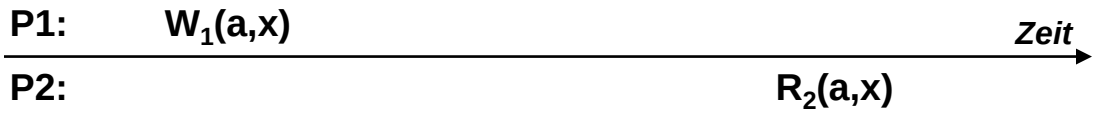
Korrektes Verhalten zweier Prozesse nach dem Modell der strikten Konsistenz:



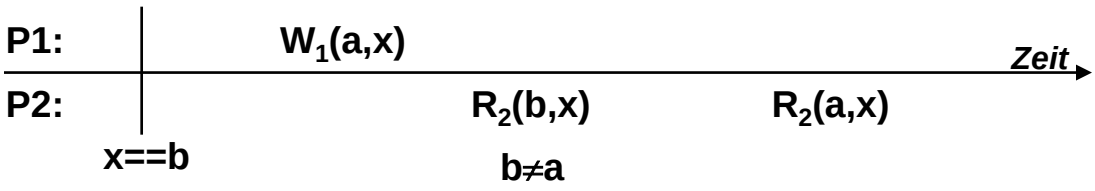
Wenn zwei Prozesse immer dieses Verhalten zeigen, verhält sich das verteilte System, in dem sie arbeiten, nach dem Modell der strikten Konsistenz korrekt.

Notation:

$W_1(a,x)$: ***Prozess P1 schreibt den Wert a in das von ihm benutzte Replikat von Datenelement x.***

$R_2(a,x)$: ***Prozess P2 liest den Wert a aus dem von ihm benutzten Replikat von Datenelement x.***

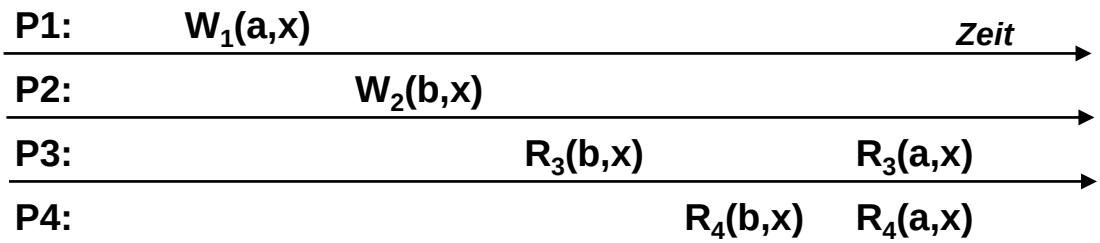
Nicht korrektes Verhalten zweier Prozesse nach dem Modell der strikten Konsistenz:



Wenn zwei Prozesse dieses Verhalten zeigen können, verhält sich das verteilte System, in dem sie arbeiten, nach dem Modell der strikten Konsistenz möglicherweise nicht immer korrekt. (Die Aktion $R_2(b,x)$ erfolgte möglicherweise sehr zeitnah nach der Aktion $W_1(a,x)$ an einem weit entfernten Ort des verteilten Systems.)

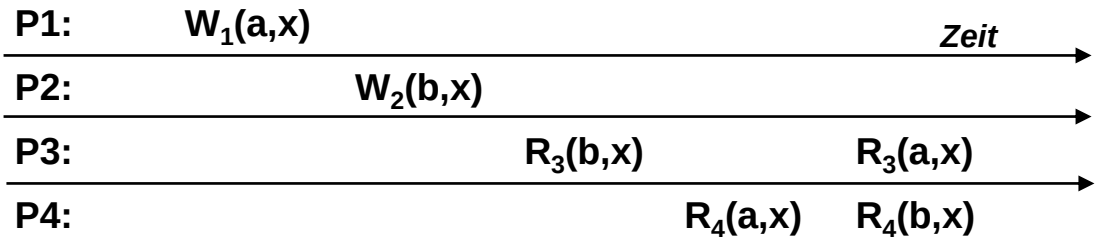
Strikte (strenge) Konsistenz

Korrektes Verhalten von Prozessen nach dem Modell der sequentiellen Konsistenz:



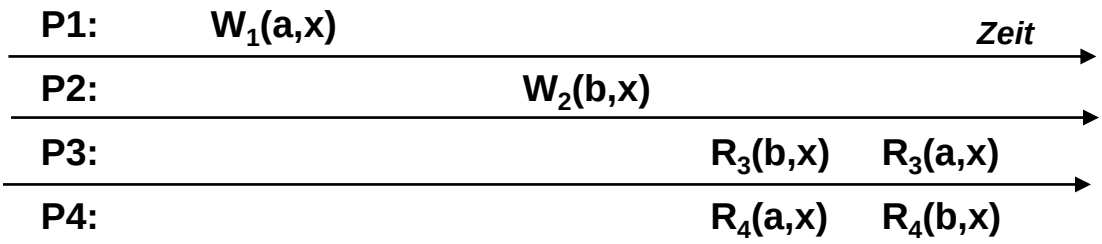
Die Prozesse P3 und P4 sehen die Ergebnisse der Schreibereignisse von P1 und P2 auf x in derselben Reihenfolge. Wenn zwei Prozesse in der Rolle von P3 und P4 immer dieses Verhalten zeigen, verhält sich das verteilte System, in dem sie arbeiten, nach dem Modell der sequentiellen Konsistenz korrekt.

Nicht korrektes Verhalten von Prozessen nach dem Modell der sequentiellen Konsistenz:



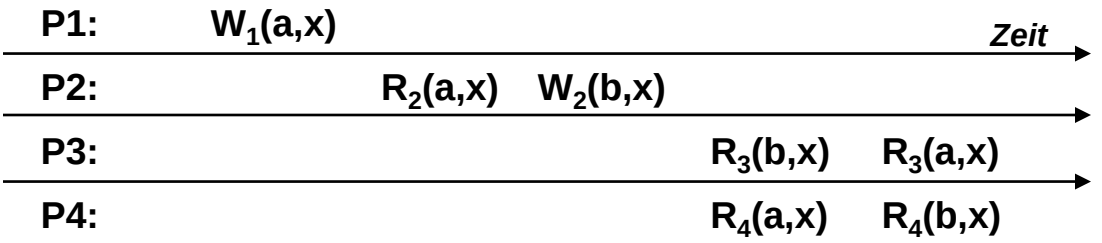
Die Prozesse P3 und P4 sehen die Ergebnisse der Schreibereignisse von P1 und P2 auf x in verschiedener Reihenfolge. Wenn zwei Prozesse in der Rolle von P3 und P4 dieses Verhalten zeigen können, verhält sich das verteilte System, in dem sie arbeiten, nach dem Modell der sequentiellen Konsistenz möglicherweise nicht immer korrekt.

Korrektes Verhalten von Prozessen nach dem Modell der kausalen Konsistenz:



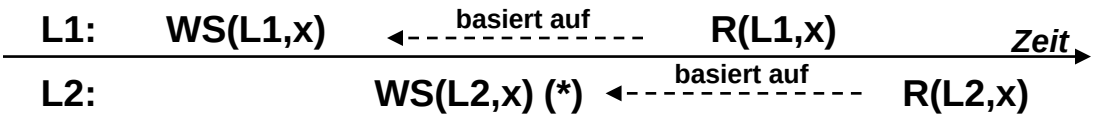
$W_1(a,x)$ und $W_2(b,x)$ sind nicht unbedingt potentiell kausal voneinander abhängig. Daher können die Prozesse P3 und P4 die Resultate dieser Ereignisse in beliebiger Reihenfolge sehen, und der gesamte Schedule ist nach dem Modell der kausalen Konsistenz korrekt.

Nicht korrektes Verhalten von Prozessen nach dem Modell der kausalen Konsistenz:



$W_1(a,x)$ und $W_2(b,x)$ sind potentiell kausal voneinander abhängig, da in die Berechnung von b möglicherweise das vorher gelesene a eingeht. Daher dürfen die Prozesse P3 und P4 die Resultate dieser Schreib-Ereignisse nicht in verschiedener Reihenfolge sehen, und der gesamte Schedule ist nach dem Modell der kausalen Konsistenz nicht korrekt.

Korrektes Verhalten eines verteilten Systems nach dem Modell der monotonen Lese-Konsistenz aus Sicht eines Benutzers, der auf die lokalen Kopien L1 und L2 des Datenelements x an ihren jeweiligen Lokalitäten zugreift:



(*) mit der Bedingung: WS(L2,x) ⊇ WS(L1,x)

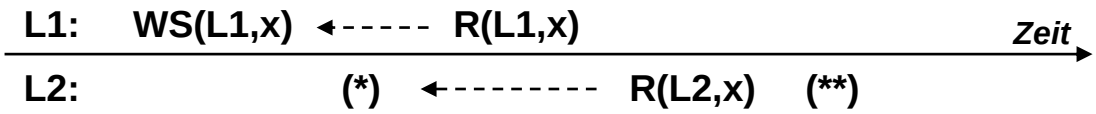
Das verteilte System verhält sich immer so, dass die Bedingung (*) gilt, bevor der Lesezugriff R(L2,x) erfolgt.

Notationen:

WS(Ln,x): ***Write-Set der lokalen Kopie n von x: Folge der bis zu diesem Zeitpunkt stattgefundenen Schreiboperationen auf die lokale Kopie Ln des Datenelements x.***

R(Ln,x): ***Leseoperation auf lokale Kopie Ln von x***

Nicht korrektes Verhalten eines verteilten Systems nach dem Modell der monotonen Lese-Konsistenz aus Sicht eines Benutzers, der auf die lokalen Kopien L1 und L2 des Datenelements x an ihren jeweiligen Lokalitäten zugreift:

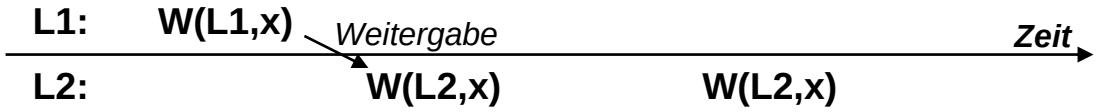


(*) WS(L2,x) *ohne* die Bedingung WS(L2,x) ⊇ WS(L1,x)
(**) WS(L2,x) *mit* der Bedingung: WS(L2,x) ⊇ WS(L1,x)

Das verteilte System kann sich so verhalten, dass (**) erst gilt, nachdem der Lesezugriff R(L2,x) erfolgt, aber nicht vorher.

Monotone Lese-Konsistenz

Korrektes Verhalten eines verteilten Systems nach dem Modell der monotonen Schreib-Konsistenz aus Sicht <u>eines</u> Benutzers, der schreibend auf die lokalen Kopien L1 und L2 des Datenelements x an ihren jeweiligen Lokalitäten zugreift:

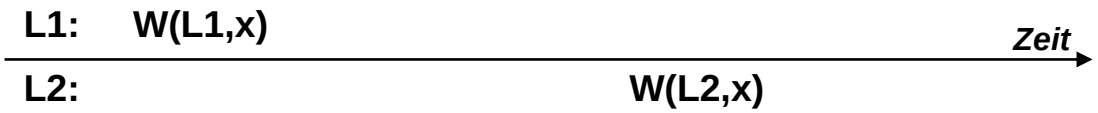


Das verteilte System verhält sich immer so, dass bevor der Benutzer eine Schreiboperation auf eine lokale Kopie durchführen kann, eine vorhergehende Schreiboperation auf eine andere Kopie an den Ort der zweiten Schreib-operation weitergegeben sein muss.

Notationen:

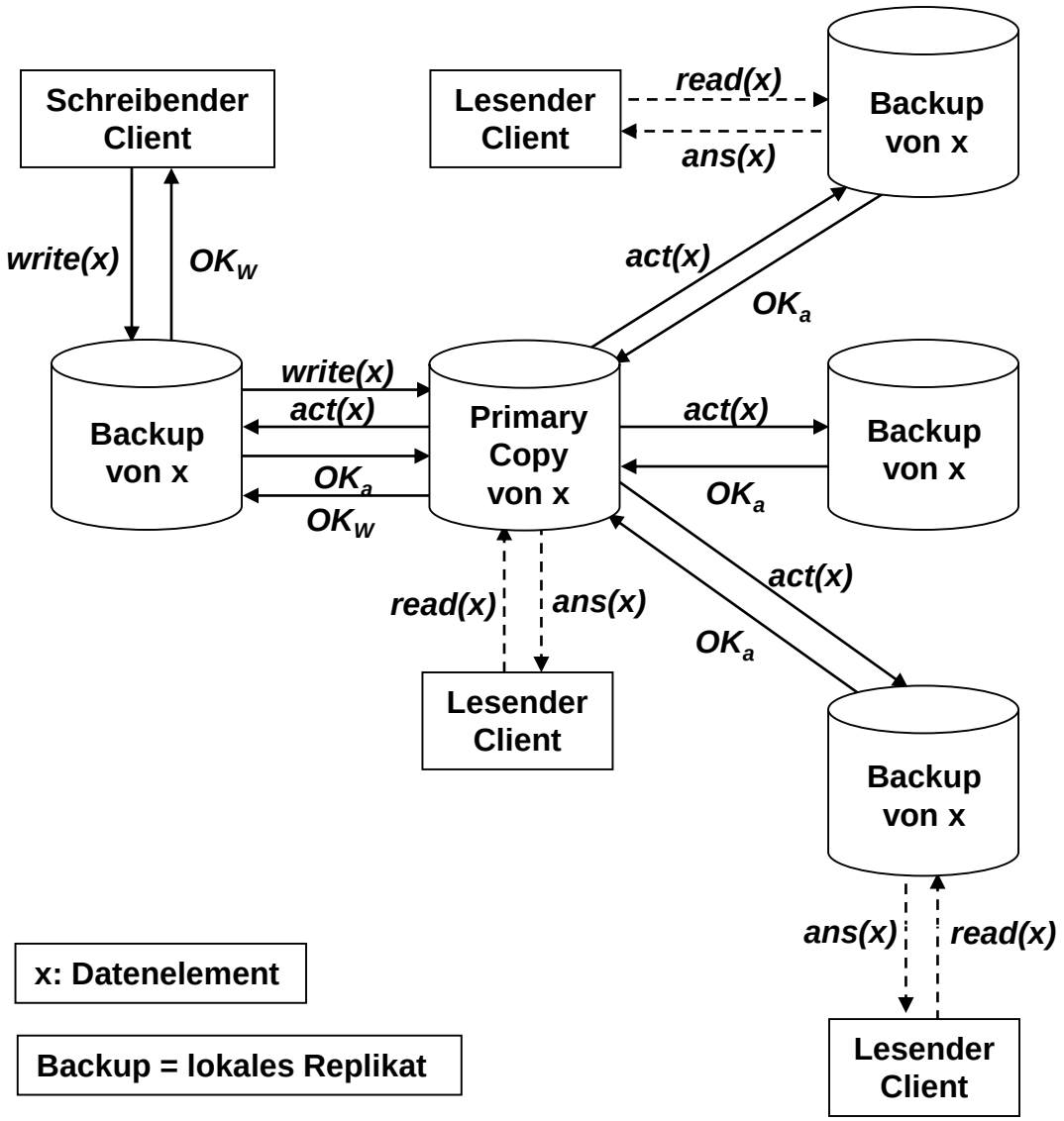
W(Ln,x): ***Schreiboperation auf lokale Kopie Ln von x***

Nicht korrektes Verhalten eines verteilten Systems nach dem Modell der monotonen Schreib-Konsistenz aus Sicht <u>eines</u> Benutzers, der schreibend auf die lokalen Kopien des Datenelements x an ihren jeweiligen Lokalitäten L1 und L2 zugreift:



Das verteilte System kann sich so verhalten, dass eine schreibende Operation auf eine Kopie x erfolgen kann, ohne dass vorhergehende Schreiboperationen auf x an einer anderen Kopie dort bekannt wären.

Monotone Schreib-Konsistenz



x: Datenelement

Backup = lokales Replikat

Primary Copy = primäres Replikat

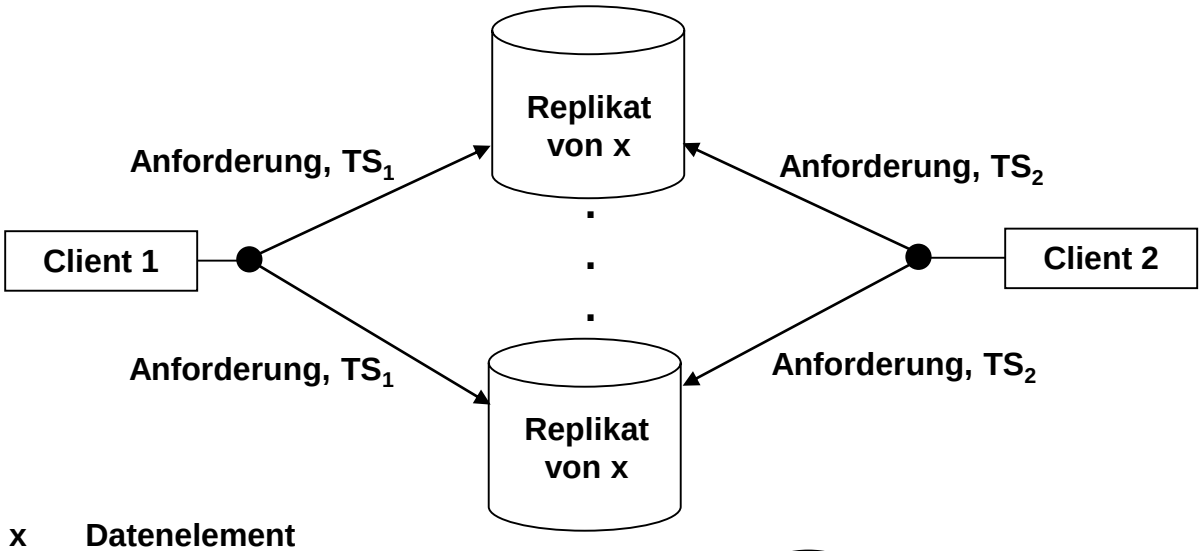
Schreibvorgang:

write(x):	Schreiboperation auf das Datenelement x
act(x):	Aktualisierung von Backup, veranlasst von Primary Copy
OK_a:	Bestätigung von act(x)
OK_W:	Bestätigung von write(x) nach Aktualisierung aller Backups

Lesevorgang:

read(x):	Leseoperation auf x
ans(x):	Antwort auf read(x) mit Wert von x

Primär-Backup-Protokoll



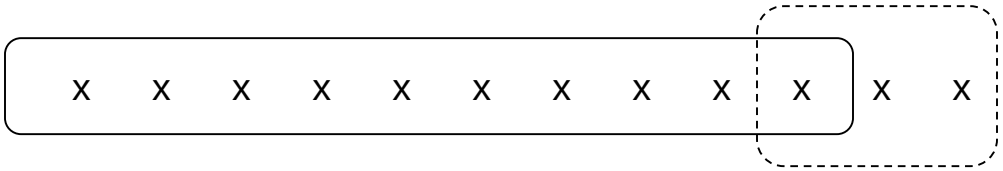
x Datenelement
● Frontend: Multicast
TS_i: Zeitstempel des Clients i, durch Lamport-Algorithmus geordnet.

Server mit Replikat von x + Replikationsmanager

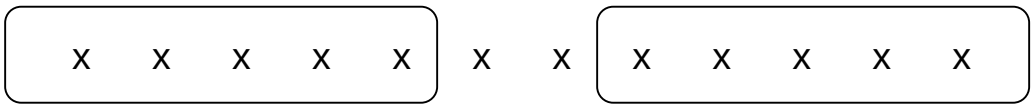
Request	**Ein Client schickt eine Anforderung an sein Frontend zur Weiterleitung an alle Replikate bzw. deren Replikate-Manager, entweder dadurch, dass er die Anforderung an ein primäres Replikat schickt, das dann die Anforderung an alle anderen Replikate weiterleitet, oder durch ein Multicast. Clients bzw. deren Frontends versehen Anforderungen mit einem Zeitstempel, z.B. einem Lamport-Zeitstempel, den sie untereinander konstruiert haben.**
Coordination	**Die Replikations-Manager einigen sich über die Reihenfolge der Ausführung der von verschiedenen Clients stammenden Anforderungen. D.h. sie einigen sich z.B. über die nebenläufige Ausführung einer bestimmten Anforderung auf ihren jeweiligen Replikaten, und danach erst kommen andere Anforderungen zur Ausführung.**
Execution	**Die Anforderung wird auf allen Replikaten ausgeführt.**
Consensus	**Die Replikate-Manager einigen sich auf das Resultat der Ausführung über ein Consensus Protokoll.**
Response	**Ein oder mehrere Replikate-Manager schicken eine Antwort zurück an das Frontend, dieses leitet die Antwort an den Client weiter.**

Aktive Replikation

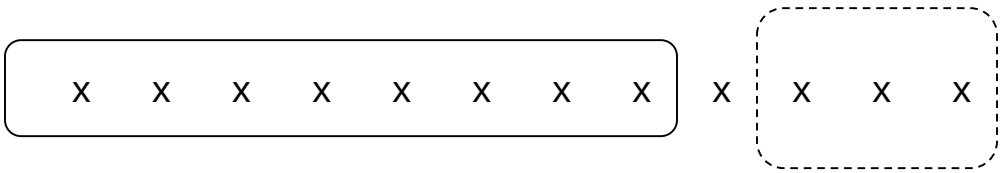
N = Anzahl der Replikate eines Datenelements x im verteilten System



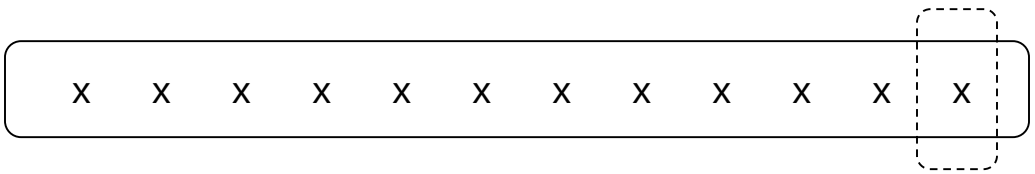
N=12 N_W=10 N_R=3 → keine Konflikte möglich



N=12 N_W=5 N_R=8 → Schreib/Schreib-Konflikte möglich



N=12 N_W=8 N_R=3 → Schreib/Lese-Konflikte möglich



N=12 N_W=12 N_R=1 → ROWA (read-one, write-all)

→ keine Konflikte möglich

Quorum-basierte Replikation