggf. vorhandene Einträge bei Schreiboperationen

Hierarchie
Parallelität
Cache-Kohärenz
Problem
Strategien
Mechanismen
Weitere Beispiele
Zusammenfassung
Material

- Invalidierung an **alle** Caches schicken
 - Aufwand ähnlich zu Snooping Bus
 - Geht das auch mit Updates?
- Explizite Verzeichnisse führen: In welchen Caches liegt welche Adresse?

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 55/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

3.1 Problem

3.2 Strategien

3.3 Mechanismen

3.4 Weitere Beispiele

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 56/67

Verallgemeinerung über Cache, Prozessoren, . . . hinaus?

- Wo finden wir noch (nach oben) verzweigende Speicherhierarchien?
- Mit Zwischenspeicher und Aktualisierungen?

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 57/67

Web-Proxy als Cache für einen Webserver

- Einfach: Nur ein Ort, an dem Updates entstehen
- **Timeout** als wichtiger Mechanismus
 - Kurzfristige Inkonsistenz ist akzeptabel!

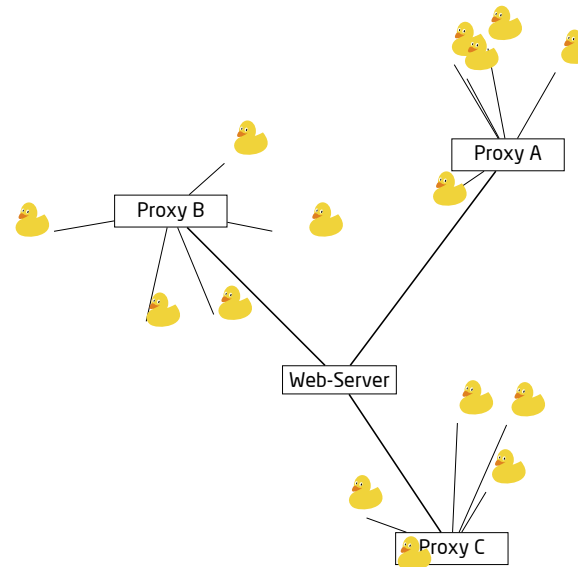


Abbildung 15.16: Web-Server mit Proxies als Caches

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 58/67

Domain Name System

Internet Domain Name System (DNS) als Cache für eine Namenstabelle [87]

- Authoritative DNS Server als “Original”
- Ähnlich zu Web-Proxy

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 59/67

- Replikation für komplexe Datenstrukturen
 - z.B. Kontostände bei Banken
- **Hohe** Konsistenzanforderungen
 - Sehr viel aufwändigere Vorschriften für Aktualisierungen
 - **Distributed Transactions**

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

[Weitere Beispiele](#)

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 60/67

Kerneigenschaften

Charakteristische Eigenschaften:

- Nach oben verzweigende Speicherhierarchie
- NUMA:

Definition 15.3 (Non-Uniform Memory Access (NUMA))

Ein System hat die NUMA-Eigenschaft, wenn der Zugriff auf unterschiedliche Adressen von unterschiedlichen Verarbeitungseinheiten aus uneinheitlich viel Zeit verbraucht. (Gegensatz: Uniform Memory Access (UMA)).

Hierarchie

Parallelität

Cache-Kohärenz

Problem

Strategien

Mechanismen

Weitere Beispiele

Zusammenfassung

Material

**GDS 15:
Speicherhierarchie**

H. Karl, WS 22/23

Folie 61/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

- Hierarchische Speicherstrukturen sind die natürlich Verallgemeinerung für einfache Caches
 - Caches mit mehreren Schichten
 - Andere Beispiele wie Web-Proxies
- Mehrere CPUs, eng mit Caches verbunden, ergeben Rechnerstrukturen wie Multiprozessor- oder Multi-Core-Systeme
 - Mit vielen Mischformen
 - Erfordert zur Auslastung viele Programme gleichzeitig oder nebenläufige Programme
- Verzweigt eine Speicherhierarchie nach oben entsteht stets das Cache-Kohärenz-Problem
 - Modell: Vertrag zwischen Programm und Speichersystem, welches Verhalten zu erwarten ist
 - Muss durch geeignete Strategie und Mechanismus realisiert werden

Hierarchie

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

GDS 15:
Speicherhierarchie

H. Karl, WS 22/23

Folie 63/67

Inhaltsverzeichnis

1. Hierarchie

2. Parallelität

3. Cache-Kohärenz

4. Zusammenfassung

5. Material

Hierarchie

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Referenzen I

- [87] *Domain names - concepts and facilities*. RFC 1034. Nov. 1987. doi: 10.17487/RFC1034.
- [AG96] S.V. Adve und K. Gharachorloo. "Shared memory consistency models: a tutorial". In: *Computer* 29.12 (1996), S. 66-76. doi: 10.1109/2.546611.
- [Fly66] M.J. Flynn. "Very high-speed computing systems". In: *Proceedings of the IEEE* 54.12 (1966), S. 1901-1909. doi: 10.1109/PROC.1966.5273.

Hierarchie
Parallelität
Cache-Kohärenz
Zusammenfassung
Material

DNS Domain Name System 77

MIMD Multiple Instruction, Multiple Data 34

MISD Multiple Instruction, Single Data 33

NUMA Non-Uniform Memory Access 79

SIMD Single Instruction, Multiple Data 32

SISD Single Instruction, Single Data 30

UMA Uniform Memory Access 79

Hierarchie

Parallelität

Cache-Kohärenz

Zusammenfassung

Material

Hierarchie
Parallelität
Cache-Kohärenz
Zusammenfassung
Material