ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. G. Alonso *C. Hagen, B. Kemme*

AR 1: Atomarität in verteilten Systemen

Theoretische Übung 4

Beispiellösung

Aufgabe 1: 2-Phase Commit Protokolle

Im 2-Phase Commit Protokoll (2PC) entscheidet der Koordinator *zuerst* Commit, indem er einen **commit**-Record in seinen Log schreibt, und sendet *dann* COMMIT-Nachrichten zu den Teilnehmern. Nehmen Sie nun an, dass die Reihenfolge dieser zwei Schritte umgedreht ist. Zeigen Sie, dass ein derartig modifiziertes 2PC nicht blockiert, wenn

a) es keine Kommunikationsfehler gibt und höchtens ein Prozess ausfallen kann, oder

b) es keine Kommunikationsfehler gibt und Prozesse die Möglichkeit haben, die Nachrichten atomar an alle Teilnehmer zu senden (*atomic broadcast*), d.h. dass entweder alle Teilnehmer die Nachricht erhalten oder kein Teilnehmer die Nachricht erhält.

Wie beeinflusst diese Modifikation des 2PC die Recovery? Geben Sie die notwendigen Schritte für die Recovery an.

1. *Ein Teilnehmer P eines 2-Phase-Commit ist blockiert, wenn die folgenden Bedingungen erfüllt sind:*
   - *P ist in einem unsicheren Zustand*
   - *Der Koordinator ist nicht erreichbar*
   - *Alle anderen Teilnehmer, die erreichbar sind, sind ebenfalls in einem unsicheren Zustand*
2. *Wir betrachten nun das modifizierte 2-PC-Protokoll, indem der Koordinator zuerst die Commit-Nachricht sendet und dann die Commit-Entscheidung fällt (in dem er den Commit-Logeintrag schreibt).*
3. *Wir benötigen nun zu dem modifizierten 2PC ein Terminierungsprotokoll im Falle eines Koordinatorausfalls, das in keinem Fall blockiert, insbesondere auch dann nicht, wenn die Bedinungen von 1 gelten. Ist ein Teilnehmer in einem unsicheren Zustand und der Koordinator ist nicht erreichbar, startet er das folgende Terminierungsprotokoll:*
   (a) *Er sendet eine Anfrage an alle anderen Teilnehmer.*
   (b) *Wenn mindestens ein anderer Teilnehmer auf Commit oder Abort entschieden hat, entscheiden sich die übrigen Teilnehmer entsprechend.*
   (c) *Wenn alle anderen erreichbarer Teilnehmer unentschieden sind, entscheiden alle Teilnehmer auf Abort.*

4. *Wir müssen nun zeigen, dass das Protokoll nicht blockiert, wenn (a) keine Kommunikationsfehler auftreten können und höchstens ein Prozess ausfallen kann, oder (b) keine Kommunikationsfehler auftreten können und Broadcast-Meldungen atomar ausgeliefert werden. Wir beweisen dies für die verschiedenen Zustände, in denen der Koordinator zu dem Zeitpunkt, zu dem er abstürzt, sein kann.*
   (a) *Hat der Koordinator noch keine Entscheidung gesendet, hat kein Teilnehmer, der Koordinator eingeschlossen, committet. Deshalb können alle Teilnehmer auf Abort entscheiden.*
   (b) *Wenn der Koordinator eine Commit-Nachricht zu mindestens einem Knoten gesendet hat, aber noch nicht den Commit-Logeintrag geschrieben hat, dann unterscheiden wir für die Situationen (a) und (b)*
      - *Es kann höchstens ein Prozess ausfallen: Dieser Prozess ist der Koordinator, deswegen sind alle Teilnehmer verfügbar (und erreichbar, da es keine Kommunikationsfehler gibt), damit auch der Teilnehmer, der die Commit-Meldung erhalten hat. Damit wird das Terminierungsprotokoll die korrekte Entscheidung Commit fällen.*
      - *Atomares Broadcast: Die Entscheidung des Koordinators erreicht entweder alle oder keinen der Prozesse. Im ersten Fall übernehmen alle die Entscheidung des Koordinators, im zweiten Fall können alle abbrechen, denn dann hat weder der Koordinator noch andere ausgefallene Prozesse die Broadcast-Meldung erhalten und deswegen wurde die Transaktion an keinem Knoten committet.*
   (c) *Der Koordinator hat bereits einen Commit-Logeintrag geschrieben*
      - *Es kann höchstens ein Prozess ausfallen: Alle Teilnehmer haben die Commit-Nachricht erhalten, da es keine Kommunikationsfehler gibt.*
      - *Atomares Broadcast: Da der Koordinator den Commit-Logeintrag nicht schreibt, bevor er seine eigene Broadcast-Nachricht empfangen hat, ist sicher, dass alle Teilnehmer die Nachricht empfangen haben, und deshalb nicht in einem unsicheren Zustand sind.*
5. *Wenn der Koordinator Recovery von einem Fehler durchführt und einen* Start 2pc-*Logeintrag ohne zugehörigen* Commit *bzw.* Abort-*Eintrag findet, muss er bei den anderen Teilnehmern anfragen, welche Entscheidung sie getroffen haben.*

Aufgabe 2: 3-Phase Commit Protokolle

Wir betrachten das 3-Phase Commit Protokoll (3PC), das nur im Falle von Knotenausfällen Konsistenz garantiert. Im Terminierungsprotokoll sammelt der neue Koordinator zuerst die Stimmen der erreichbaren Teilnehmer ein und entscheidet auf Grundlage dieser Stimmen, wie weiter vorgegangen werden soll (mit Hilfe der Terminierungsregeln TR1 - TR4). Der neue Koordinator kann jedoch auch allein auf Grund seines eigenen Zustandes über den Ausgang einer Transaktion entscheiden und muss somit nicht bei den übrigen Teilnehmern anfragen. Entwickeln Sie ein derartiges Terminierungsprotokoll und zeigen Sie, dass es Konsistenz garantiert. Was sind die Vorteile und die Nachteile Ihres Protokolls im Vergleich zu dem ursprünglichen?

In dem modifizierten 3PC Protokoll wird bei Ausfall des Koordinators K_0 zuerst ein neuer Koordinator K_1 gewählt. K_1 entscheidet alleine über den Ausgangs einer Transaktion, in dem er folgende Terminierungsregeln anwendet:

- *TR1: Falls K_1 die Transaktion bereits abortet hatte, sendet er den übrigen Teilnehmern eine Abort-Nachricht.*
- *TR2: Falls K_1 die Transaktion bereits committet hatte, sendet er den übrigen Teilnehmern eine Commit-Nachricht.*
- *TR3: Falls K_1 im unsicheren Zustand ist, d.h. er hat das Pre-Commit des vorherigen Koordinators noch nicht erhalten, selbst aber "YES" gewählt, dann entscheidet er auf abort, sendet den übrigen Teilnehmern eine Abort-Nachricht und abortet.*
- *TR4: Falls K_1 committable ist, d.h. er hatte das Pre-Commit des vorherigen Koordinators bereits erhalten, dann sendet er zuerst eine Pre-Commit-Nachricht an die übrigen Teilnehmer, wartet auf ACKs und sendet dann die Commit-Nachricht.*

Das modifizierte Protokoll unterscheidet sich im Prinzip nur in der Regel TR3, da das "unsicher" von K_1 bereits ausreicht, um die Transaktion abzubrechen.

Wir zeigen nun, dass dieses Protokoll Konsistenz garantiert, indem wir zuerst die Terminierungsregeln überprüfen und dann Überlegungen anstellen, was passiert, wenn K_1 vor der vollständigen Beendigung der Transaktion auf allen Knoten ausfällt.

- *K_1 wendet TR1 an. Das heisst, dass K_0 mindestens eine "No"-Stimme erhalten hatte und deshalb kein pre-commit gesendet hat. Alle anderen Teilnehmer haben die Transaktion deshalb ebenfalls abgebrochen oder sind in einem unsicheren Zustand. Die Abort-Nachricht informiert die unsicheren Knoten über den Ausgang der Transaktion.*
- *K_1 wendet TR2 an. K_1 kann nur committet haben, wenn er von K_0 die Commit-Meldung erhalten hatte. Das bedeutet, dass alle Teilnehmer vorher mit "Yes" gestimmt hatten, die Pre-Commit-Nachricht empfangen hatten und ihr ACK gesendet hatten. Es genügt nun, die Commit-Nachricht zu senden, denn es kann sein, dass einige Knoten diese noch nicht erhalten haben.*
- *K_1 wendet TR3 an. Das ist der eigentlich interessante Fall. Denn dies fügt die Möglichkeit einer Zustandsänderung für einige Teilnehmer hinzu, die es im ursprünglichen 3PC nicht gegeben hat. Wenn K_1 unsicher ist, dann können die übrigen Teilnehmer die Transaktion abortet haben, selbst unsicher sein, oder committable sein. In ersteren beiden Fällen stellt die Abbruchentscheidung von K_1 kein Problem dar. Für einen committable Teilnehmer heisst das aber, dass er die Transaktion abbrechen muss, obwohl er sicher weiss, dass alle Teilnehmer in der ersten Runde mit "Yes" gestimmt hatten. Das kam im ursprünglichen Protokoll nicht vor. Wichtig für die Konsistenz ist jedoch, dass noch kein Knoten die Transaktion committet haben kann, so dass auch in diesem Fall alle dieselbe Entscheidung über den Ausgang der Transaktion treffen.*
- *K_1 wendet TR4 an. Dadurch, dass K_1 committable ist, weiss er, dass alle Knoten "Yes"-Stimmen abgegeben haben und keiner die Transaktion abgebrochen hat. Er sendet die Pre-Commit-Nachricht nochmal, da sie evtl. noch nicht bei allen Teilnehmern angekommen ist und führt das Protokoll dann genau weiter durch, wie es K_0 getan hätte.*

Was passiert aber nun, wenn K_1 während dieses Vorgangs abbricht? Man stelle sich folgendes Szenario vor: Der nun neu gewählte Koordinator K_2 ist zu diesem Zeitpunkt committable. Es ist also möglich, dass K_1 zum Zeitpunkt des Absturzes die Transaktion bereits committet hatte, committable war oder unsicher war. Im ersten Fall muss die Transaktion von allen committet werden, in letzterem Fall hat K_1, Regel TR3 anwendend, evtl. die

Transaktion bereits abgebrochen, d.h. die Transaktion muss von allen abgebrochen werden. K_2 kann in diesem Fall nicht selbständig entscheiden. In dem Fall, dass alle verfügbaren Knoten committable sind, kann nicht einmal in Zusammenarbeit eine Entscheidung gefunden werden und das System blockiert. Die Terminierungsregeln von K_2 lauten wie folgt:

- *TR1 und TR2 bleiben identisch mit derselben Begründung wie bei oben.*
- *Auch TR3 bleibt identisch, denn wenn er unentschieden ist, hat er das evtl. von K_1 gesendete Pre-Commit noch nicht erhalten, selbst noch kein ACK gesendet und deshalb kann bis jetzt kein Teilnehmer die Transaktion committet haben. Er kann sicher das Abort einleiten.*
- *TR4 lautet für ihn: Falls er selbst committable ist, wird die Entscheidung blockiert, bis K_1 wieder verfügbar ist und eine Entscheidung mitteilen kann.*

Der Vorteil dieses Protokolls ist, dass der neu gewählte Koordinatore selbständig entscheiden kann und deswegen Nachrichten eingespart werden. Die Nachteile sind, dass Transaktionen, die eigentlich committet werden könnten, unnötig abgebrochen werden (im Falle von TR3), und dass das System blockieren kann, wenn auch der neue Koordinator ausfällt.

(Die Übungen sind Kapitel 7 des [BHG87] entnommen. Falls Sie Zeit haben, versuchen Sie einige der übrigen Aufgaben zu lösen.