

BP 2 Software-gestützte Pufferung: Virtueller Speicher

3 In Software realisierte Pufferspeicher

3.1 Virtueller Speicher

- Pufferung auf externen Speichern (vorwiegend Plattenspeichern) angelegter 'virtueller' Adreßraums im Arbeitsspeicher (paging)
- ◆ Speicherplatz wird nur für tatsächlich belegte Teile der virtuellen Adreßräume angelegt
- ◆ Adressierung der Puffer erfolgt in der Terminologie der Hardwareüberlegungen physikalisch! Ist also für Betriebssystemfunktionen unproblematisch
- ◆ Interpretation von Befehls- und Datenadressen erfolgt bezüglich des jeweils eingestellten Adreßraums
- ◆ Virtuelle Adreßräume können überlappend sein!
- ◆ Betreiben gemeinsamer Bereiche unproblematisch
- ◆ Zur Vereinheitlichung des Speicherzugriffs im Betriebssystem, werden Dateien beim Öffnen in den virtuellen Adreßraum gelegt (memory mapped files); können also in mehrere Adreßräume eingeblendet sein oder in einem Adreßraum mehrfach

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.1

BP 2 Software-gestützte Pufferung: Virtueller Speicher

◆ Vergleich der Vorgehensweisen bei 'caching' und 'paging'!

Ebene E_i	Arbeitsspeicher	Plattenspeicher
Ebene E_{i-1}	Cache	Arbeitsspeicher
Pufferadr.	kontextbezogen (virtuell) oder speicherbezogen (physikalisch)	kontextbezogen
Speicheradr.	physikalische Adr.	Blockadr. am Plattenspeicher
Index	Teil der Adresse	ermittelt über Tabelle
Index bestimmter Adresse	zeitl. unveränderlich	zeitl. veränderlich
Zeile	Pufferzeile	Kachel
Zeilenlänge	Pufferzeilenlänge	Kachelgröße
Markierung	versch. Varianten	physikalische Markierung
write on hit	write-back/through	write-back
write on miss	allocating/nonallocating	allocating
Bedeutungsgleichheit	möglich, falls virtuell	vermieden durch geeignete Kachelwahl
Mehrdeutigkeit	möglich, falls virtuell	vermieden durch geeignete Kachelwahl
Holstrategie	auf Anforderung	auf Anforderung oder (evtl. nur teilwei- se) vorausschauend

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.2

Ersetzungsstrategie	LRU	Second Chance
Plazierungsstrategie	durch Adr. festgelegt	Vermeidung von Bedeutungsähnlichkeit und Mehrdeutigkeit; Effizienzüberlegungen
Mehrdeutigkeit	falls virtuell adressiert, möglich	durch Wahl des Plazierungsortes vermieden
Bedeutungsähnlichkeit:	falls virtuell adressiert, möglich	durch Wahl des Plazierungsortes vermieden

◆ Wahl der Strategien

- **Holstrategie:**
 - Vorausschauend (prefetching)
 - Bei Fehlzugriff (on demand, demand paging)
- **Plazierungsstrategie:**
 - Wo frei (derzeit Normalfall)
 - Unter Berücksichtigung von Erfordernissen effizienter Pufferspeichernutzung
 - Bei NUMA-Architekturen Berücksichtigung der Threadaffinität
- **Ausräum-(Ersetzungs-)Strategie:**
 - Siehe Systemprogrammierung I



Kann durch vorausschauende Einlagerung die Zahl der Einlagerungen eines Prozesses reduziert werden?

- ◆ **Satz:** Zu jeder vorausschauend einlagernden Strategie kann eine Strategie konstruiert werden, die nur auf Anforderung einlagert und dabei höchstens so viele Einlagerungen verursacht wie die vorausschauende.

Beweis:

Sei A eine vorausschauende Strategie und r_0, r_1, r_2, \dots eine Referenzfolge.

Es sei S_t die Menge der nach dem t-ten Zugriff im Arbeitsspeicher befindlichen Seiten.

X_t bzw. Y_t seien die beim Übergang von S_{t-1} zu S_t ein- bzw. ausgelagerten Seiten, d. h.

$S_t = S_{t-1} + X_t - Y_t$ mit $X_t \cap Y_t = \emptyset$, $X_t \cap S_{t-1} = \emptyset$ und $Y_t \subseteq S_{t-1}$.

Konstruktion einer Anforderungsstrategie A' :

P_t seien die zum Zeitpunkt t von A aber nicht von A' eingelagerten Seiten.

R_t seien die zum Zeitpunkt t von A wieder ausgelagerten Seiten, die aber noch nicht von A' ausgelagerten wurden, also $R_t = (R_{t-1} + Y_t - X_t) \cap S'_t$.

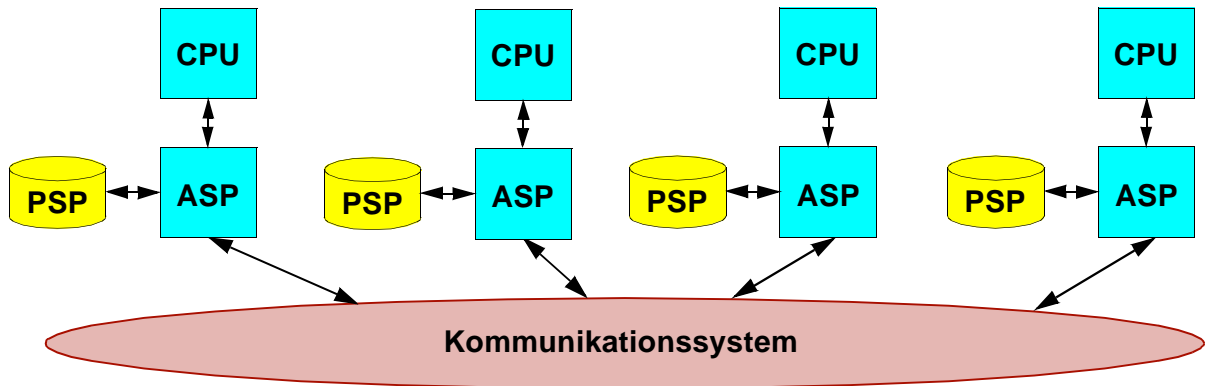
Mit diesen Bezeichnungen gilt $S'_t = S_t + R_t - P_t$.

- Bis erstmalig alle Kacheln belegt sind, muß A mindestens so viele Einlagerungen wie A' gemacht haben, da ja A' nur angesprochene Seiten einlagert.
- Sind zu einem Zeitpunkt unter A' alle Kacheln belegt sind und es muß eine Einlagerung erfolgen, so wird eine Seite aus $R_{t-1} + Y_t$ ausgelagert.
- Angenommen diese Vorgehensweise funktioniere bis t und es sei eine Seite einzulagern, die unter A bereits im Arbeitsspeicher ist und nicht zu Y_t gehört. Dann kann R_{t-1} nicht leer sein, denn es muß unter A eine der Seiten aus S'_{t-1} ausgelagert worden sein, um Platz für r_t zu haben. Wenn A' eine der Seiten aus $R_{t-1} + Y_t$ auslagert, so muß diese bis zum nächsten Ansprechen auch von A wieder eingelagert werden.
- Also ist auch bis zum nächsten Zeitpunkt die Zahl der Einlagerungen unter A mindestens so groß wie unter A' .

3.2 Verteilter, gemeinsam genutzter Speicher



Hardwarestruktur



09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.7



Programmierparadigmen

◆ Nachrichtensysteme

Prozesse (Threads) versenden und empfangen Nachrichten

◆ Gemeinsamer Speicher

Prozesse benutzen den ASP als Pufferspeicher



Anforderungen für gemeinsam genutzten Speicher

◆ Beispiel: parallele Bearbeitung von Satellitendaten mit evtl. unterschiedlichen Programmen

- Differenzierte Behandlung der Regionen bei fork: 'text'-Segmente können gemeinsam genutzt werden
- Manche Regionen (z. B. solche die Dateien beherbergen) können für den erzeugten Prozeß irrelevant sein
- Bei manchen Regionen kann der Anfangszustand relevant sein, aber es sollen erzeugender und erzeugter Prozeß eigene Kopien besitzen
- Algorithmen für Realisierung von gemeinsam genutzten Speicher sowohl für NORMA- als auch für NUMA-Architekturen von Interesse

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.8



Wichtige Gesichtspunkte

- Struktur und Granularität
- Kohärenz-Semantik
- Skalierbarkeit
- Implementierung
 - Datenlokalisierung und Zugriff
 - Kohärenzprotokoll (write-invalidate, write-update)
 - Ersetzungsstrategie
 - Seitenflattern (Thrashing)
 - Interaktionsmechanismen

09.06.99

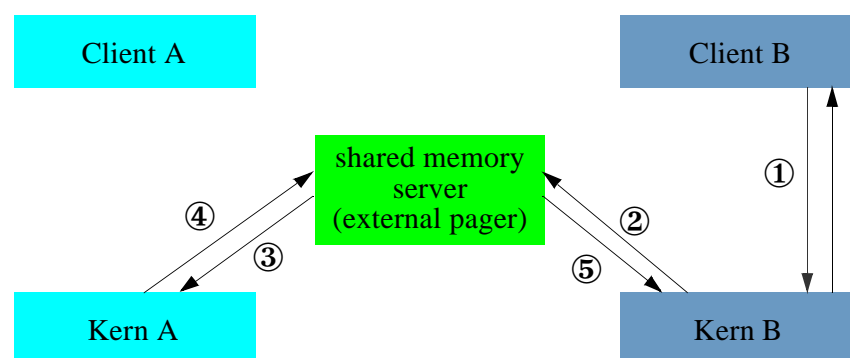
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.9



Grundsätzliche Vorgehensweise

Zuletzt von A angesprochene Seite wird von B lesend angesprochen



- ① Zugriff mit Seitenfehler
- ② data_request
- ③ lock_request (read only)
- ④ lock_completed
- ⑤ data_provided (read only)

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.10

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher



Probleme?

- ◆ **Performanz:** Eine Seite könnte ständig zwischen zwei oder mehr Prozessoren hin- und hergeschoben werden



Seitenflattern (thrashing)

- ◆ **Lösung:** Replizieren von Seiten



Kohärenzprotokoll erforderlich

- ◆ **Frage:** Anforderungen an das Kohärenzprotokoll anwendungsabhängig



abgestufte Kohärenzprotokolle

09.06.99

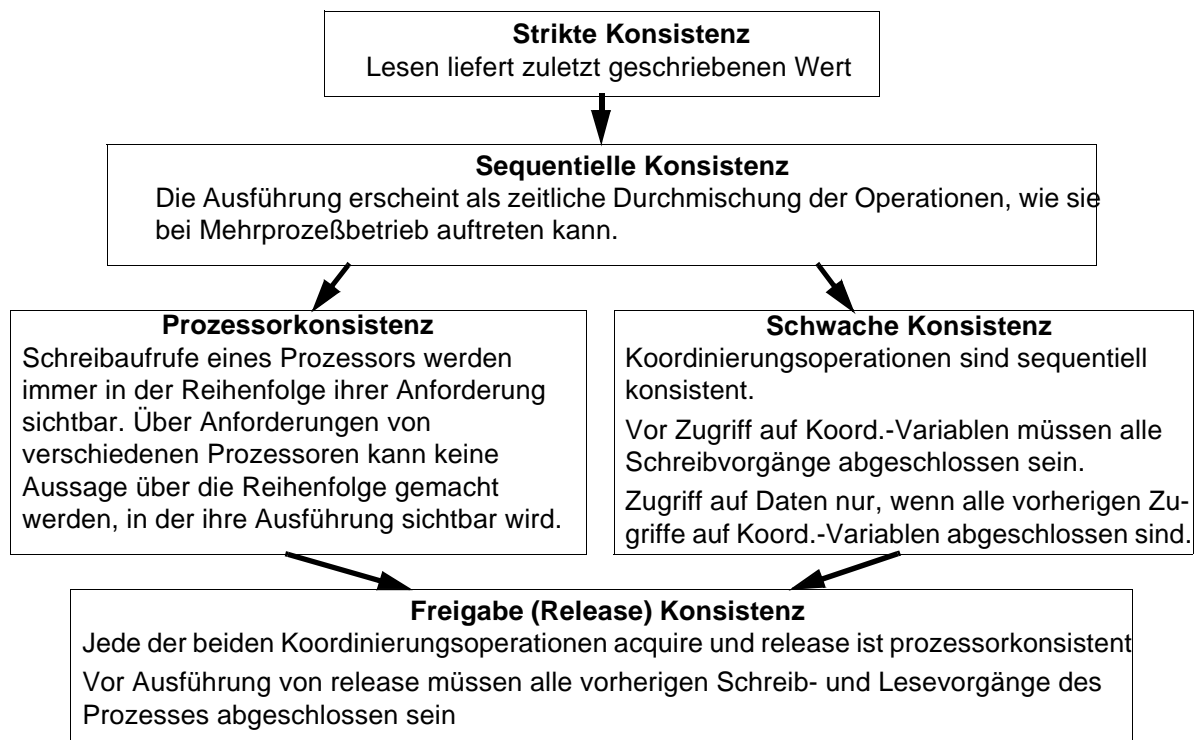
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.11

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher



Kohärenzarten



09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.12

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

□ Präzisierung der Vorstellungen

Gharachorloo, K.; Lenoski, D.; Laudon J.; Gibbons, P.; Gupta, A.; Hennessy, J.: Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors. Proc. 17th Annual Int. Symp. on Computer Architecture, May 28-31, 1990, Seattle Washington, pp.15-26.

◆ Bezeichnungen

- Eine **Load-Operation** eines Prozessors P_i wird zu einem Zeitpunkt als *ausgeführt bezüglich P_k* betrachtet, wenn ein späterer Anstoß einer Store-Operation durch P_k das Ergebnis der Load-Operation nicht beeinflussen kann.
- Eine **Store-Operation** eines Prozessors wird zu einem Zeitpunkt als *ausgeführt bezüglich P_k* betrachtet, wenn ein späterer Anstoß einer LOAD-Operation durch P_k den Wert liefert, den die Store-Operation oder eine spätere, die gleiche Speicherstelle betreffende geschrieben hat.
- Eine **Speicheroperation** gilt als *ausgeführt*, wenn sie bezüglich aller Prozessoren ausgeführt ist.
- Eine **Load-Operation** gilt als *global ausgeführt*, wenn sie ausgeführt ist und die Store-Operation, die die Ursache für den erhaltenen Wert ist, ausgeführt ist.

09.06.99

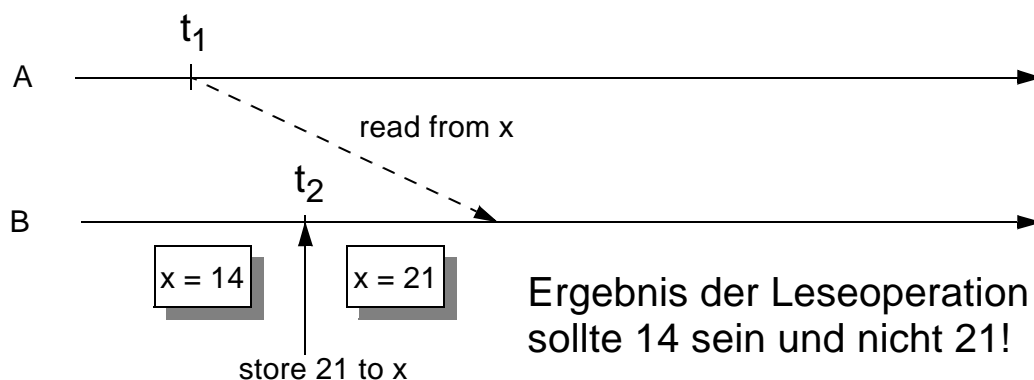
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.13

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

□ Strikte Konsistenz

Lesen liefert zuletzt geschriebenen Wert



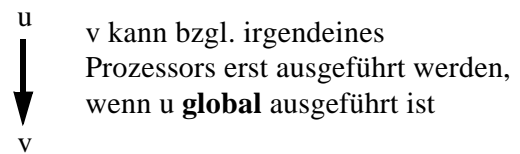
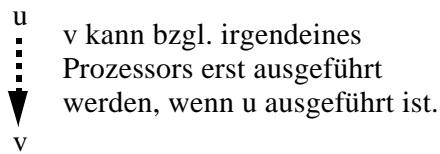
➡ Kaum realisierbar in einem verteilten System

09.06.99

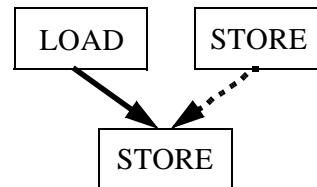
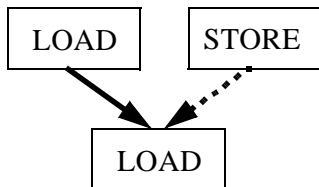
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.14

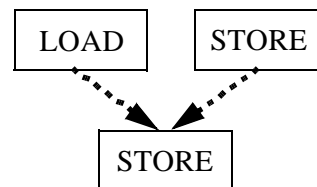
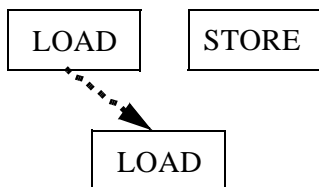
Graphische Charakterisierung verschiedener Konsistenzmodelle



Sequentielle Konsistenz



Prozessor-Konsistenz



09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.15

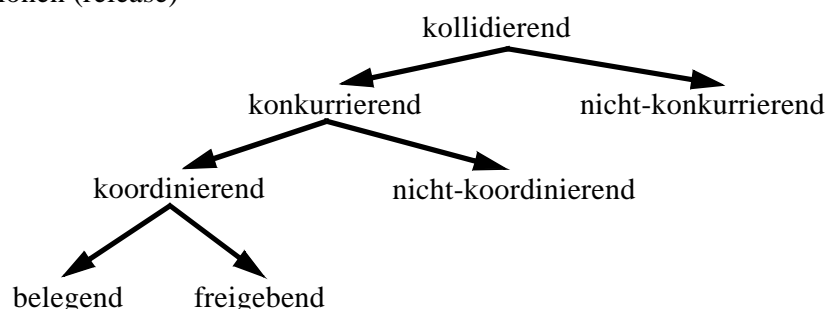
Klassifikation von Speicherzugriffen

Mehrere Speicherzugriffe sind

- **kollidierend** (conflicting), wenn sie den gleichen Speicherort betreffen und wenigstens einer eine STORE-Operation ist;
- **konkurrierend** (competing), wenn sie in Konflikt stehen und gleichzeitig zur Ausführung anstehen können,
- **koordinierend** (synchronizing), wenn sie zur Erzwingung von Ausführungsreihenfolgen konkurrierender Zugriffe benutzt werden.

Sie sind unterteilbar in

- Belegoperationen (acquire) und
- Freigabeoperationen (release)



09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.16

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

- ◆ Jede Speicheroperation eines Programms sei mit einer der Marken $m_kollidierend$, $m_konkurrierend$, $m_nicht_konkurrierend$, $m_koordinierend$, $m_nicht_koordinierend$, $m_belegend$ oder $m_freigebend$ markiert.
- ◆ Definition
Ein Programm enthält **genügend koordinierende Zugriffe**, wenn folgendes gilt:
Es seien u und v zwei beliebige, kollidierende Zugriffe der beiden Fäden P_u und P_v , von denen wenigstens einer mit $m_nicht_konkurrierend$ markiert ist. Dann muß bei jedem Ablauf in dem v nach (vor) u ausgeführt wird, wenigstens ein mit $m_koordinierend$ markierter Schreibzugriff (Lesezugriff) von P_u existieren und ein Lesezugriff (Schreibzugriff) von P_v , die u und v so trennen, daß der Schreibaufwurf vor dem Leseaufruf erfolgt. Der Schreibaufwurf muß mit $m_freigebend$, der Leseaufruf mit $m_belegend$ markiert sein.

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

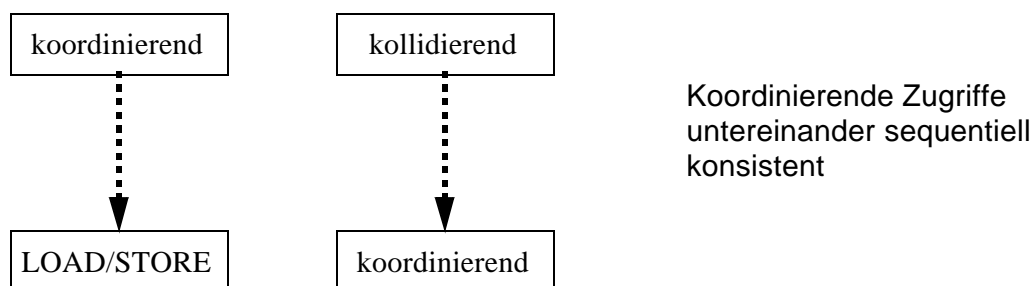
3.17

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

- ◆ Ein Programm heißt **richtig markiert**, wenn gilt:
 - Alle kollidierenden Zugriffe sind mit $m_kollidierend$ markiert,
 - alle konkurrierenden Zugriffe sind mit $m_konkurrierend$ markiert und
 - das Programm enthält genügend mit $m_koordinierend$ markierte Zugriffe.

□ Weitere Konsistenzmodelle:

- ◆ Schwache Konsistenz

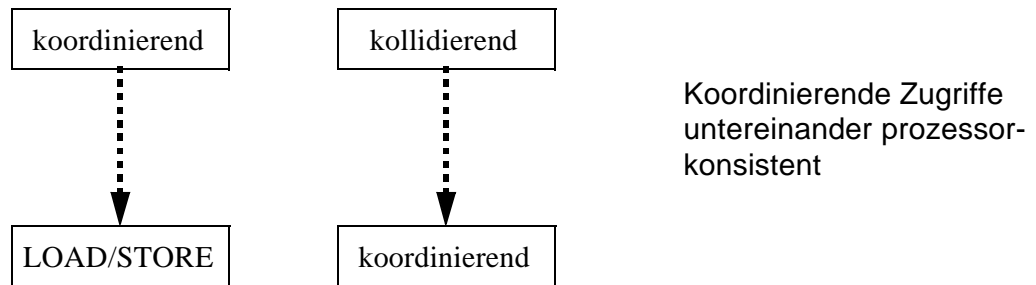


09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.18

◆ Freigabe-Konsistenz

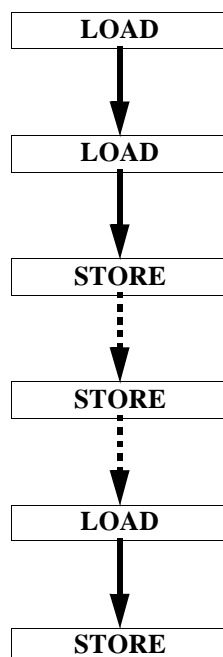


09.06.99

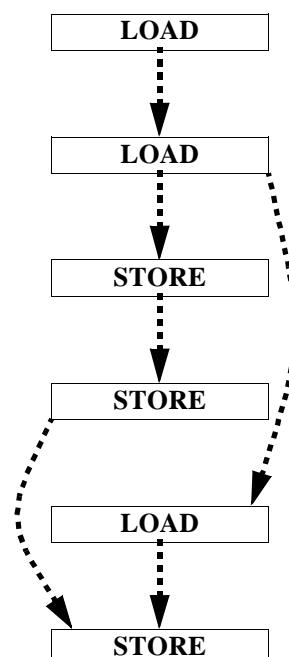
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.19

□ Modellhafte Beispiele



Sequentielle Konsistenz

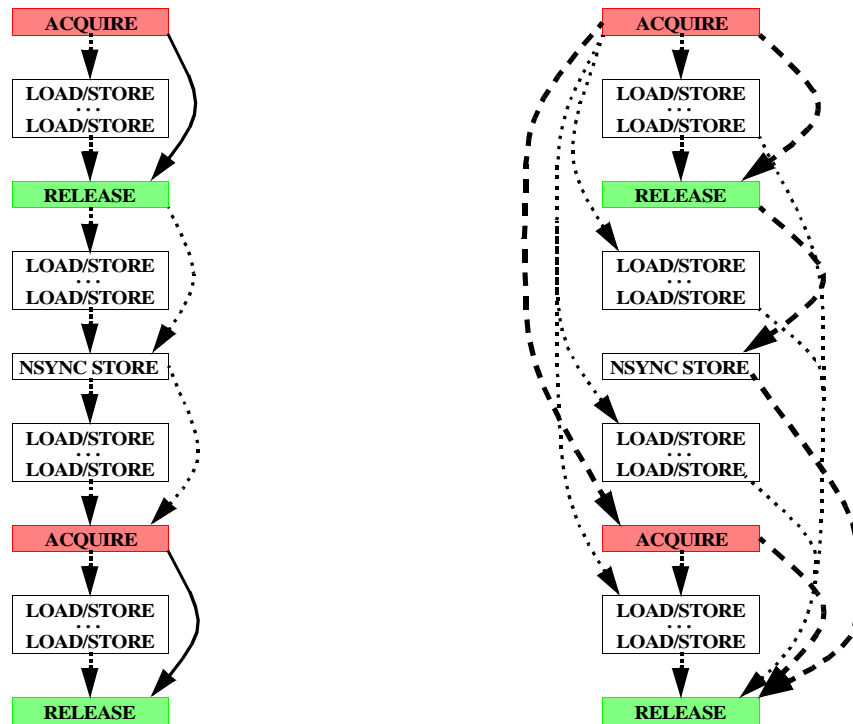


Prozessor-Konsistenz

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

3.20



Schwache Konsistenz

Freigabe-Konsistenz

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

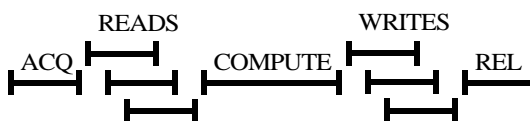
3.21



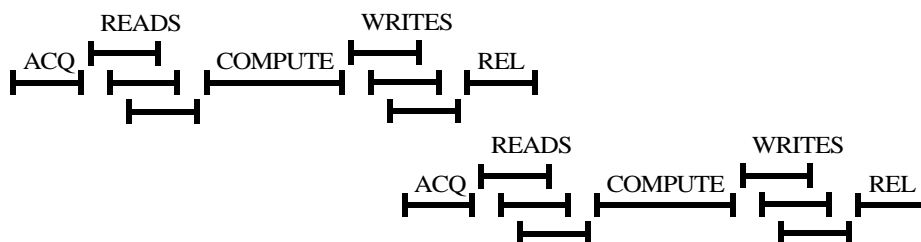
Überlappung bei der Bearbeitung einer verteilten Hash-Tabelle



Sequentielle Konsistenz



Schwache Konsistenz



Freigabe-Konsistenz

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.22

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

- **Satz: Richtig markierte Programme liefern bei Freigabe-Konsistenz dieselben Ergebnisse wie bei sequentieller Konsistenz.**

Beweis:

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.23

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

- **Algorithmen für die Platzierung von Seiten**

	nicht vervielfachend	vervielfachend
Nicht migrierend	zentralisiert (central server)	allgemein vervielfachend (full-replication)
migrierend	migrierend (migration)	vervielfachend für Lesezugriff (read-replication)

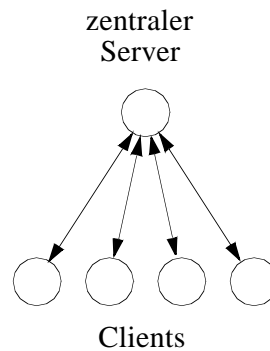
09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.24

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

◆ Zentraler Server (Central-server algorithm)



Client	zentraler Server
Datenanforderung senden	Anforderung entgegennehmen Datenzugriff durchführen Antwort senden
Antwort entgegennehmen	

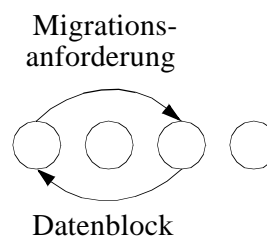
09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.25

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

◆ Migrationsalgorithmus



Client	Verwalter
Falls Block nicht lokal, Speicherort ermitteln und Anforderung senden	Anforderung entgegennehmen Block senden
Antwort entgegennehmen Datenzugriff durchführen	

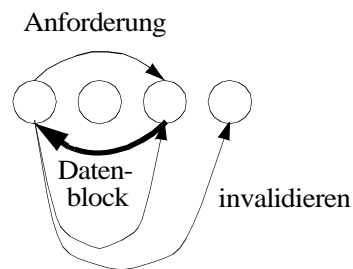
09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.26

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

◆ Vervielfachung für lesenden Zugriff (Schreibaufruf)



Client	Verwalter
Falls Block nicht lokal, Speicherort ermitteln und Anforderung senden	Anforderung entgegennehmen Block senden
Antwort entgegennehmen Invalidierung an alle anderen	Invalidierung vollziehen
Zugriff	

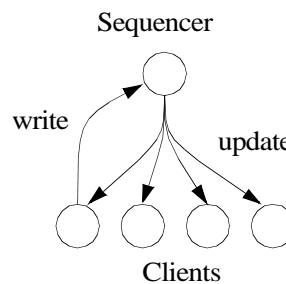
09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.27

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

◆ Allgemein vervielfachend



Client	Sequencer	Verwalter
Falls 'write' Daten an Sequencer	Sequenznummer anfügen Multicast	Daten empfangen 'update' lokal
Ausführung bestätigt 'update' lokal		

09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.28

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

◆ Vergleichende Analyse

Stumm, M.; Zhou, S.: Algorithms Implementing Distributed Shared Memory. Computer, May 1990, pp. 54-64.

pZeit für Versenden oder Empfangen einer kurzen Nachricht (einige Byte)
typischerweise einige msec

PZeit für Versenden oder Empfangen einer langen Nachricht (8 KByte)
typischerweise 20 bis 40 msec

SZahl beteiligter Knoten

rZahl der Leseaufrufe relativ zur Zahl der Schreibaufufe

fWahrscheinlichkeit für einen Zugriffsfehler bei Zugriff zu nicht-replizierten Datenblöcken unter dem Migrationsalgorithmus

f'Wahrscheinlichkeit für einen Zugriffsfehler bei Zugriff zu nicht-replizierten Datenblöcken bei Vervielfachung für lesenden Zugriff

$$\text{Zentraler Server } C_z = \left(1 - \frac{1}{S}\right) 4p \text{ Migration } C_m = f(2P + 4p)$$

$$\text{Leseervielfachung } C_{lv} = f' \left(2P + 4p + \frac{Sp}{r+1}\right) \text{ Allg. Vervielf. } C_{av} = \frac{1}{r+1}(S+2)p$$

Typischerweise: $P \approx 20p$

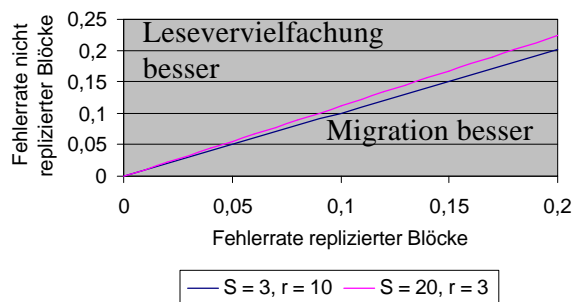
09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

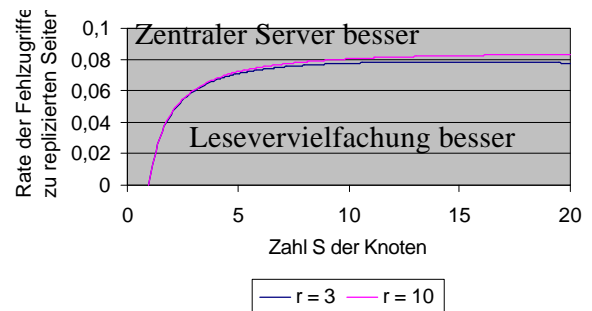
3.29

BP 2 Software-gestützte Pufferung: Verteilter gemeinsamer Speicher

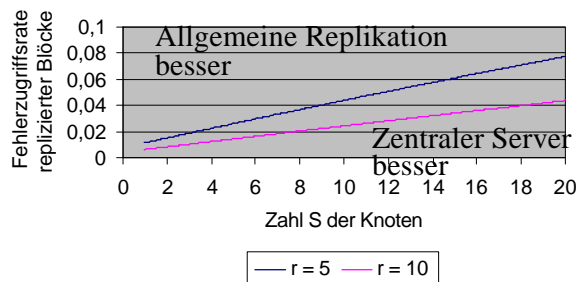
Migration versus Leseervielfachung



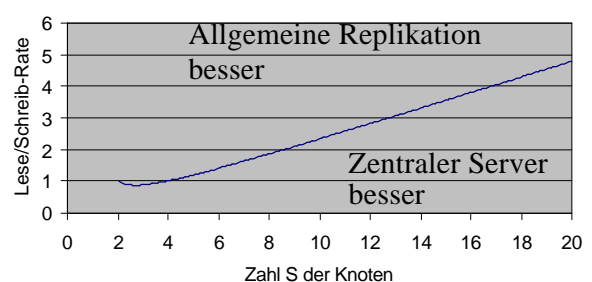
Zentraler Server versus Lesereplikation



Leseervielfachung versus allgemeine Vervielfachung



Zentraler Server versus allgemeine Replikation



09.06.99

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

3.30