

Verteilte Systeme

Vorlesung 8
Sebastian Iwanowski
FH Wedel

Verteilte Systeme

1. Innovative Beispiele aus der Praxis
2. Allgemeine Anforderungen und Techniken verteilter Systeme
3. Die Client-Server-Beziehung und daraus entstehende Fragestellungen
4. Nebenläufigkeitstechniken in Java
5. Entfernte Aufrufe
6. Objektmigration
7. Agententechnologie
8. Dienstevermittlung
- 9. Synchronisation von Daten**
10. Konzepte zur Erzielung von Fehlertoleranz
11. Web Services

Inhalt heute:

Überblick Synchronisation von Daten

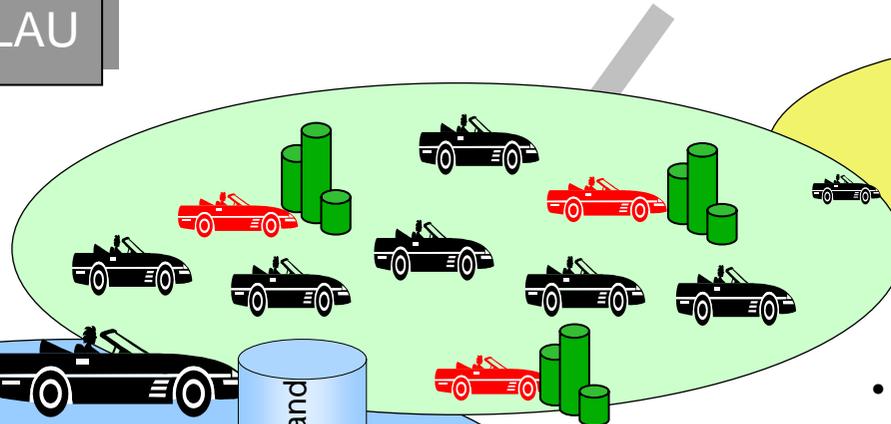
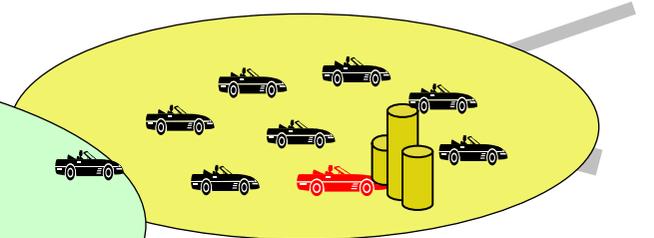
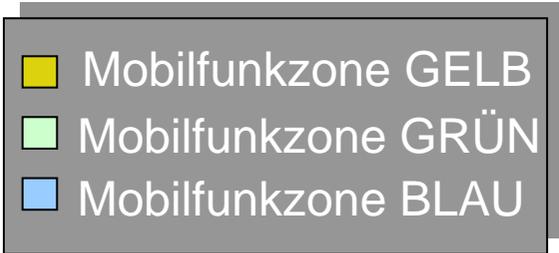
Datenzentrierte Synchronisation

Client-zentrierte Synchronisation

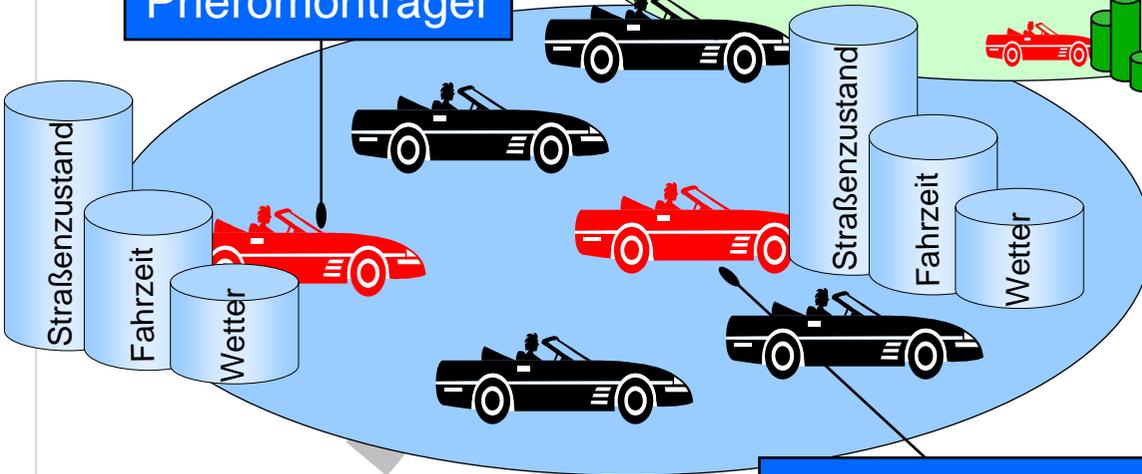
Überblick Synchronisation von Daten

Anwendungsgebiet

Bsp.: Car Swarm Intelligence



Pheromonträger



Pheromonträger

- Pheromonträger speichern alle Informationen ihrer Zone
- Jedes Fahrzeug meldet Infos an die Pheromonträger *seiner* Zone
- Jedes Fahrzeug kann Infos der Pheromonträger einer beliebigen Zone lesen
- Jedes Fahrzeug entscheidet autonom, wo es lang fährt

Welche Daten sind zu synchronisieren ?

Kernproblem: Synchronisation von Datenkopien

Gründe für Datenkopien:

- Zuverlässigkeit
- Leistung

Probleme bei der Synchronisation von Datenkopien

- Unsichere Kommunikationskanäle zwischen den Kopien
- Kopien häufig nicht vollständig gegenseitig bekannt
- Mobilität von Client oder Server
- Ort und Zeit der Aktualisierungen nicht vorhersehbar

Weitere Anwendungen

Primäres Anwendungsgebiet für Datenkopien: Internet

- **Webseiten**
- **e-mails**
- **Web Services**

Synchronisation = Konsistenzerhalt

Typen von Konsistenzforderungen

- datenzentriert *global auf das Netz bezogen*
- Client-zentriert *individuell aus der Sicht eines Clients*

Typen von Konsistenzlösungen

Wer ?

- Server-getrieben (push)
- Client-getrieben (pull)

Wie ?

- eifrig (eager, greedy)
- träge (lazy)

Datenzentrierte Synchronisation

Überblick über datenzentrierte Synchronisation

Datenzentrierte Konsistenzforderungen



- **strenge Konsistenz**
- **sequentielle Konsistenz**
- **kausale Konsistenz**

- **schwache Konsistenz**

Strenge Konsistenz

Definition:

Jede Leseoperation gibt den zuletzt geschriebenen Wert zurück

Realisierung: **nicht möglich !**

Begründung:

wegen der beschränkten Datenübertragungsgeschwindigkeit

Sequentielle Konsistenz

Definition (Lampport, 1979):

Alle Schreiboperationen werden von jedem Client in der Reihenfolge gesehen, in der sie stattfanden.

Realisierung:

Diese Techniken sind auch für andere Konsistenzforderungen verwendbar !

- **Primärbasiertes Schreiben**

- 1) *Client kommuniziert nur mit seinem Lieblingsserver*
- 2) *Lieblingsserver kommuniziert mit Primärserver*
- 3) *Primärserver kommuniziert mit allen Servern, die Datenkopien halten*

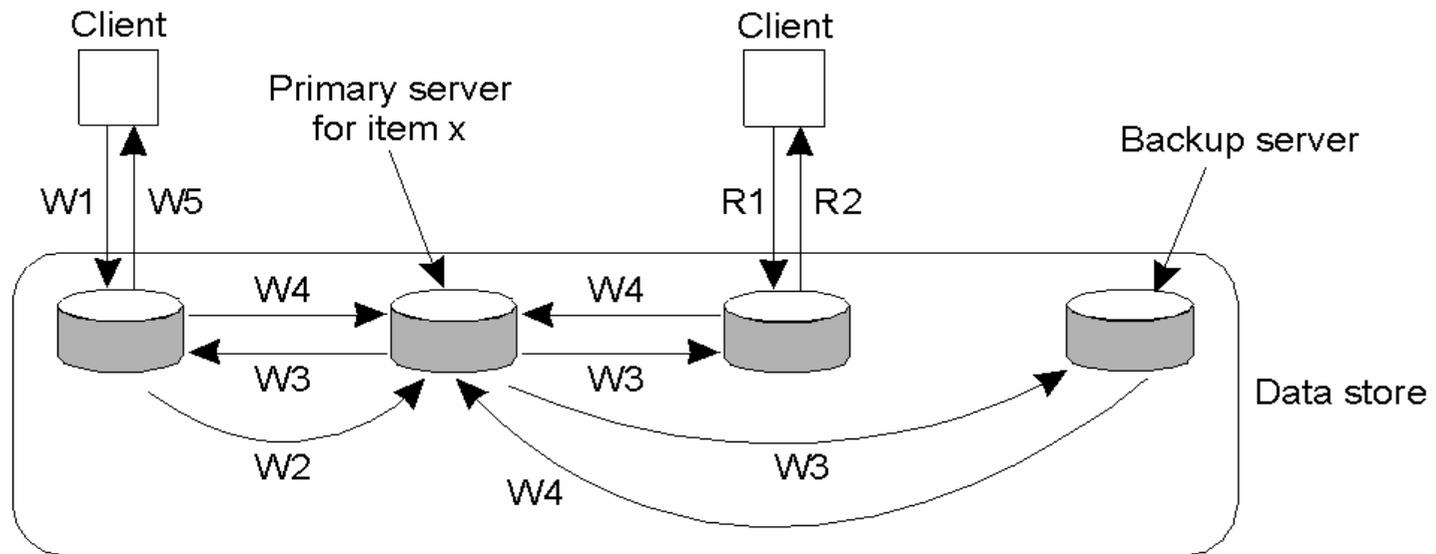
- **Mehrfaches Schreiben (replicated write)**

Client kommuniziert mit allen Servern, die Datenkopien halten

Primärbasiertes Schreiben

Entferntes Schreiben:

Lieblingsserver teilt Änderungen einem festen Primärserver mit



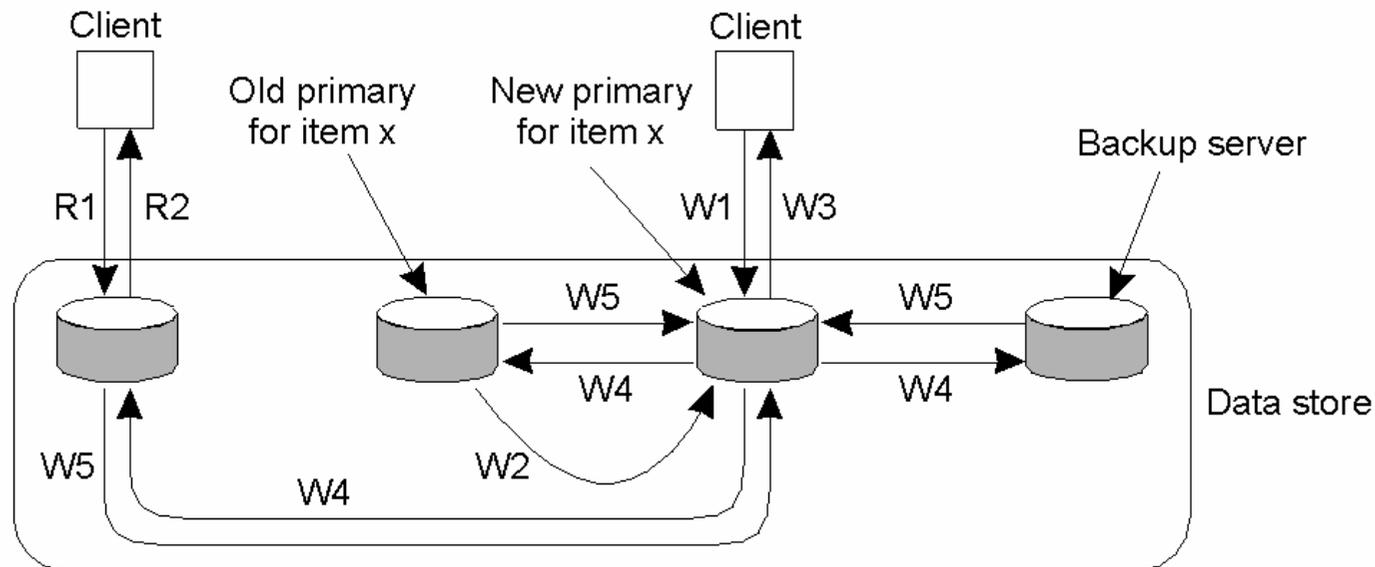
W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Primärbasiertes Schreiben

Lokales Schreiben:

Lieblingsserver kann der Primärserver werden



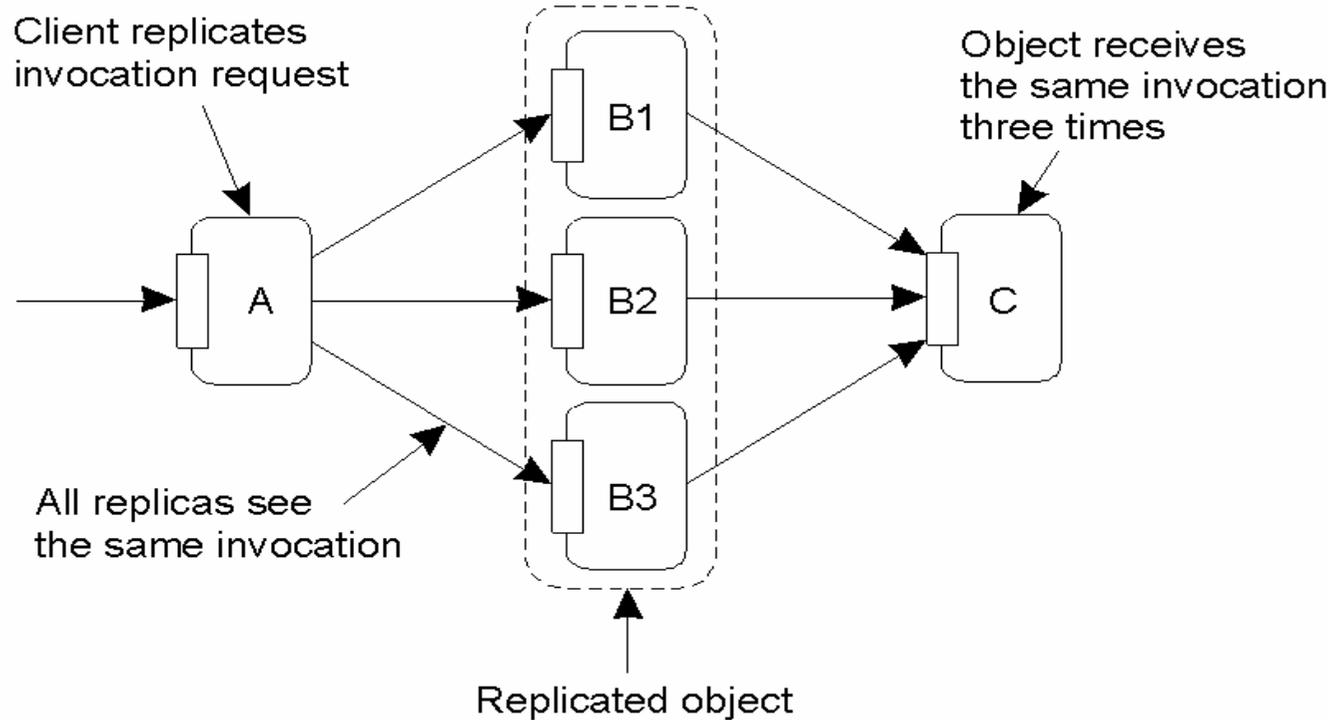
W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Mehrfaches Schreiben

Probleme durch mehrfaches Schreiben:

1) Aktualisierungen können neue Prozesse auslösen

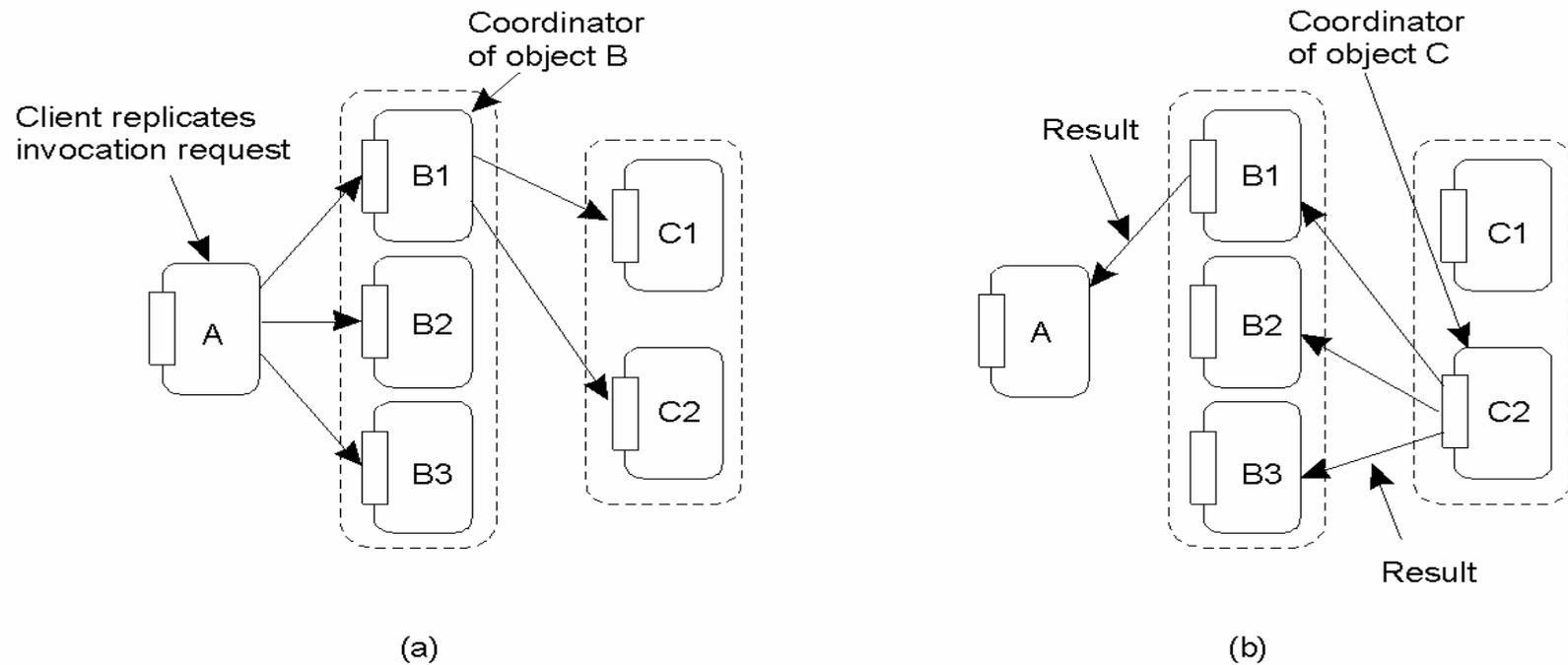


Mehrfaches Schreiben

Probleme durch mehrfaches Schreiben:

1) Aktualisierungen können neue Prozesse auslösen

Lösung: Bestimmung von Koordinatoren



Mehrfaches Schreiben

Probleme durch mehrfaches Schreiben:

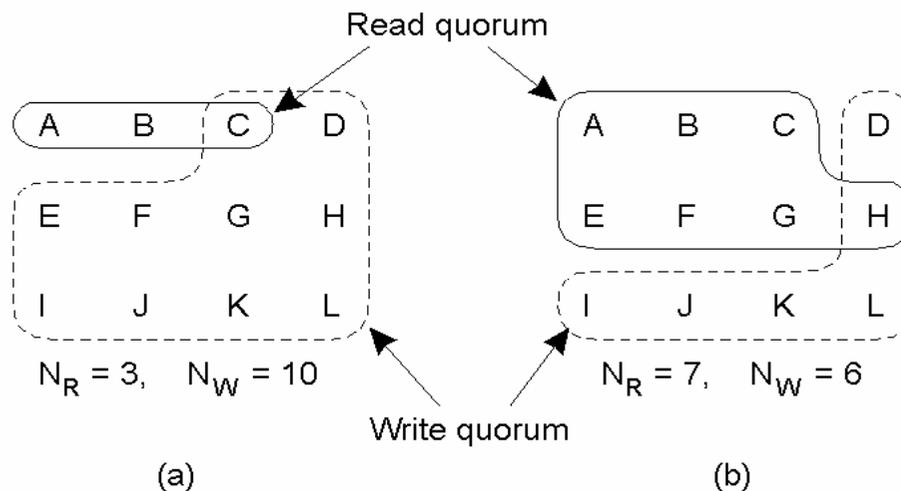
2) Mehrere Clients wollen dasselbe aktualisieren

Lösung: Quorum-basierter Zugriff

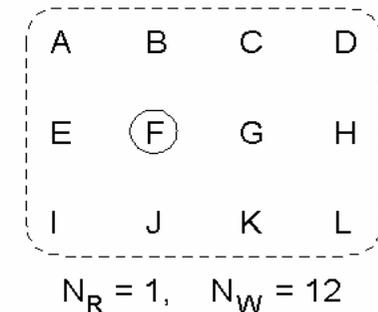
N_R : Lesequorum N_W : Schreibquorum

Konsistenzforderungen: i) $N_R + N_W > N$

ii) $N_W > N / 2$



Spezialfall ROWA:



Weichere Konsistenzforderungen

Kausale Konsistenz:

Nur die Schreiboperationen, die kausal voneinander abhängig sind, müssen von jedem Client in der Reihenfolge gesehen werden, in der sie statt fanden.

Spezialfall FIFO:

*Nur die Schreiboperationen, die kausal voneinander abhängig sind **und die vom selben Client** ausgeführt werden, müssen von jedem Client in der Reihenfolge gesehen werden, in der sie statt fanden.*

Weichere Konsistenzforderungen

Schwache Konsistenz:

Synchronisierungsvariable wacht über Bündel von Operationen. Nur das gesamte Bündel wird von allen Clients an derselben Ausführungsposition gesehen (relativ zu den anderen Bündeln).

Lösung:

- Anforderungsoperation `acquire`
- Freigabeoperation `release`

eifrige vs. träge Synchronisierung

Client-zentrierte Synchronisation

Client-basierte Konsistenz

Anforderung:

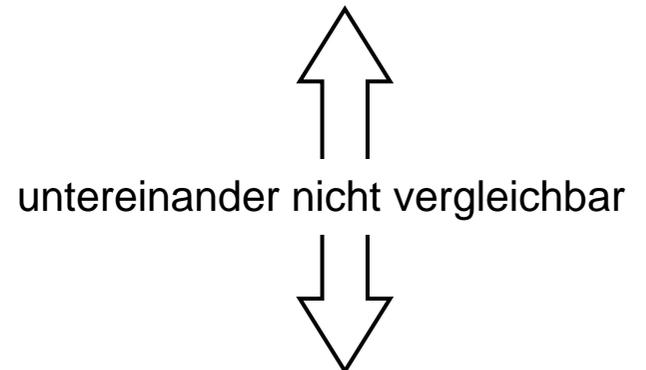
Client darf nur Daten sehen, die er selbst nicht als inkonsistent beweisen kann.

Erschwerendes Problem:

Clients arbeiten im Laufe der Zeit mit unterschiedlichen Kopien.

Konsistenzforderungen:

- monotonen Lesen
- monotonen Schreiben
- Read your Writes
- Write follows Read



Monotones Lesen

Definition:

Client bekommt beim wiederholten Lesen desselben Datensatzes mindestens genau so aktuelle Werte zurück wie beim letzten Mal

Lösung: Verwaltung von Mengen von Schreib-IDs

- **Lesemenge:** Menge aller Schreib-IDs, die für diesen Lesevorgang relevant sind
- **Schreibmenge:** Menge aller Schreib-IDs, die von diesem Client durchgeführt worden sind

Jeder Server, von dem gelesen werden soll, bekommt mit der Leseanforderung die aktuelle Lesemenge.

Vor Rückgabe des Wertes überprüft der Server eventuell erforderliche Aktualisierungen und führt diese durch.

Reicht es aus, wenn die Lesemenge eine Teilmenge der Schreibmenge ist ?

Weitere Konsistenzforderungen

Monotones Schreiben:

Client schließt jede Schreiboperation auf einem Datensatz ab, bevor er eine neue auf demselben Datensatz beginnt.

Read your Writes:

Jede Änderung, die durch denselben Client durchgeführt worden ist, wird in nachfolgenden Leseoperationen berücksichtigt.

Write follows Read:

Das Schreiben eines Datensatzes erfolgt nur auf Kopien, die mindestens so aktuell sind wie alle, die zuvor gelesen wurden.

Wer initiiert das Anlegen von Datenkopien ?

- automatisch: Permanente Kopien
- **Server ergreift Initiative: push-Caches**

aus Kapazitätsgründen

Heranschieben der Daten an häufig fragende Clients

Lösung: Regelungstechnischer Ansatz

gibt Schwellen vor für:

- *Löschen einer Datenkopie*
- *Erstellen einer Datenkopie*

- **Client ergreift Initiative: Caches**
zum Verbessern der Antwortzeiten

Was wird bei Aktualisierungen weitergegeben ?

- 1) Ein aktualisierter Datensatz**
- 2) Benachrichtigung über einer Aktualisierung**
- 3) Eine Aktualisierungsoperation, mit der man einen alten Datensatz auf den neuesten Stand bringen kann**

Wie wird eine Aktualisierungen weitergegeben ?

- **push**

 - für permanente Kopien*

 - für Server-initiierte Kopien*

 - für Client-initiierte Kopien bei Verwendung großer Caches durch mehrere Clients*

- **pull**

 - für Client-initiierte Kopien bei kleinen individuellen Caches*

- **Mischform lease:**

 - Client bitten Server um Versorgung mit Aktualisierungen (pull)*

 - Server verspricht und übernimmt Aktualisierungen für bestimmte Zeit (push)*

Epidemische Aktualisierungsprotokolle

Wie verteilt man Aktualisierungen schnell übers Netz ?

- **Infektiöse Server**

push: Server infiziert beliebigen anderen

- **Empfängliche Server**

pull: Server fragt beliebigen anderen

Epidemische Aktualisierungsprotokolle

- **Variante: Geschwätzige Server**

Server infiziert mit vorgegebener Wahrscheinlichkeit anderen Server

Wenn der andere Server schon infiziert war, nimmt die Wahrscheinlichkeit ab

- **Zum Löschen von Daten: *Sterbeurkunden***

***Beim nächsten Mal:
Konzepte zur Erzielung von Fehlertoleranz***