PD Dr. rer. nat. habil. Sven Groppe

Übungen zur Vorlesung

Mobile und Verteilte Datenbanken

WS 2012/2013

Blatt 6

Lösung

Aufgabe 1: Nicht-blockierendes 2PC durch Prozeß-Paare

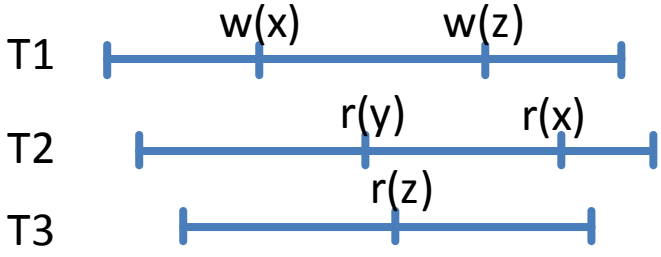
Blockierungen im 2PC-Protokoll können verhindert werden, wenn der Koordinator als Prozeß-Paar fehlertolerant realisiert wird. Dabei sendet der Koordinator-TM sämtliche für die Commit-Bearbeitung relevanten Zwischenzustände an einen Backup-TM in einem anderen Rechner. Nach Ausfall des Koordinators übernimmt der Backup-TM dessen Rolle. Wie hoch ist der Overhead einer solchen Lösung gegenüber einem 3PC-Protokoll?

Lösung:

Für ein zentralisiertes 2PC sind drei Nachrichten vom Koordinator zum Backup-TM zu schicken: vor Absenden der PREPARE-Nachrichten, vor Schreiben des Commit-Ergebnisses und vor Schreiben des Ende-Satzes. Der Erhalt dieser Nachrichten ist vom Backup-TM jeweils zu quittieren, so daß insgesamt 6 zusätzliche Nachrichten anfallen. Dies verursacht für N<3 einen höheren Aufwand als 3PC, für N>3 ist der Aufwand geringer. Da der Koordinator jeweils synchron auf die Quittung warten muß, ergibt sich jedoch eine signifikante Antwortzeiterhöhung, vor allem wenn die Kommunikation mit dem Backup über ein Weitverkehrsnetz erfolgen muß.

Aufgabe 2: Vergleich von Synchronisationsverfahren

An einem Rechner sei folgender Schedule von drei Transaktionen mit Zugriff auf ausschließlich lokale Objekte gegeben, wenn keine Synchronisation erfolgt:



Bestimmen Sie für jedes der folgenden Synchronisationsverfahren die vorkommenden Blockierungen und Rücksetzungen sowie die sich ergebende Serialisierungsreihenfolge:

a. Zeitmarkenverfahren (Basic Timestamp Ordering)
b. rückwärtsorientierte, optimistische Synchronisation, bei der die Validierung ausschließlich gegenüber bereits beendeten Transaktionen erfolgt

c. vorwärtsorientierte, optimistische Synchronisation, bei der die Validierung gegenüber laufende Transaktionen erfolgt
d. RX-Sperrverfahren mit Wait/Die-Deadlock-Vermeidung
e. RX-Sperrverfahren mit Wound/Wait-Deadlock-Vermeidung
f. RX-Sperrverfahren mit Deadlock-Erkennung
g. RX-Sperrverfahren mit Mehrversionen-Synchronisation und Deadlock-Erkennung.

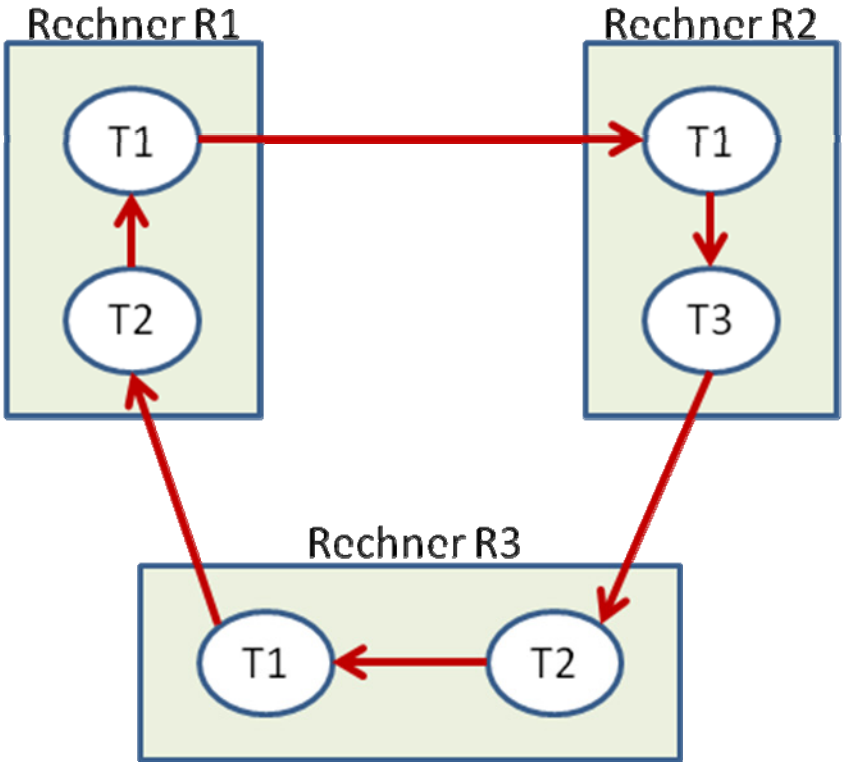
Lösung:

Vorwärts orientierte Validierungsverfahren: Write-Set der zu validierenden Transaktion wird mit Read-Sets aller laufenden Transaktionen verglichen
Rückwärts orientiertes Validierungsverfahren: Read-Set wird verglichen mit Write-Sets aller Transaktionen, die während der Laufzeit dieser Transaktion validiert wurden. Problem: langlaufende Transaktionen

a) Rücksetzung von T1;	T2 < T3
b) Rücksetzung von T2;	T3 < T1
c) Rücksetzung von T1 oder T2;	T3 < T2 oder T3 < T1
d) Wait (T1), Die (T2);	T3 < T1
e) Wound (T3), Wait (T2);	T1 < T2
f) Wait (T1), Wait (T2);	T3 < T1 < T2
g) keine Blockierung/Rücksetzung;	T2 < T3 < T1

Aufgabe 3: Deadlock-Erkennung

Wenden Sie den Algorithmus von Obermarck (wie er in R* Anwendung findet) zur Erkennung des folgenden Deadlocks an. Geben Sie jeden Zwischenschritt an (Annahme: *ts (T1) < ts (T2) < ts (T3)*). Wieviele Nachrichten werden zur Auflösung des Deadlocks benötigt?



Lösung:

Lokale Wartegraphen (EXT = EXTERNAL):

R1: EXT -> T2 -> T1 -> EXT
R2: EXT -> T1 -> T3 -> EXT
R3: EXT -> T3 -> T2 -> EXT

R2 sendet keine Nachricht, da *ts (T1) < ts (T3)*

Nachricht von R1 nach R2: EXT -> T2 -> T1 -> EXT

Vervollständigung in R2 zu EXT -> T2 -> T1 -> T3 -> EXT;

keine Folgeaktion, da *ts (T2) < ts (T3)*

Nachricht von R3 nach R1: EXT -> T3 -> T2 -> EXT

Vervollständigung in R1 zu EXT -> T3 -> T2 -> T1 -> EXT

Nachricht von R1 an R2,

Vervollständigung ergibt T3 -> T2 -> T1 -> T3

Erkennung des Deadlocks in R2 (=> Rücksetzung der lokalen Transaktion T2)

Deadlock wurde mit 3 Nachrichten aufgelöst
(2 Nachrichten, falls erste Nachricht von R1 nach R2 nicht gesendet wird)

Beispiellösungen aus:

Rahm, E.: Mehrrechner-Datenbanksysteme, Addison-Wesley 1994.