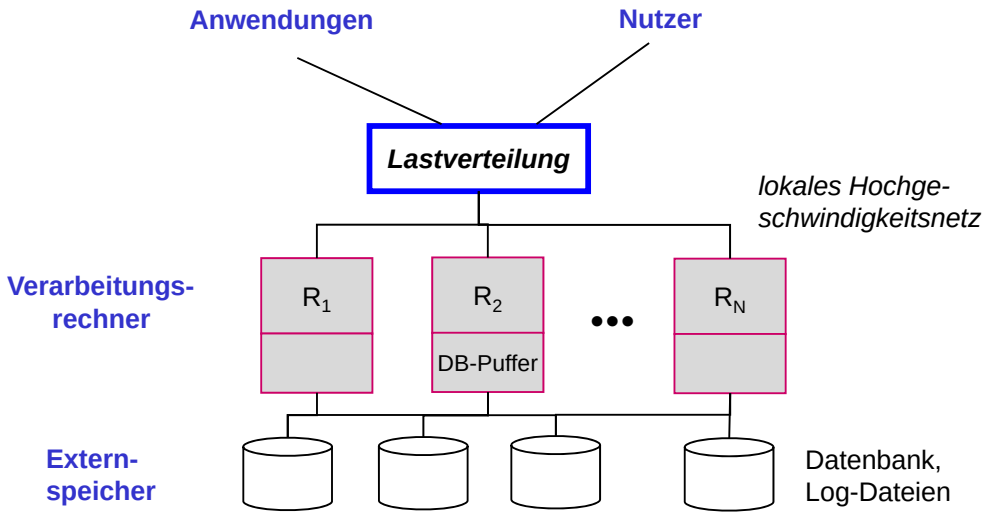
8. Shared-Disk-DBS

- Einführung
- Synchronisation
  - zentrale Sperrverfahren
  - Schreib-, Leseautorisierungen
  - verteilte Sperrverfahren
- Kohärenzkontrolle
  - Teilprobleme
  - On-Request-Invalidierung
  - Haltesperren
- Logging
- Nahe Kopplung
- Beispiel-Implementierungen
  - IBM Parallel Sysplex
  - Oracle

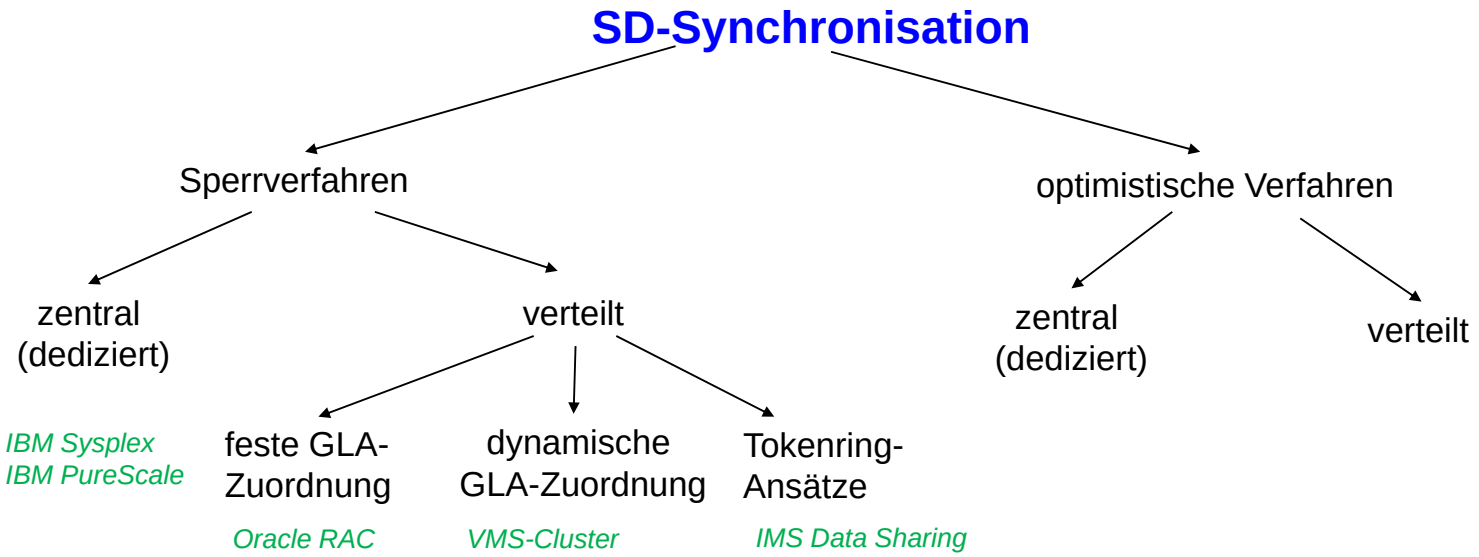
Grobaufbau eines Shared-Disk-PDBS



- keine DB-Partitionierung unter Rechnern
  - Änderungen in der Rechneranzahl relativ einfach verkraftbar
  - hohes Potenzial zur Lastbalancierung und Intra-Query-Parallelisierung
- lokale Rechneranordnung erforderlich
- Nachrichten-basierter Externspeicherzugriff (z.B. über SAN)

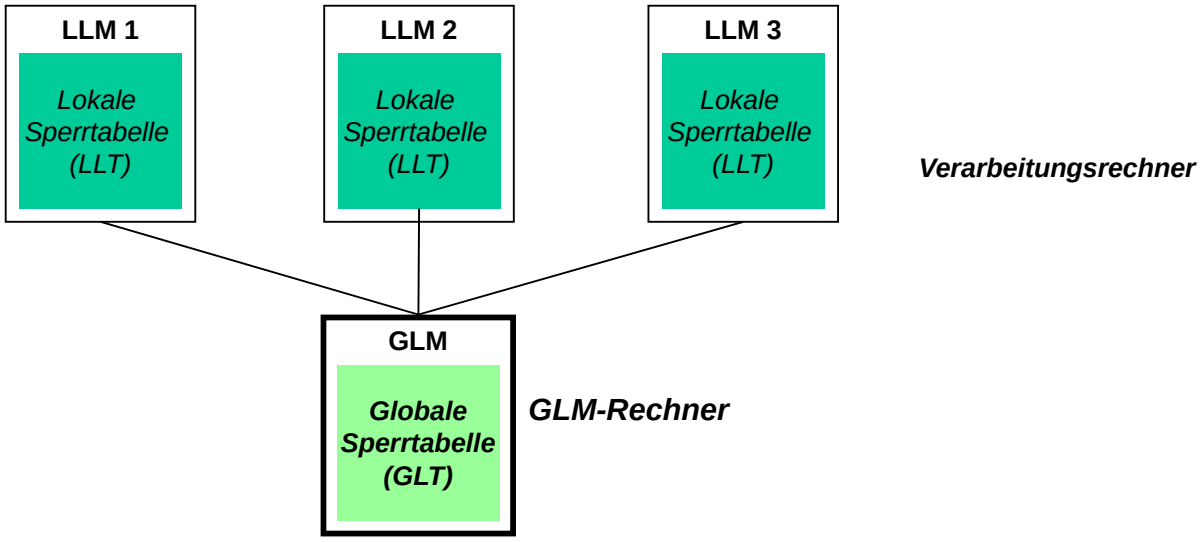
Shared-Disk-DBS: Neue Probleme

- Globale Synchronisation (Concurrency Control)
  - Wahrung der globalen Serialisierbarkeit
  - Möglichst wenige Synchronisationsnachrichten !
- Kohärenzkontrolle
  - dynamische Replikation von DB-Seiten im Hauptspeicher
  - Zugriff auf invalidierte Seiten zu vermeiden (mit geringem Aufwand)
- Lastverteilung
  - Unterstützung rechnerspezifischer Lokalität
- Logging und Recovery
  - Crash-Recovery durch überlebende Rechner
  - Erstellung einer globalen Log-Datei durch Mischen lokaler Log-Daten
- Nutzung einer nahen Kopplung
- Intra-Query-Parallelität



- Sperrverfahren verwenden ***globalen Lock-Manager (GLM)***
  - zu jedem Objekt hat genau einer der Rechner die globale Synchronisationsverantwortung **(GLA = Global Lock Authority)**
  - jeder Rechner hat lokalen Lock-Manager (LLM) für lokale Transaktionen

Zentrale Sperrverfahren

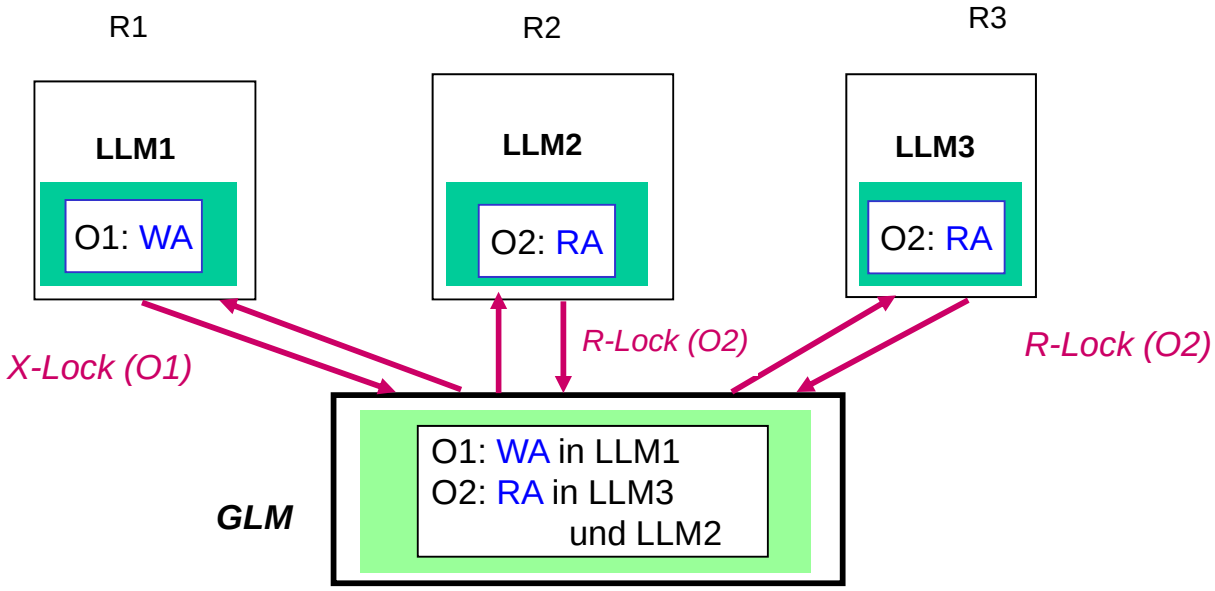


- zentrale Sperrverfahren: GLM für gesamte DB auf einem Rechner
  - Basisschema: 2 Nachrichten pro Sperranforderung
  - Nachrichtenbündelung erlaubt Reduzierung des Kommunikations-Overheads, verlängert jedoch die Antwortzeiten

Schreib- und Leseautorisierungen

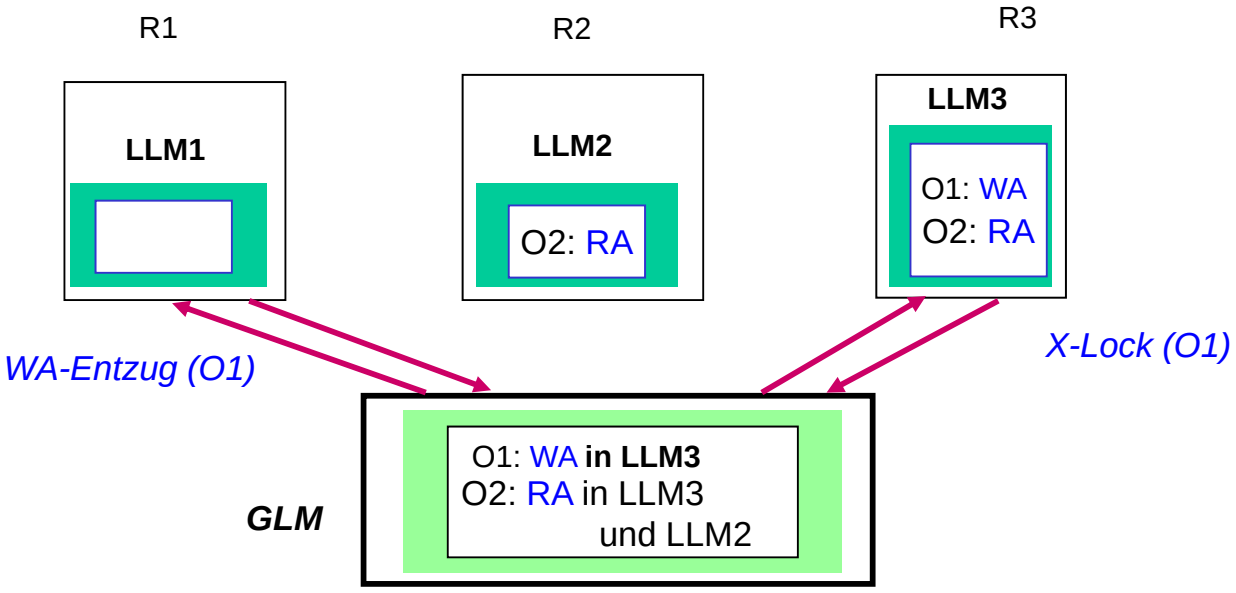
- Lokalität ermöglicht Einsparung globaler Sperranforderungen
- Vergabe von Schreib- und Leseautorisierungen an LLM
  - GLM erteilt bei Sperranforderung Schreibautorisierung, falls anfordernder Rechner allein "Interesse" an dem Objekt besitzt *(sole interest);* bei Leseanforderung wird Leseautorisierung erteilt, sofern keine Schreibanforderung vorliegt
- Schreibautorisierung ermöglicht lokale Gewährung und Freigabe von Schreib- und Lesesperren (Leseautorisierung nur von Lesesperren) durch den LLM
  => Einsparung von Nachrichten an den GLM
  - LLM behält Autorisierungen nach Transaktionsende, um spätere Anforderungen lokal zu bedienen
- Schreibautorisierung verzögert Sperrzuteilung an andere Rechner; Leseautorisierung verzögert Zuteilung von Schreibsperren (vorheriger Entzug der Autorisierungen durch GLM)

Einsatz von Autorisierungen



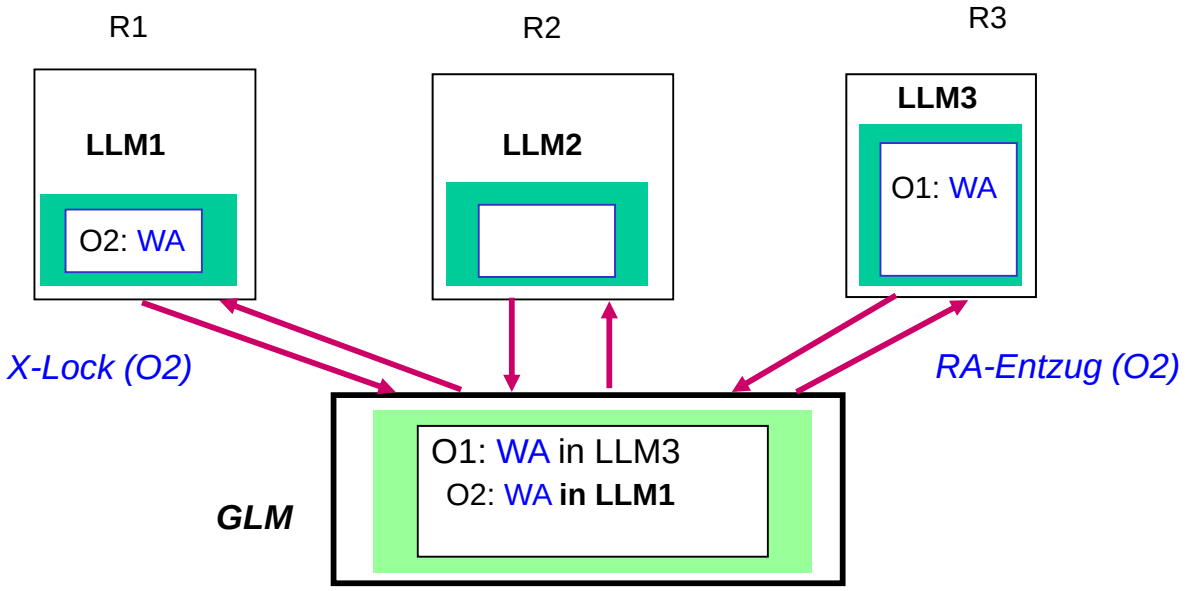
GLM = Global Lock Manager
LLM = Local Lock Manager
RA = Read Authorization
WA = Write Authorization

Einsatz von Autorisierungen: WA-Entzug



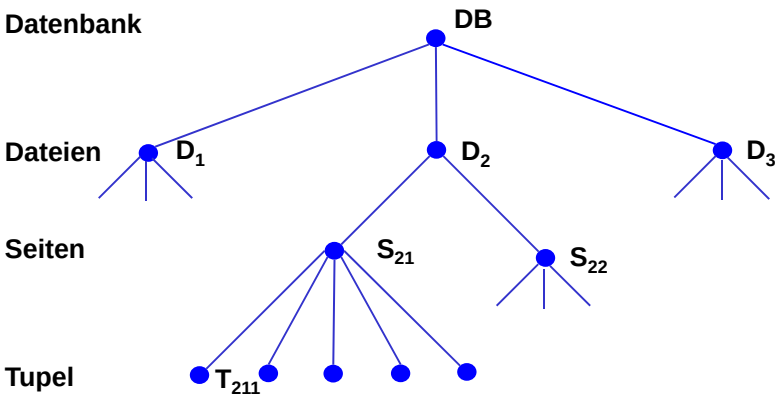
GLM = Global Lock Manager
LLM = Local Lock Manager
RA = Read Authorization
WA = Write Authorization

Einsatz von Autorisierungen: RA-Entzug



GLM = Global Lock Manager
LLM = Local Lock Manager
RA = Read Authorization
WA = Write Authorization

Hierarchische Autorisierungen



- Autorisierungen können auf mehreren Ebenen der Objekthierarchie verwaltet werden
- Schreibautorisierung für Datei beinhaltet z. B. eine *implizite Schreibautorisierung* für alle zugehörigen Seiten und Sätze => vollkommen lokale Synchronisierung auf Datei
- Leseautorisierungen für Datei ermöglicht lokale Vergabe/Rückgabe von Lesesperren für alle Objekte der Datei

Beobachtungen

- Instabilität von Schreibautorisierungen
  - eine externe Referenz verursacht bereits Entzug der Schreibautorisierung
  - "wichtige" DB-Bereiche werden i.a. von mehreren Rechnern referenziert
  - Gefahr häufiger Sole-Interest/WA-Wechsel (teuer)
- Leseautorisierungen sind stabiler
  - erfordert keine *rechnerspezifische* Lokalität
  - geringere Abhängigkeiten zur Partitionierbarkeit der Last und Lastverteilung
  - unterstützt parallele Lesezugriffe auf Objekt in verschiedenen Rechnern

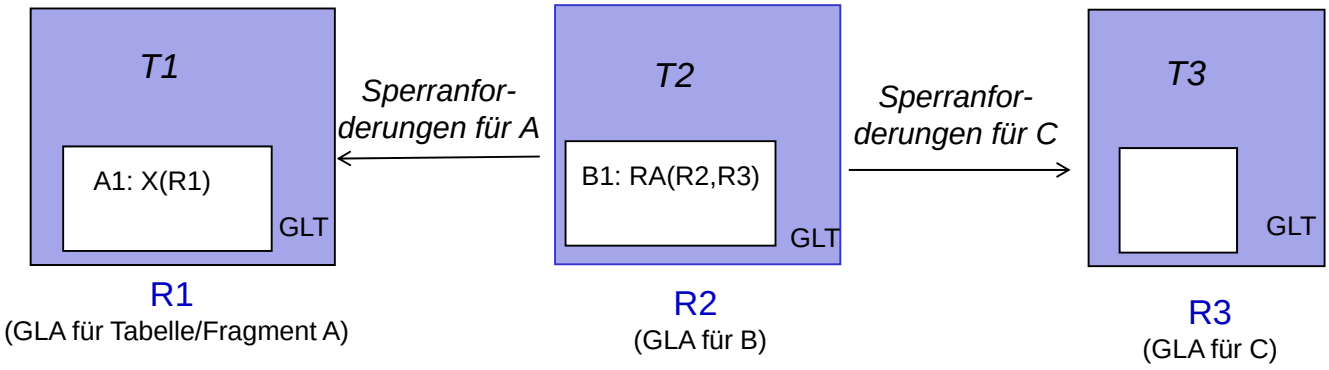
Verteilte Sperrverfahren

- Generelle Probleme zentraler Sperrverfahren:
  - Engpass-Gefahr
  - Verfügbarkeitsprobleme

  => Verteilung der Synchronisationsverantwortung auf mehrere Rechner
- Unterfälle
  - Feste vs. dynamische Verteilung der GLA
  - Synchronisation auf dedizierten Rechnern oder allgemeinen Verarbeitungsrechnern
  - Art der GLA-Partitionierung (Mapping Objekt-ID -> GLA-Rechner) analog zur Fragmentierung in Parallelen DBS, z.B.
    - Hash-Partitionierung oder
    - logische Fragmente (Tabellen, Tabellenbereiche)

Sperrverfahren mit logischer GLA-Verteilung

- Ziel: Koordinierung von GLA- und Lastverteilung zur Minimierung von Kommunikation
  - lokale Sperrvergabe für Objekte mit GLA auf Verarbeitungsrechner der referenzierenden Transaktion
- Nutzung von Leseautorisierungen sinnvoll

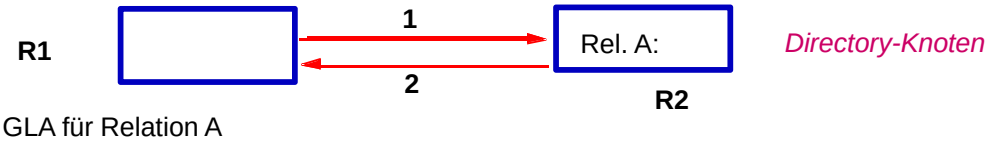


Dynamische GLA-Zuordnung

- Rechner, an dem ein Objekt zuerst referenziert wird, bekommt GLA für dieses Objekt
- Verwaltung der GLA-Zuordnung:
  - zentrale Tabelle
  - replizierte Tabelle
  - *(hash-) partitionierte Tabelle*
- zusätzliche Nachrichten zur Feststellung des GLM-Rechners
- GLA kann bei häufigen externen Sperranforderungen migrieren
- Beispielrealisierung: Distributed Lock Manager (DLM) für VMS-Cluster

Dynamische GLA-Zuordnung: Beispiel

1) Erste Sperranforderung für Relation A (in Rechner R1)



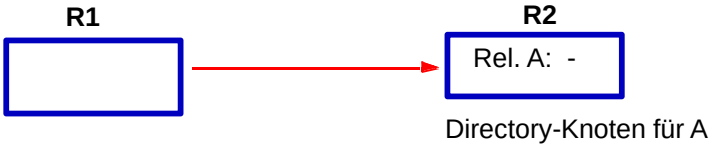
=> R1 erhält GLA für Relation A

alle weiteren Sperranforderungen in R1 bezüglich Relation A werden lokal synchronisiert

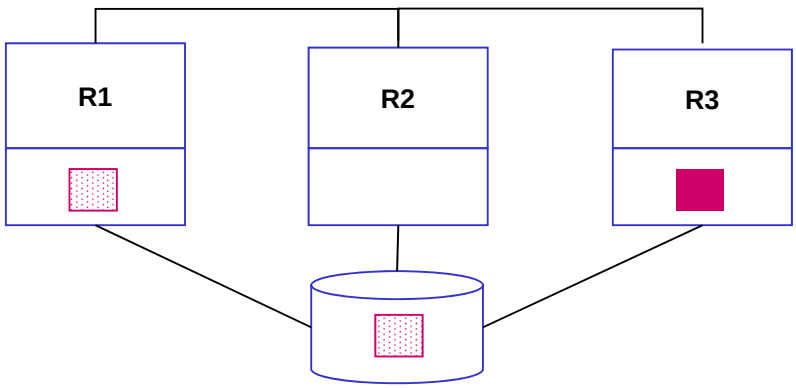
2) Sperranforderung auf Relation A in Rechner R3

=> 4 Nachrichten für Sperre (R3 vermerkt sich, daß R1 GLM für A ist
(für Sperrkonversionen, Sperrfreigabe)

3) Freigabe der letzten Sperre auf A (optional)

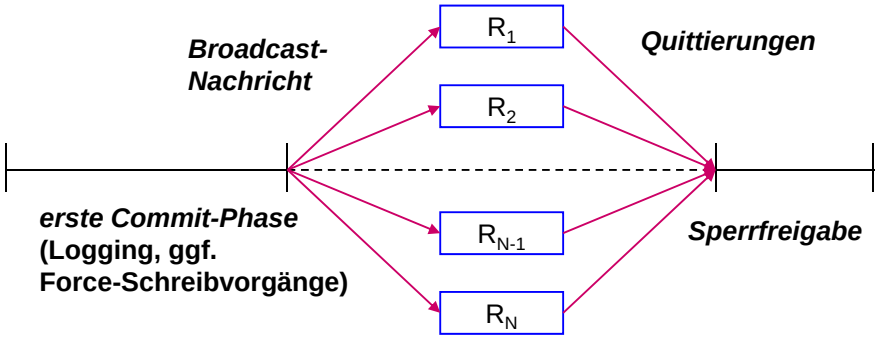


Kohärenzkontrolle



- Teilprobleme
  1. Vermeidung oder Erkennung von Pufferinvalidierungen
     - Broadcast-Invalidierung
     - On-Request-Invalidierung
     - Vermeidung ("Buffer Purge" bzw. Haltesperren)
  2. Update-Propagierung: Bereitstellung der aktuellen Objektversionen
     - direkter Transfer von Seiten vs. Seitentausch über Externspeicher
     - Einfluß der Ausschreibstrategie (FORCE vs. NOFORCE)

Broadcast-Invalidierung



- Broadcast-Nachricht am Ende jeder Update-Transaktion zur Meldung, welche Seiten geändert wurden
  - betroffene Seiten werden in anderen Rechnern aus dem Puffer eliminiert
  - Schreibsperren werden erst nach Quittierung aller Broadcast-Nachrichten freigegeben
    => Erhöhung der Sperrdauer und Konfliktgefahr
- sehr hoher Kommunikations-Overhead
  - Overhead pro Transaktion steigt mit der Rechneranzahl
    => für größere Anzahl von Rechnern unbrauchbar
- in frühen SD-Implementierungen zusammen mit FORCE eingesetzt: Ausschreiben der geänderten Seiten vor Commit
  -> Update-Propagierung mit Austausch von Änderungen über Platte

Update-Propagierung bei NOFORCE

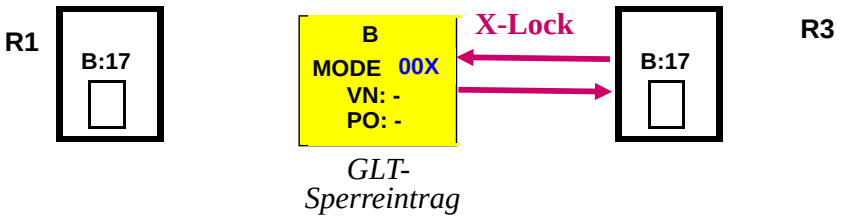
- NOFORCE:
  - Geänderte Seiten werden bei Commit aus Performance-Gründen nicht auf Externspeicher durchgeschrieben
  - Externspeicher-Version einer Seite potentiell veraltet
- Einführung eines Page-Owners für geänderte Seiten
  - übermittelt aktuelle Version einer Seite an anfordernde Rechner
  - schreibt geänderte Seite – asynchron - auf Externspeicher/Platte zurück (danach wird kein Page-Owner mehr benötigt)
- Realisierungs-Alternativen
  - *dynamische Page-Owner-Zuordnung*: Rechner, an dem die letzte Änderung erfolgt, wird zum Page-Owner
  - *feste Page-Owner-Zuordnung*: Änderungen werden beim Unlock zum Page-Owner transferiert (falls die Änderung an einem anderem Rechner durchgeführt)

On-Request-Invalidierung

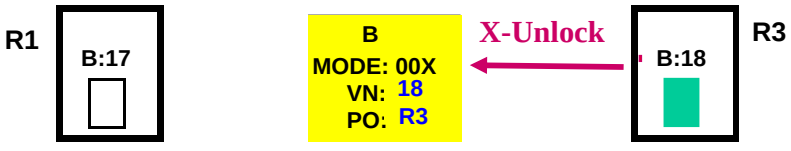
- Versionsnummer (Zähler) im Seitenkopf u. globaler Sperrtabelle
  - bei jeder Änderung: Erhöhung der Nummer in Seite und Ablage des aktuellen Wertes in der globalen Sperrtabelle
  - vor globaler Sperranforderung wird überprüft, ob Seite im lokalen Puffer vorliegt und mit welcher Versionsnummer
  - Invalidierung wird bei Sperrbearbeitung über Versionsnummern festgestellt
  - Alternative zu Versionsnummern: Invalidierungsvektoren (siehe MRDBS-Buch)
- Globale Sperrtabelle führt auch Page-Owner (bei dynamischer Page-Owner-Zuordnung)
- weitgehende Einsparung von Nachrichten zur Kohärenzkontrolle
  - Anpassung der erweiterten Sperrinfo bei Freigabe von Schreibsperren (Unlock), also ohne eigene Nachrichten
  - Erkennung von Pufferinvalidierungen auf Seitenebene bei Lock-Request („on request“), d.h. ohne zusätzliche Nachrichten
  - Bestimmung des Page-Owners bei Lock-Request
  - Nachrichten für Page-Requests, jedoch i.a. schneller als I/O

On-Request-Invalidierung: Beispiel

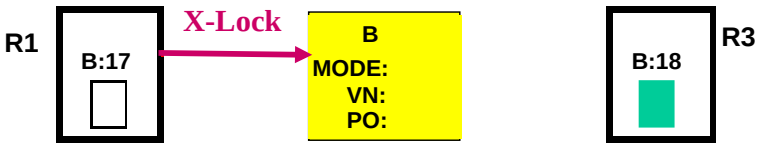
1.) erste Änderung von Seite B in R3



2.) Situation nach Freigabe der Schreibsperre durch R3



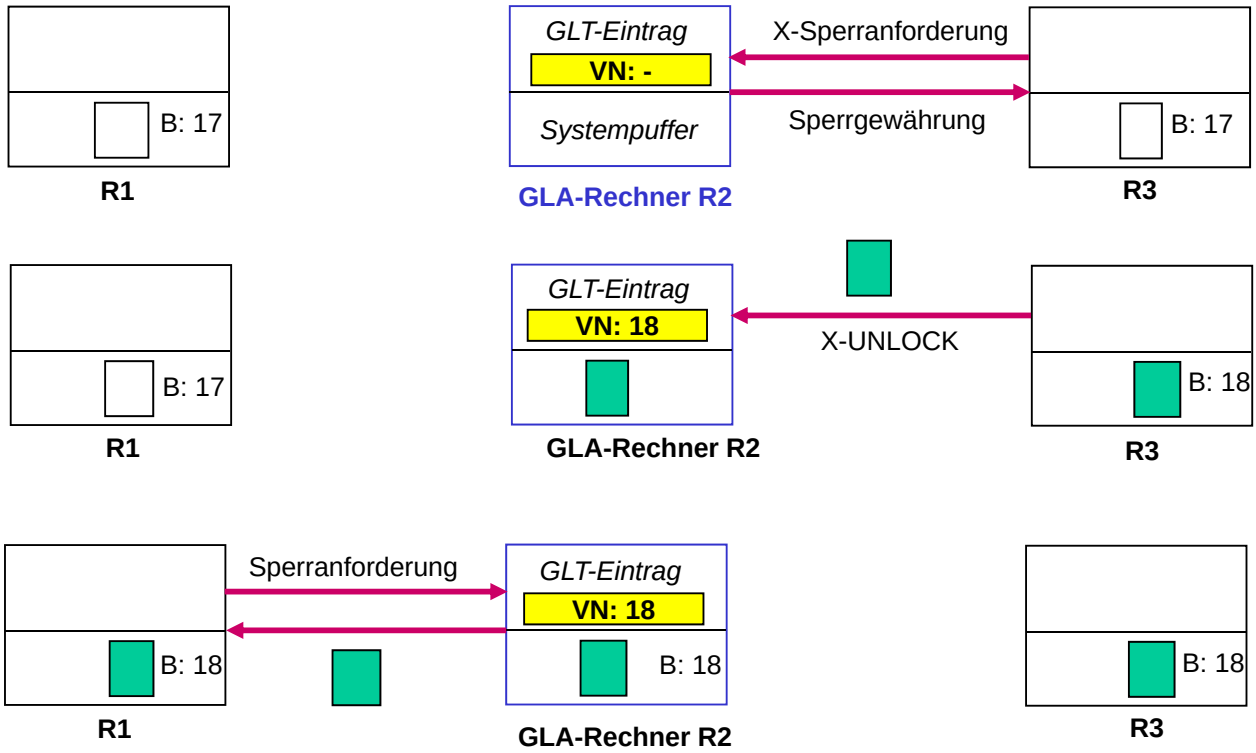
3.) Bei Sperranforderung durch R1 wird Pufferinvalidierung erkannt



Feste Page-Owner-Zuordnung

- zu jeder Seite (Partition) existiert ein Rechner, dem vorab die „Page Ownership" zugeordnet wurde
  - alle Änderungen sind zum Page-Owner zu transferieren
- gemeinsame Zuordnung von GLA und Page-Ownerships ermöglicht starke Kommunikationseinsparungen
  - GLA und Page-Ownership zu einem Objekt am gleichen Rechner
  - Transfer geänderter Seiten bei EOT zusammen mit der X-Sperrfreigabe
  - Kombination der Page-Requests mit Lock-Requests
  - oft Seitenbereitstellung mit Sperrgewährung
  - Kohärenzkontrolle (Erkennung von Invalidierungen und Update-Propagierung) ohne zusätzliche Nachrichten
- Besonders interessant für Primary-Copy-Sperrverfahren
  - Globale Synchronisation u. Seitenübertragungen auf alle Rechner verteilt
  - lokale GLA erspart Page-Transfers zum/vom Owner
  - Koordinierung von GLA- und Lastzuordnung reduziert #Seitenübertragungen

Kombinierte GLA- und Page-Owner-Zuordnung („Primary-Copy-Ansatz“)



Vermeidung von Pufferinvalidierungen

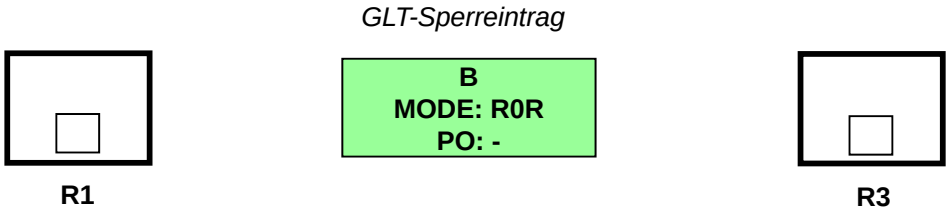
- Invalidierung nur solcher Seiten möglich, die nicht durch Sperre geschützt sind => Invalidierungen werden vermieden, falls Seiten vor Sperrfreigabe aus dem Puffer eliminiert werden
- ***Buffer Purge*** vor Sperrfreigabe bei EOT nicht akzeptabel
  - impliziert FORCE; keine Nutzung von Inter-Transaktions-Lokalität
- ***Haltesperren***: Seiten im Puffer werden durch spezielle, rechnerbezogene Haltesperren vor einer Invalidierung geschützt
  - jede Seite im Puffer ist durch reguläre (Transaktions-) Sperre oder durch Haltesperre geschützt
  - Haltesperren sind nicht kompatibel mit externen Schreibanforderungen => bevor externe Änderung (Invalidierung) erfolgt, muss Haltesperre freigegeben (entzogen) werden
  - Seite wird vor Aufgabe der Haltesperre aus dem Puffer entfernt

Haltesperren und Autorisierungen

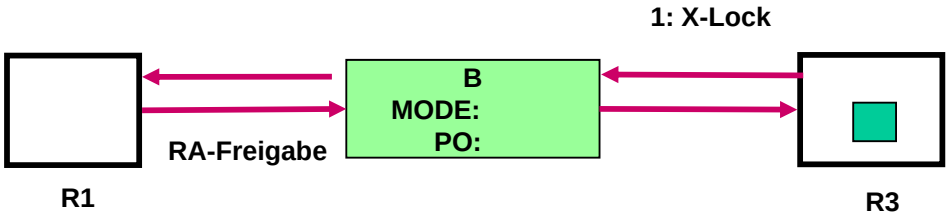
- Lese- und Schreibautorisierungen (auf Seitenebene) können zur Realisierung des Haltesperrenkonzeptes verwendet werden!
  - *Nutzung zur lokalen Sperrbearbeitung + Vermeidung von Pufferinvalidierungen*
- zwei Typen von Haltesperren: RA und WA
  - WA-Haltesperre nur in einem Rechner möglich: erlaubt lokale Synchronisation aller Zugriffe
  - RA-Haltesperren gleichzeitig in mehreren Rechnern möglich: lokale Synchronisation von Lesesperren
- Keine Versionsnummern etc. wir bei Erkennungsansätzen erforderlich

Beispiel Haltesperren

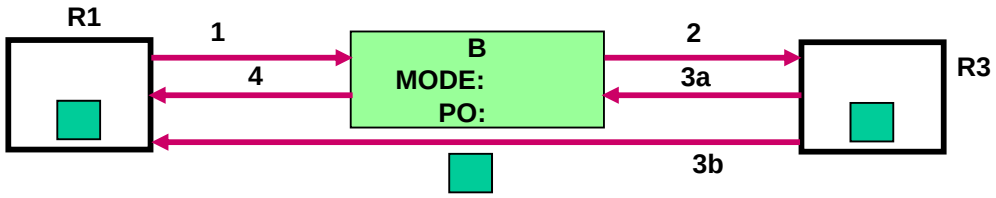
1.) Ausgangslage



2.) Änderung in R3

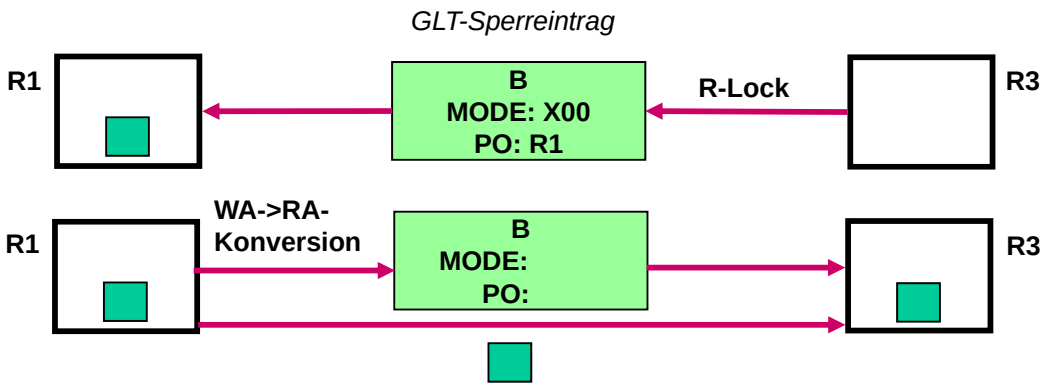


3.) X-Request (1) durch R1

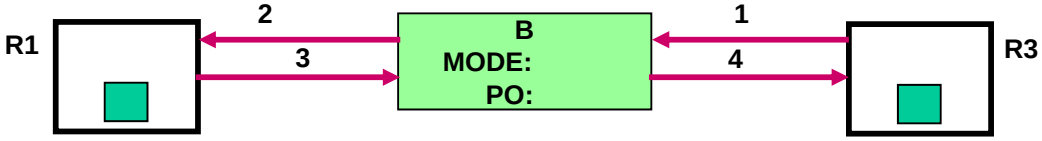


Beispiel Haltesperren (2)

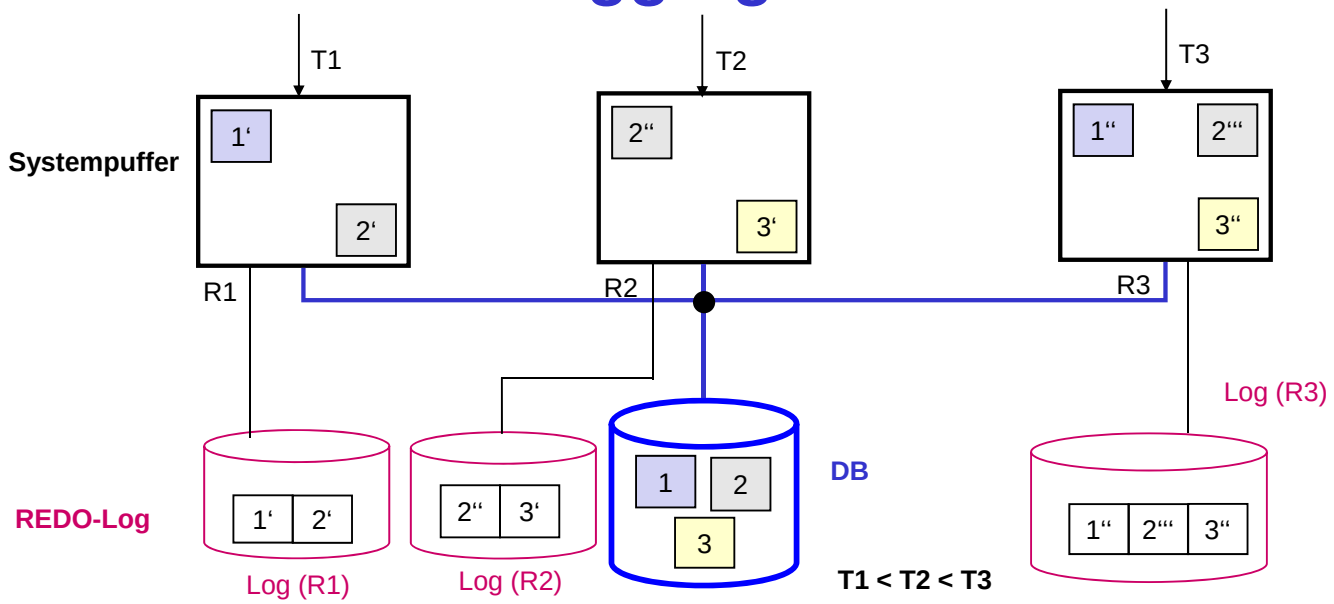
Leseanforderung in R3



X-Request (1) durch R3 verursacht RA- und PO-Entzug an R1 (2)

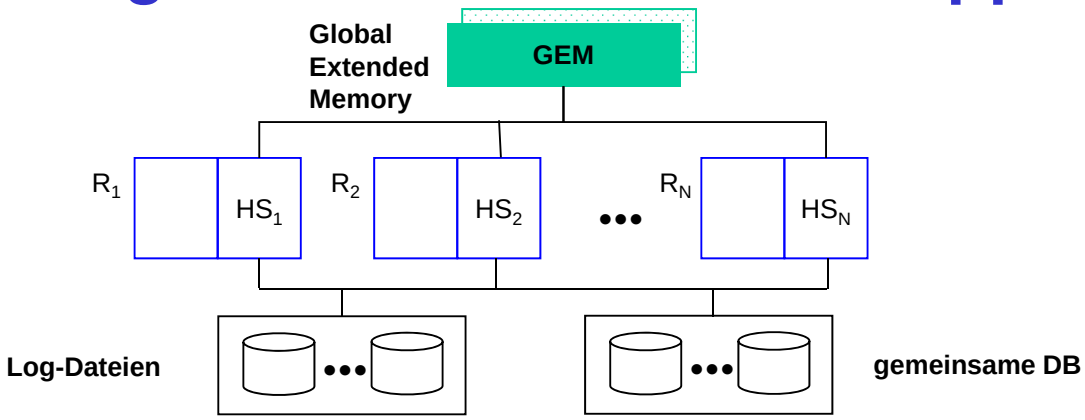


Logging



- jeder Rechner führt lokale Log-Datei, in der Änderungen von Transaktionen des Rechners protokolliert sind
- globaler Log (Mischen der lokalen Log-Dateien) wegen Gerätefehler
- globaler Log ggf. bereits für Crash-Recovery erforderlich (wenn geänderte Seiten direkt zwischen den Rechnern ausgetauscht werden)

Nutzung einer nahen Rechnerkopplung

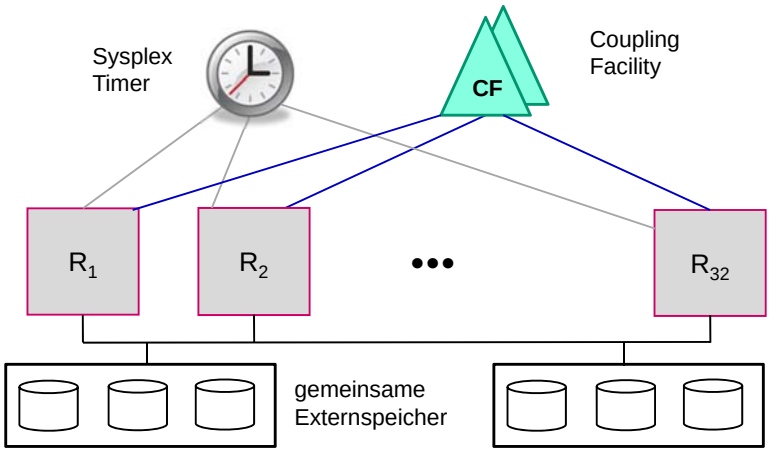


- Gemeinsame Halbleiterspeicherbereiche:
  - beschleunigte Kommunikation (Austausch geänderter Seiten)
  - Globale Datenstrukturen zur Synchronisation (Sperrtabelle) und Lastverteilung
  - Globaler Systempuffer und Allokation von DB-Dateien
  - Logging (lokale und globale Log-Dateien)
- Speichereigenschaften:
  - Nicht-Flüchtigkeit; schneller, synchroner Zugriff (< 10 Mikrosek.)
  - Seiten- und Eintragszugriffe
  - spezielle Synchronisationsinstruktionen (Compare & Swap)

IBM Parallel Sysplex

- Shared-Disk-Architektur für zOS-Umgebungen (DB2, IMS, Adabas ...)
- nahe Kopplung von IBM-Großrechnern über Coupling Facility (CF) und Sysplex Timer
- CF: Spezialprozessor für globale Kontrollaufgaben:
  - Sperrbereich (lock structure)
  - Pufferbereich (cache structure)
  - Listenbereich (list structure)
- Dynamische Lastbalancierung

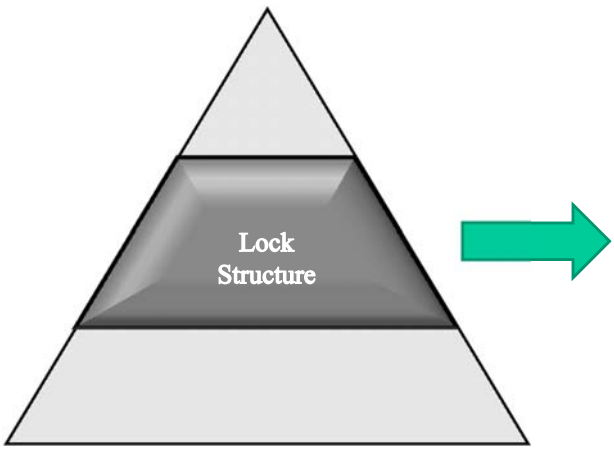
„Global Workload Manager“)



Quelle: IBM

Sysplex Global Locking

- Globales Sperrinteresse auf Rechnerebene wird in CF verwaltet
  - reduzierte Sperrinfo auf Ebene von Hash-Klassen
  - entspricht grob Schreib- bzw. Leseautorisierungen
  - Transaktionssperren / Wartebeziehungen bei LLMs (.IRLM =Internal Resource Lock Manager) der Knoten
- Kommunikation zwischen Rechnern nur für (seltene) Konfliktfälle



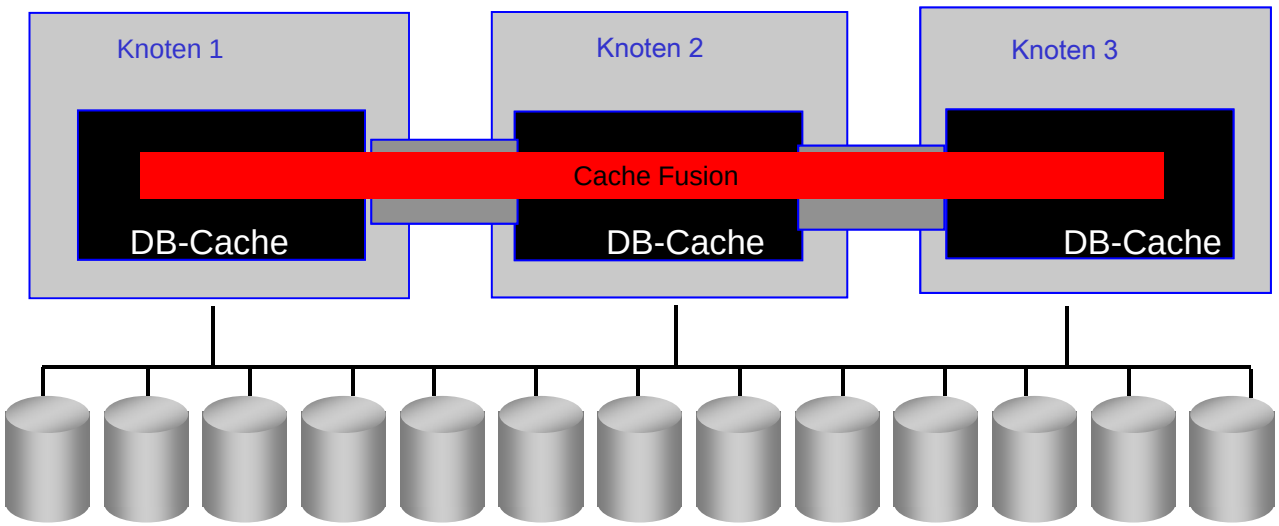
Lock Table (Hash Table)

Exclusive	Shared Lock Status						
00	1	1	0	0	0	0	0
02	1	0	0	0	0	0	0
…							
…							
01	0	0	0	0	0	0	0

Sysplex Kohärenzkontrolle

- FORCE in globalen Puffer der CF (ca 100 µs pro Seite)
- Multicast-Invalidierung
  - CF kennt Pufferstatus jedes Knotens
  - für in CF geschriebene geänderte Seite werden hardware-gestützt Rechner mit betreffender Seite über Invalidierung informiert
  - Bitvektor pro Rechner zeigt pro Pufferrahmen an, ob Invalidierung vorliegt
- Lesen der aktuellen Seiten bei Invalidierung von CF-Puffer bzw. Externspeicher
- asynchrones „Cast-Out“ geänderter Seiten vom CF-Puffer in Externspeicher

Oracle Real Application Clusters

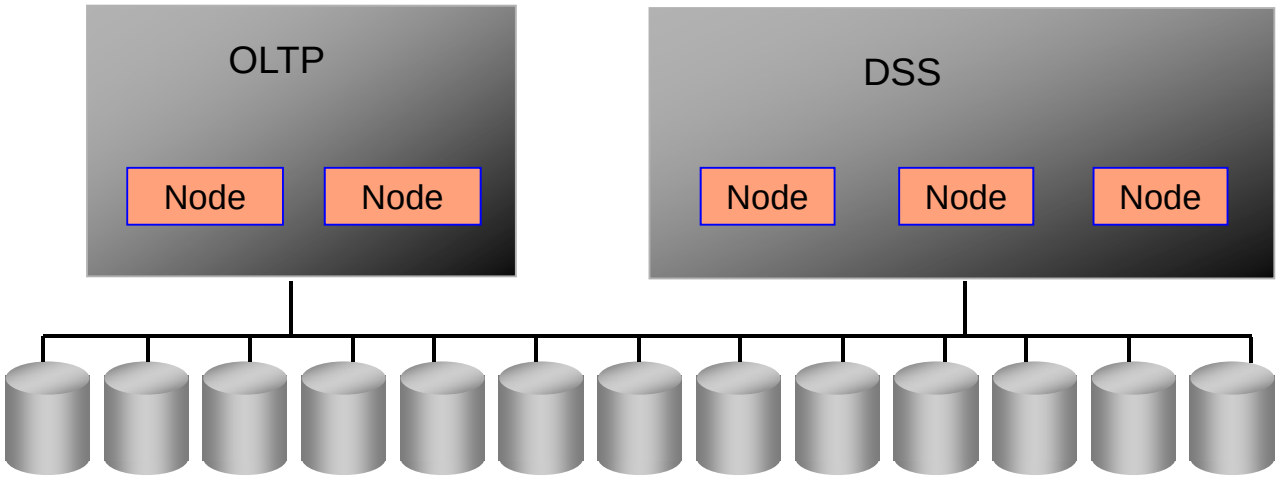


Quelle: Oracle

- Verteiltes Sperrprotokoll (Distributed Lock Manager) auf dedizierten Rechnern
  - Sperrzuständigkeit wird über Hash-Funktion festgelegt
- Kohärenzkontrolle über Haltesperren
  - Noforce-Ansatz ("fast commit")
  - Austausch geänderter Seiten ursprünglich über Externspeicher, jetzt direkter Transfer („cache fusion“)

Oracle Real Application Clusters

Kombinierter Einsatz für OLTP + DSS-Queries



Quelle: Oracle

Zusammenfassung

- Optimierungsziele
  - minimale Zahl von Nachrichten für Synchronisation u. Kohärenzkontrolle
  - koordinierte Lösung von Synchronisation und Kohärenzkontrolle
- Nutzung von Lokalität zur Einsparung externer Sperranforderungen
  - Nutzung einer lokalen GLA
  - Schreibautorisierungen und Leseautorisierungen
  - hierarchischer Sperransatz
- Kohärenzkontrolle
  - On-Request-Invalidierung oder Vermeidungsansatz über Haltesperren
  - Update-Propagierung: NOFORCE + direkter Seitenaustausch zwischen Rechnern
  - affinitätsbasierte Lastverteilung (Lokalität) begrenzt Pufferinvalidierungen und Seitentransfers
- nahe Kopplung kann effektiv genutzt werden (-> IBM Sysplex)