Zentralübung Rechnerstrukturen im SS2007

Cache-Kohärenz und -Konsistenz

David Kramer
kramer@ira.uka.de

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur

03.07.08

Organisatorisches

Klausur

- Anmeldung zur Klausur via QISPOS
- Anmeldeschluss ist der 01.08.2008
- Danach besteht kein Anrecht mehr auf Klausurteilnahme!
- Abmeldeschluss ist der 10.08.2008
- Klausurtermin: 11.8.2008 um 13 Uhr

Cache - Motivation

Memory Bottleneck

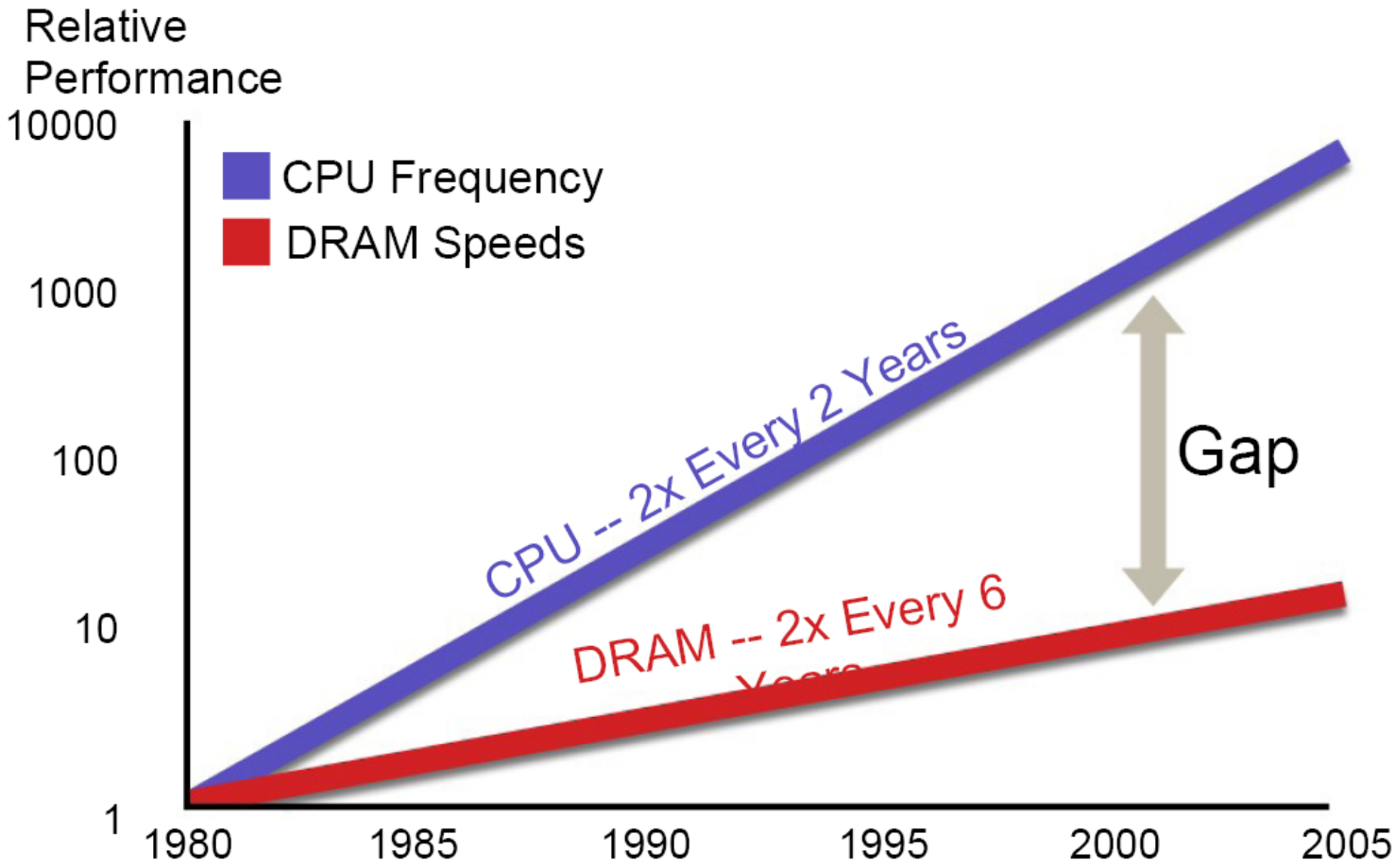


Abbildung: Quelle: http://blogs.sun.com/toddjobson/resource/

Cache - Motivation (forts.)

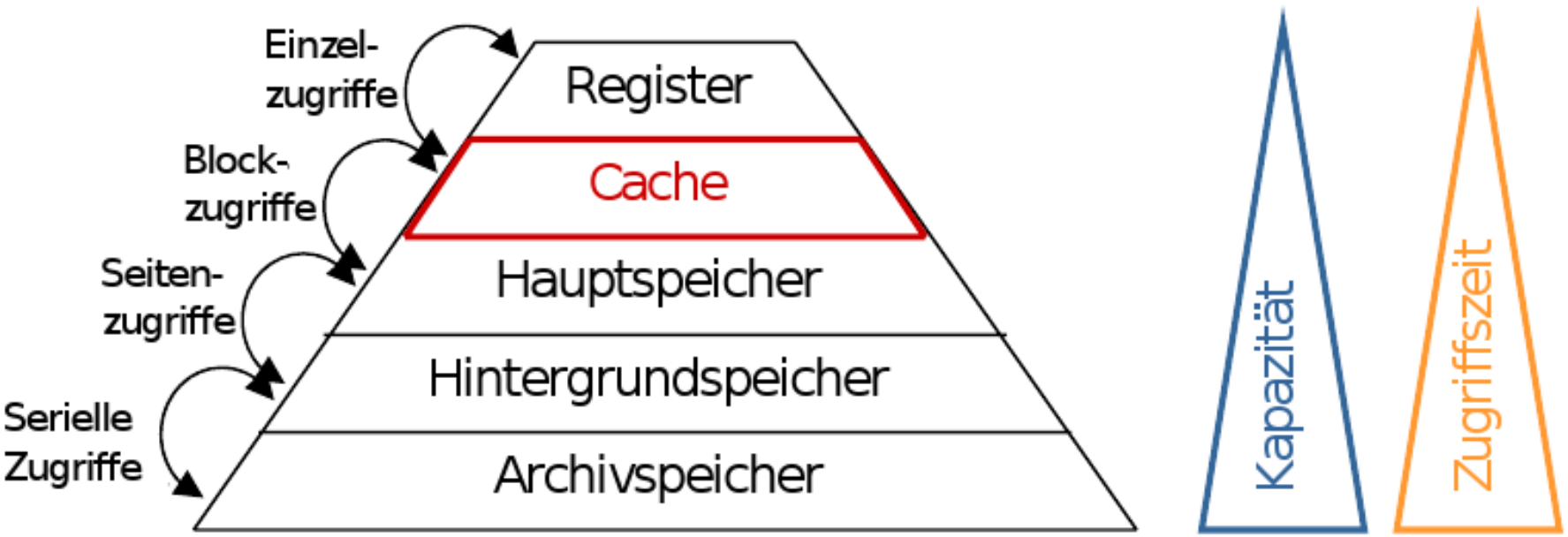


Abbildung: Speicherhierachie

- Ausnutzung der räumlichen und zeitlichen Lokalität von Anwendungen
- 90 / 10 - Regel

Cache - Aufbau

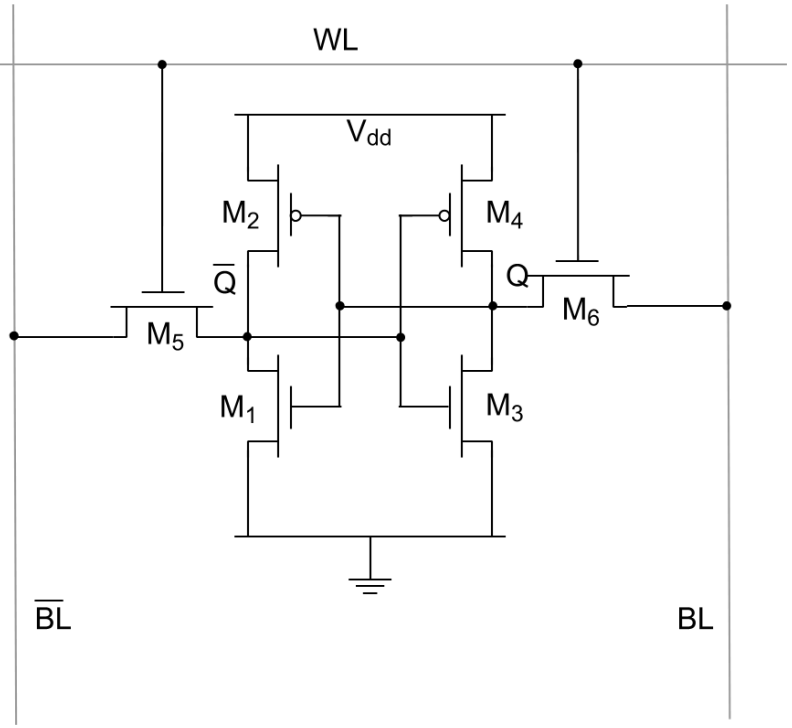


Abbildung: 6T-SRAM-Zelle

- Größe: über $100\,F^2$
- Sehr schnell

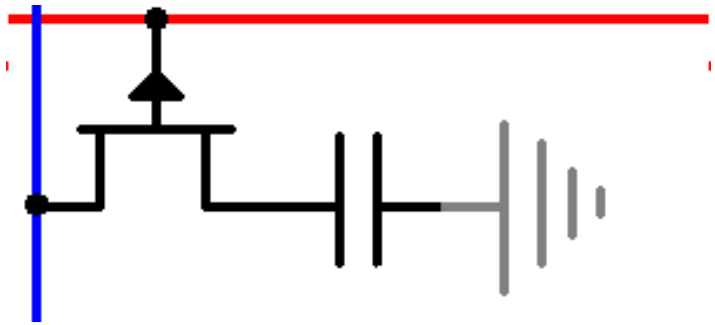


Abbildung: DRAM-Zelle

Quellen: Wikipedia

- Größe: ca. $6\,-\,10\,F^2$
- Destructive Read
- Refreshzyklen notwendig

Cache - Parameter

Organisation

- Direkt abgebildet (direct mapped)
- Satzassoziativ (set associative)
- Vollassoziativ (fully associative)

Hierachie

- Mehrere Level (L1, L2, L3)
- im Multiprozessorfall: private vs. shared
- inclusive vs. exclusive

Größe

Charakterisiert durch:

- Cachelinesize
- # Cachelines bzw. Assoziativität * # Sätzen

Cache - Parameter (forts.)

Ersetzungstrategie

- least recently used (LRU)
- least frequently used (LFU)
- Round Robin
- Random

Schreibstrategie

- write-through vs. write-back
- write-allocation vs. no write-allocation

Cachekohärenzprotokoll (im Multiprozessorfall)

- Busbasiert
- Verzeichnisbasiert

Aufgabe 1: Cacheleistung

Cache-Performance

- Hit-Rate r_H
- Miss-Rate $r_M = 1 - r_H$
- Zugriffszeit bei Hit t_H
- Zugriffszeit bei Miss t_M

Mittlere Zugriffszeit

$$t_a = \underbrace{r_H * t_H}_{Hit} + \underbrace{r_M * t_{Mem}}_{Miss}$$

Konkretisierung des Miss-Zeig

$$t_a = \underbrace{r_{H1} * t_{L1}}_{Hit\,L1} + \underbrace{r_{M1} * (\underbrace{r_{H2} * t_{L2}}_{Hit\,L2} + \underbrace{r_{M2} * t_{Mem}}_{Miss\,L2})}_{Miss\,L1}$$

Aufgabe 1: Cacheleistung (forts.)

Ein Prozessorkern verfügt über eine 2-stufige Speicherhierachie! Der L1-Cache hat eine Zugriffszeit von $t_{L1} = 1\,ns$, der L2-Cache hat eine Zugriffszeit von $t_{L2} = 6\,ns$ und beim Hauptspeicher liegt die Zugriffszeit bei $t_{HS} = 128\,ns$.
In dieser Aufgabe kann der Hauptspeicher sämtliche Daten speichern. Gehen Sie in Ihren Berechnungen davon aus, dass die Zugriffszeit der niederen Ebenen schon in den höheren Ebenen integriert ist.

a) Bei der Ausführung eines Benchmarks werden folgende Hit-Raten gemessen: $r_{H1} = 75\,\%$ und $r_{H2} = 60\,\%$. Berechnen Sie die mittlere Zugriffszeit t_a.

Aufgabe 1: Cacheleistung (forts.)

Ein Prozessorkern verfügt über eine 2-stufige Speicherhierachie! Der L1-Cache hat eine Zugriffszeit von $t_{L1} = 1\,ns$, der L2-Cache hat eine Zugriffszeit von $t_{L2} = 6\,ns$ und beim Hauptspeicher liegt die Zugriffszeit bei $t_{HS} = 128\,ns$.

a) Bei der Ausführung eines Benchmarks werden folgende Hit-Raten gemessen: $r_{H1} = 75\,\%$ und $r_{H2} = 60\,\%$. Berechnen Sie die mittlere Zugriffszeit t_a.

Antwort $t_a = t_{L1} * r_{H1} + (1 - r_{H1}) * (t_{L2} * r_{H2} + t_{HS} * (1 - r_{H2}))$

$$t_a = 1\,ns * 75\,\% + (1 - 75\,\%) * (6\,ns * 60\,\% + 128\,ns * (1 - 60\,\%))$$

$$t_a = 0.75\,ns + 0.25 * (3,6\,ns + 51,2\,ns) = \underline{14,45\,ns}$$

Aufgabe 1: Cacheleistung (forts.)

Ein Prozessorkern verfügt über eine 2-stufige Speicherhierachie! Der L1-Cache hat eine Zugriffszeit von $t_{L1} = 1\,ns$, der L2-Cache hat eine Zugriffszeit von $t_{L2} = 6\,ns$ und beim Hauptspeicher liegt die Zugriffszeit bei $t_{HS} = 128\,ns$.

b) Dieser Prozessorkern bildet die Grundlage eines CMP-Systems. Um die Kommunikation zwischen den Kernen zu beschleunigen, wird ein L3-Cache eingesetzt, dessen Zugriffszeit $t_{L3} = 18\,ns$ beträgt. Bei der Ausführung des obigen Benchmarks auf dem CMP-System wird eine Hitrate des L3-Cache von $r_{H3} = 60\,\%$ gemessen. Welche Leistungssteigerung wird durch den zusätzlichen L3-Cache erreicht?

Aufgabe 1: Cacheleistung (forts.)

b) $t_{L1} = 1\,ns$, $r_{H1} = 75\,\%$, $t_{L2} = 6\,ns$, $r_{H2} = 60\,\%$
$t_{L3} = 18\,ns$, $r_{H3} = 60\,\%$, $t_{HS} = 128\,ns$

Antwort Zugriffszeit:

$t_a = t_{L1} * r_{H1} + (1 - r_{H1}) * (t_{L2} * r_{H2} + (1 - r_{H2}) * (t_{L3} * r_{H3} + t_{HS} * (1 - r_{H3})))$

$t_a = 1\,ns * 75\,\% + (1 - 75\,\%) * (6\,ns * 60\,\% + (1 - 60\,\%) * (18\,ns * 60\,\% + 128\,ns * (1 - 60\,\%)))$

$t_a = 0,75\,ns + 0,25 * (3,6\,ns + 0.4 * (10,8\,ns + 51,2\,ns))$

$t_a = 0.75\,ns + 0,25 * (28,4\,ns)$

$t_a = 0.75\,ns + 7,1\,ns = \underline{7,85\,ns}$

Speed-Up:

$S = \frac{t_{a(L2)}}{t_{a(L3)}}$

$S = 14,45\,ns/7,85\,ns \sim \underline{1,84}$

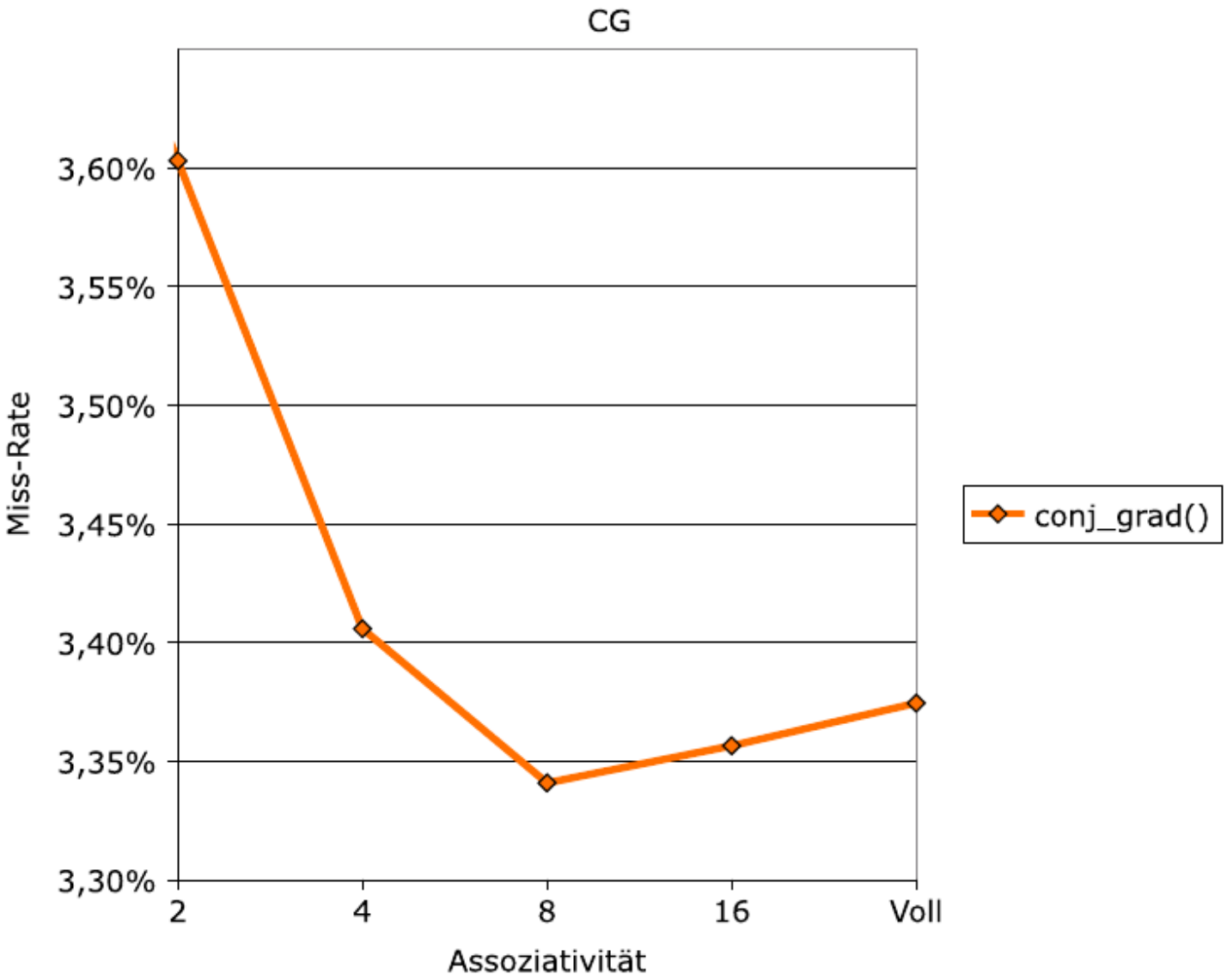
Aufgabe 2: Beweise

Beweisen oder widerlegen Sie folgende Behauptungen:
Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.

Aufgabe 2: Beweise - Motivation



Aufgabe 2: Beweise

Beweisen oder widerlegen Sie folgende Behauptungen:
Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.

Widerlegung:

Analog zu **Belady's Anomaly** [1]!

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

4-fach assoziativ

Folge:

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	**A**
B	**B**

C
D

4-fach assoziativ

A	**A**
B	**B**
C	
D	

Folge: $ABCDAB$

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	**A**	E
B	**B**	F

C
D

4-fach assoziativ

A	**A**
B	**B**
C	E
D	F

Folge: $ABCDABEF$

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	**A**	E
B	**B**	F

C	**C**
D	**D**

4-fach assoziativ

A	**A**	C
B	**B**	D
C	E	
D	F	

Folge: $ABCDABEFCD$

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	**A**	E	A	E
B	**B**	F	B	F

C	**C**
D	**D**

4-fach assoziativ

A	**A**	C	E
B	**B**	D	F
C	E	A	
D	F	B	

Folge: $ABCDABEFCDABEF$

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	**A**	E	A	E
B	**B**	F	B	F

C	**C**	**C**
D	**D**	**D**

4-fach assoziativ

A	**A**	C	E
B	**B**	D	F
C	E	A	C
D	F	B	D

Folge: $ABCDABEFCD(ABEFCD)^n$

Definition: A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zeiten Satz des 2-fach assoziativen Cache

Aufgabe 2: Beweise (forts.)

Beweisen oder widerlegen Sie folgende Behauptungen:
Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

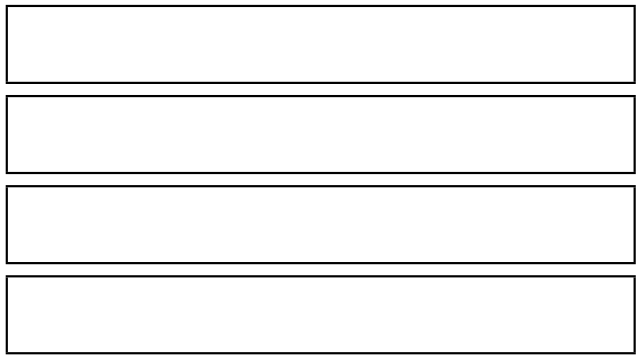
a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped



Vollassoziativ

Folge:

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A
B
C
D

Vollassoziativ

A
B
C
D

Folge: $ABCD$

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A
B
C
D E

Vollassoziativ

A E
B
C
D

Folge: $ABCDE$

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	**A**
B	**B**
C	**C**
D	E

Vollassoziativ

A	E
B	A
C	B
D	C

Folge: $ABCDEABC$

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	**A**
B	**B**
C	**C**
D	E D E

Vollassoziativ

A	E	D
B	A	E
C	B	
D	C	

Folge: $ABCDE(ABCDE)^n$

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 2: Beweise (forts.)

b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	**A**	**A**	
B	**B**	**B**	
C	**C**	**C**	
D	E	D	E

Vollassoziativ

A	E	D	C
B	A	E	
C	B	A	
D	C	B	

Folge: $ABCDE(ABCDE)^n$

Definition: A wird auf die Erste, B auf die Zweite, C auf die Dritte und D und E auf die vierte Cachezeile des DM-Cache abgebildet.

Verständnisfragen

a) Welche Eigenschaft von Anwendungen werden von Caches ausgenutzt?

Verständnisfragen

a) Welche Eigenschaft von Anwendungen werden von Caches ausgenutzt?

Antworten:

a) Räumliche und Zeitliche Lokalität

Verständnisfragen (forts.)

b) Welche Arten von Cache-Misses können unterschieden werden?

Verständnisfragen (forts.)

b) Welche Arten von Cache-Misses können unterschieden werden?

Antworten:

b) Conflict-, Capacity-, Compulsory-Misses

Im Mehrprozessorfall:

- Coherency-Miss
- Unterscheidung in: False-Sharing-Miss bzw. True-Sharing-Miss

Verständnisfragen (forts.)

c) Warum ist der Aufbau des Hauptspeichers aus SRAM-Zellen nicht sinnvoll?

Verständnisfragen (forts.)

c) Warum ist der Aufbau des Hauptspeichers aus SRAM-Zellen nicht sinnvoll?

Antworten:

c) SRAM-Zellen benötigen viel Chipfläche, deswegen ist der Aufbau des Hauptspeichers aus SRAM-Zellen entweder zu teuer oder man könnte nur geringe Kapazitäten anbieten.

Caches in Multiprozessorsystemen

Problem bei Verwendung von Caches in MP-Systemen

- Wahrung der Konsistenz
- CPUs operieren auf lokalen Kopien
- Daten in den Caches können sich von den Daten im Hauptspeicher unterscheiden

Konsistenz

Ein System ist konsistent, wenn alle Kopien eines Datenwortes im Hauptspeicher und den verschiedenen Cache-Speichern identisch sind

Cache-Kohärenz

Ein System ist Cache-Kohärent, wenn bei einem Zugriff immer auf das aktuellste Datum zugegriffen wird.

Kohärenzprotokolle

Bus-Snooping

- Write-Back Invalidate Protokoll
  - MSI [2]
  - MESI [2]
  - MOESI [3]
- Write-back Update Protokoll
  - Dragon [2]

Verzeichnisbasiert

Wird in DSM-Systemen benötigt. (siehe [2], Kapitel 8)

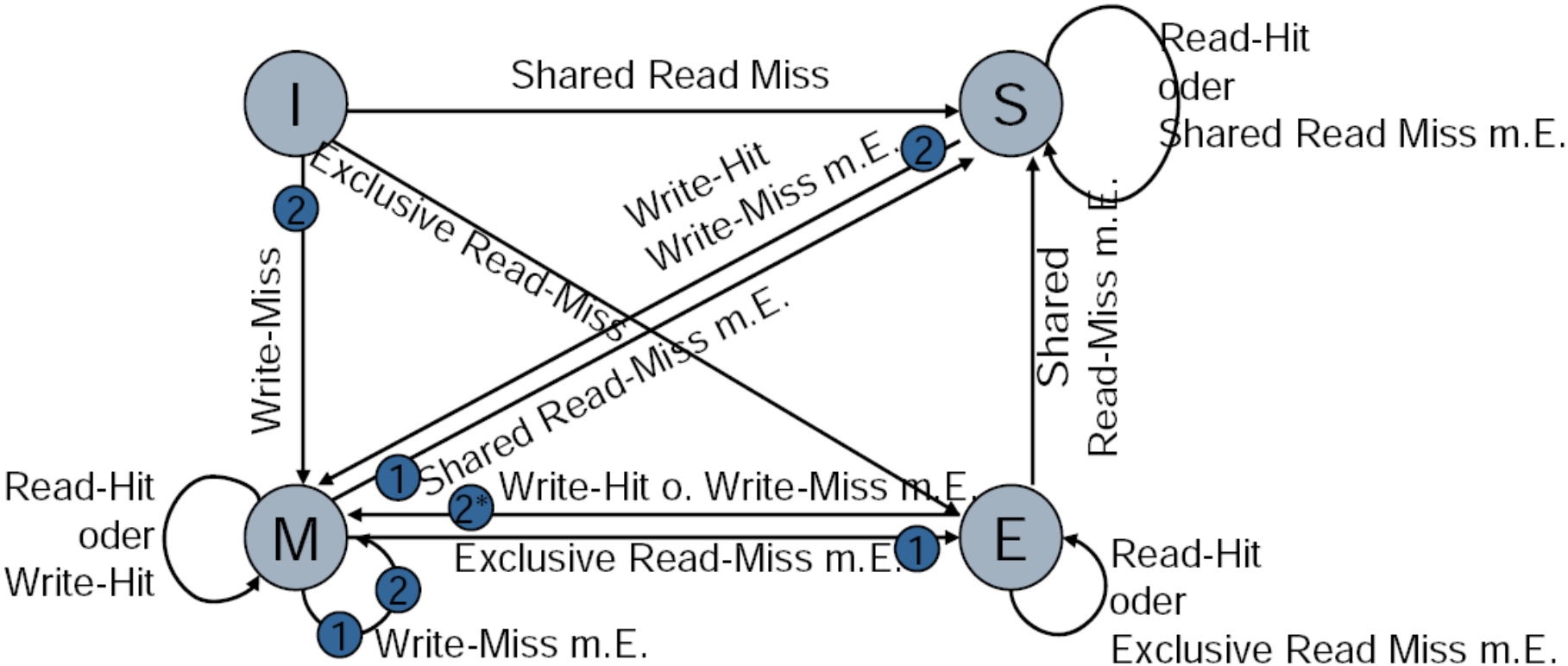
Bus-Snooping

- Beobachtung des Speicherbusses hinsichtlich Speicherzugriffe anderer Bus-Master
- Bus-Snooping erfordert einen **globalen Speicherbus**
- Jeder Cache muss um sog. **Snoop-Logik** und **Steuersignale** zu anderen Caches erweitert werden.
  - Invalidate-Signal
  - Shared-Signal
  - Retry-Signal

Das MESI-Protokoll

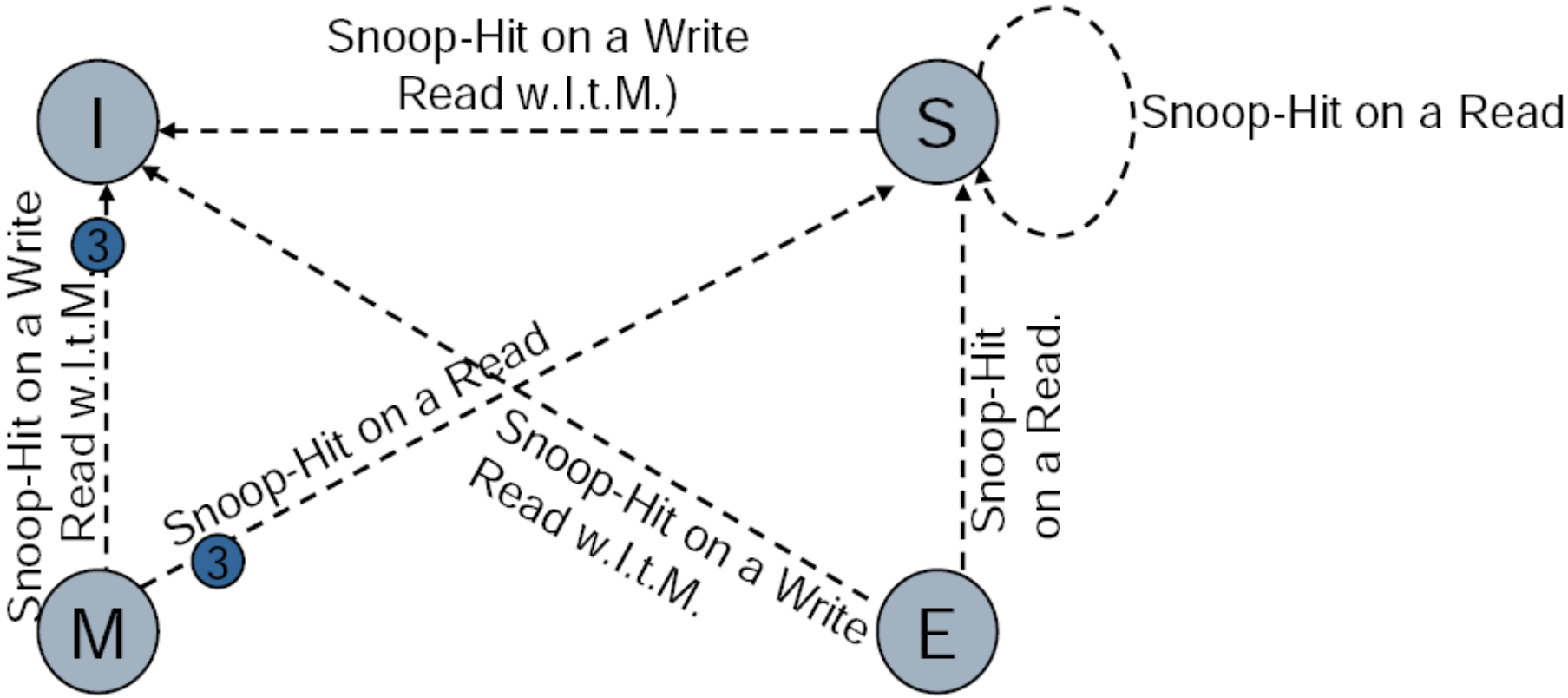
- Write-Back Invalidation Protokoll
- MESI: Zustandsautomat mit 4 Zuständen
  - M: Modified
  - E: Exclusive unmodified
  - S: Shared unmodified
  - I: Invalid
- Jede Cachezeile befindet sich in einem dieser Zustände
- d.h. jede Cachezeile wird um 2 Zustandsbits erweitert
- Zustandsübergänge werden durch lokale und entfernte Speicherzugriffe ausgelöst

MESI-Zustandsautomat (1)



1. Cache-Zeile wird in den Hauptspeicher zurückkopiert
2. Cache-Zeilen in den anderen Caches mit gleicher Blockadresse werden invalidiert

MESI-Zustandsautomat (2)



3 Retry-Signal wird aktiviert und danach wird die Cache-Zeile in den Hauptspeicher kopiert

Aufgabe 2.1: MESI Cachekohärenz-Protokoll

Ein Dreiprozessorsystem sei speichergekoppelt. Die Caches haben je eine Größe von zwei Cachezeilen, welche je genau ein Speicherwort aufnehmen können. Die Füllung des Caches erfolgt von der niedrigsten Cachezeile aufwärts, sofern noch freie Zeilen zur Verfügung stehen, andernfalls wird gemäß LRU-Strategie verdrängt. Als Cache-Kohärenzprotokoll komme das MESI-Protokoll zum Einsatz.

- Vervollständigen Sie die auf gegebene Tabelle: Geben Sie jeweils Inhalt der Cache-Zeile und MESI-Zustand an.

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6						
2	rd 2						
1	rd 4						
3	rd 4						
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4						
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3	3/S			3/S		
3	wr 3	3/I			3/I		3/M
2	wr 7						

Aufgabe 2.1: MESI Cachekohärenz-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3	3/S			3/S		
3	wr 3	3/I			3/I		3/M
2	wr 7			7/M			

MOESI-Protokoll

- Erweiterung des MESI-Protokolls um einen weiteren Zustand
- Neuer Zustand O: Owned (shared modified)

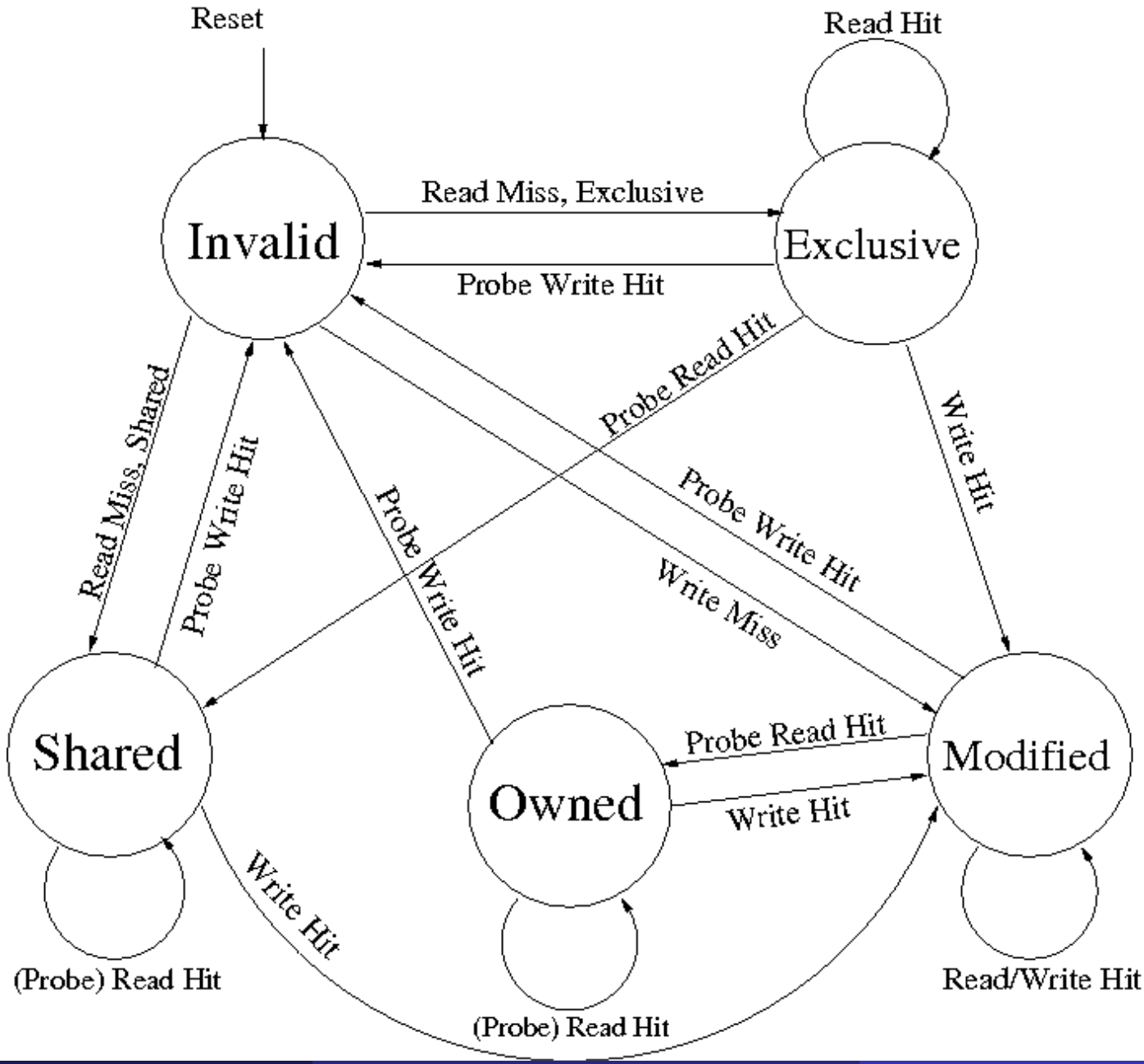
Vorteil

- Durch Cache-Cache-Transfers werden 2 Hauptspeicherzugriffe eingespart

Nachteile

- Komplexere Logik notwenig
- 3 Zustands-Bits nötig

MOESI Zustandsautomat (nach [3])



Aufgabe 2.2: MOESI Cache-Kohärenz-Protokoll

Ein Zweiprozessorsystem sei speichergekoppelt. Die Caches haben je eine Größe von drei Cachezeilen, welche je genau ein Speicherwort aufnehmen können. Die Füllung des Caches erfolgt von der niedrigsten Cachezeile aufwärts, sofern noch freie Zeilen zur Verfügung stehen, andernfalls wird gemäß LRU-Strategie verdrängt. Als Cache-Kohärenzprotokoll komme MOESI zum Einsatz. Der Cache sei initial leer.

- Vervollständigen sie die gegebene Tabelle: Geben Sie jeweils Inhalt der Cache-Zeile und MOESI-Zustand an.

Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1						
2	rd 2						
1	rd 2						
1	rd 4						
2	wr 6						
1	rd 6						
2	rd 2						
1	wr 6						
1	wr 4						
2	rd 3						
1	rd 5						
2	rd 6						
2	wr 5						

Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6						
2	rd 2						
1	wr 6						
1	wr 4						
2	rd 3						
1	rd 5						
2	rd 6						
2	wr 5						

Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6	6/S				6/O	
2	rd 2						
1	wr 6						
1	wr 4						
2	rd 3						
1	rd 5						
2	rd 6						
2	wr 5						

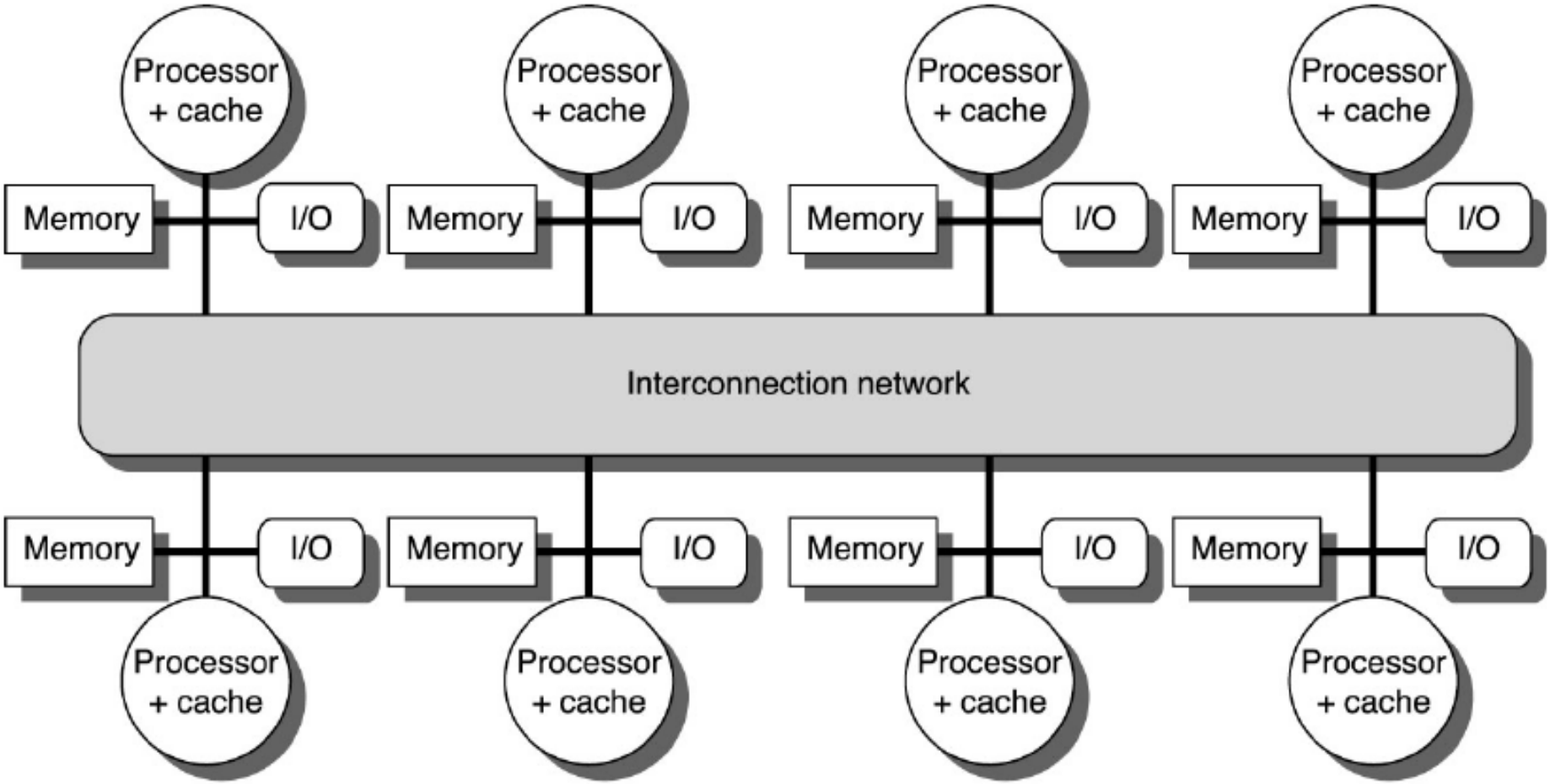
Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6	6/S				6/O	
2	rd 2		2/S		2/S		
1	wr 6						
1	wr 4						
2	rd 3						
1	rd 5						
2	rd 6						
2	wr 5						

Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6	6/S				6/O	
2	rd 2		2/S		2/S		
1	wr 6	6/M				6/I	
1	wr 4						
2	rd 3						
1	rd 5						
2	rd 6						
2	wr 5						

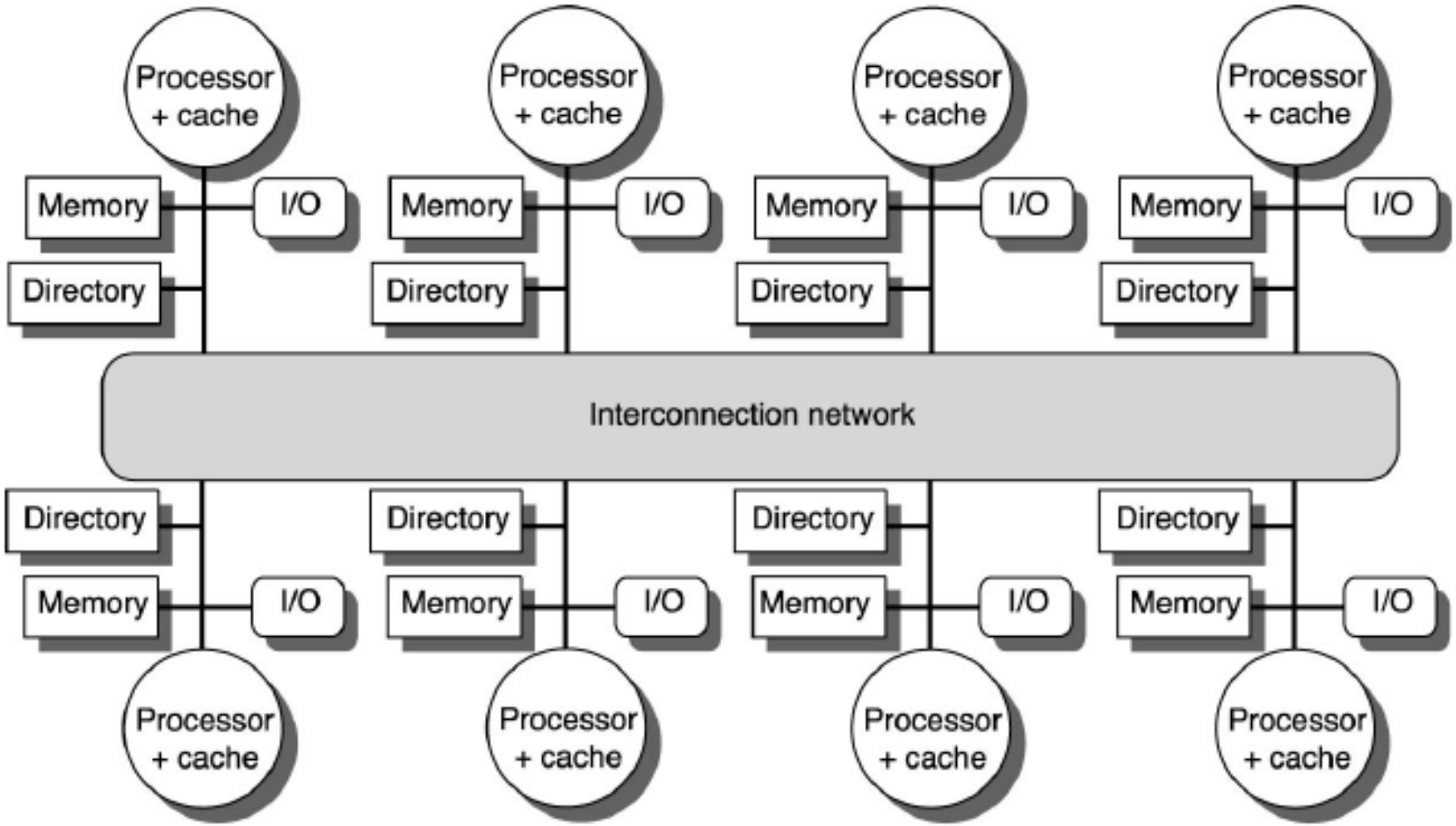
Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6	6/S				6/O	
2	rd 2		2/S		2/S		
1	wr 6	6/M				6/I	
1	wr 4			4/M			
2	rd 3					3/E	
1	rd 5		5/E				
2	rd 6						
2	wr 5						

Proz.	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	-	-	-	-	-	-
1	rd 1	1/E					
2	rd 2				2/E		
1	rd 2		2/S		2/S		
1	rd 4			4/E			
2	wr 6					6/M	
1	rd 6	6/S				6/O	
2	rd 2		2/S		2/S		
1	wr 6	6/M				6/I	
1	wr 4			4/M			
2	rd 3					3/E	
1	rd 5		5/E				
2	rd 6	6/O					6/S
2	wr 5		5/I		5/M		

Distributed-Shared-Memory (DSM)-System



Verzeichnisbasierte Cache-Kohärenzprotokolle



Verzeichnisbasierte Cache-Kohärenzprotokolle - Zusammenfassung

Einfachste Methode zur Wahrung der Kohärenz in DSM-Systemen

Nur Speicherzugriffe auf den lokalen Speicher werden in den Cache geladen!

- DSM-Systeme haben kein gemeinsamer Speicherbus, den eine Snooping-Logik überwachen könnte
- Herstellen der Cache-Kohärenz über Verzeichnistabellen
- Implementierung in Hard- oder Software möglich
- Die Tabelle protokolliert für jeden Blockrahmen, ob dieser in den lokalen oder einem entfernten Cache-Speicher als Cache-Block übertragen worden ist.
- Zustände werden analog zu den MESI-Zuständen definiert

Aufgabe 2.3: Verständnisfragen

a) Welche zusätzliche Hardware wird für das MESI-Protokoll benötigt?

Aufgabe 2.3: Verständnisfragen

a) Welche zusätzliche Hardware wird für das MESI-Protokoll benötigt?

Antwort:

1. Zustandsbits zur Speicherung des MESI-Zustands
2. Snooping-Logik zur Überwachung des Speicherbusses
3. Signale zu den weiteren Caches
   - Invalidate
   - Shared
   - Retry

Aufgabe 2.3: Verständnisfragen

b) Warum läßt sich MESI nicht in Distributed Shared Memory (DSM) Systemen einsetzen?

Aufgabe 2.3: Verständnisfragen

b) Warum läßt sich MESI nicht in Distributed Shared Memory (DSM) Systemen einsetzen?

Antwort:

b) In DSM-Systemen existiert kein gemeinsamer Speicherbus, den die Snooping-Logik überwachen könnte.

Aufgabe 2.3: Verständnisfragen

c) Welche Protokolle stellen in DSM-Systemen die Cache-Kohärenz sicher?

Aufgabe 2.3: Verständnisfragen

c) Welche Protokolle stellen in DSM-Systemen die Cache-Kohärenz sicher?

Antwort:

c) In DSM-Systemen kommen verzeichnisbasierte Cache-Kohärenzprotokolle zum Einsatz.

Literatur

1. Belady et al.: An anomaly in space-time characteristics of certain programs running in a paging machine, Communications of the ACM, Volume 12, Issue 6, Pages: 349 - 353, 1969
2. David E. Culler, Jaswinder Pal Singh: Parallel Computer Architecture - A Hardware/Software Approach Morgan Kaufmann, 1999, ISBN 1-55860-343-3
3. AMD64 Architecture Programmer's Manual Vol 2 'System Programming' `http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf`

Ausblick

- Nächste Übung: 17. Juli 2008
- Thema: Fehlertoleranz

Zentralübung Rechnerstrukturen im SS2007

Cache-Kohärenz und -Konsistenz

David Kramer
kramer@ira.uka.de

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur

03.07.08