Aufgabe 8: Atomic Transactions

Literatur:

[1] Lampson, B. W.; et al.
Distributed Systems - Architecture and Implementation
Springer Verlag, 1981

[2] Tanenbaum, A. S.
Modern Operating Systems
Prentice Hall, 1992

[3] Singhal, M.; Shivaratri, N. G.
Advanced Concepts in Operating Systems
McGraw-Hill, Inc: New York et al. 1994

a) Geben Sie einen Überblick, wie bei Lampson Transaktionen implementiert werden!

b) Welche Fehlerklassen werden in dem vorgestellten Fehlermodell unterschieden? Geben Sie jeweils ein Beispiel!

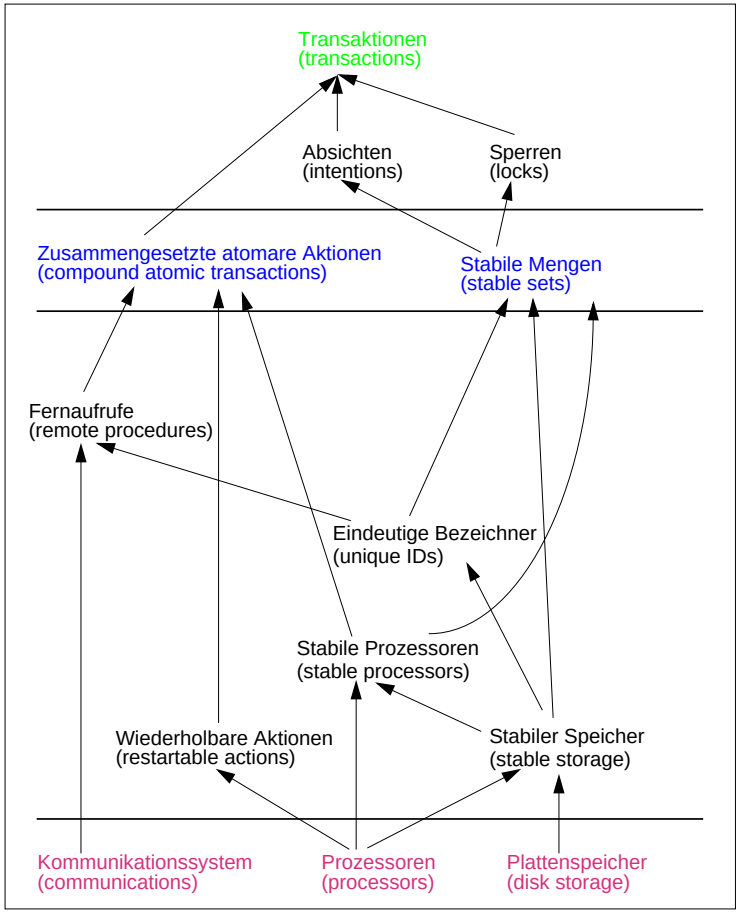
c) Wie kann stabiler Speicher implementiert werden?

d) Beschreiben Sie das "Zwei Phasen Commit Protokoll"

e) Welche Schwierigkeiten können bei diesem Protokoll auftreten und wie kann man sie lösen?

Lösungsvorschlag Aufgabe 8: Atomic Transactions

a) Geben Sie einen Überblick, wie bei Lampson Transaktionen implementiert werden!



Basierend auf fehlerbehafteten physikalischen Geräten werden logische, bis zu einem gewissen Grad fehlerfreie Geräte definiert, mit denen (auf einem bereits hohen Niveau) Transaktionen leicht implementiert werden können.

b) Welche Fehlerklassen werden in dem vorgestellten Fehlermodell unterschieden?

erwünscht
Der Aufruf liefert das erwartete, positive Ergebnis.

Fehler (unerwünscht, erwartet)
Es wird nicht die gewünschte Reaktion erzielt, aber der Fehler wird im Programm abgefangen und behandelt.

Ruin (unerwünscht, unerwartet)
Diese Fehler sind sehr selten und die Fehlerbehandlung ist unverhältnismäßig aufwendig, so daß auf eine Fehlerbehandlung verzichtet wird. Er muß aber in der Spezifikation aufgeführt sein.

Beispiel Lesen einer Seite:

Eine zu lesende Seite des Plattenspeichers ist "good" - die Get-Operation auf den Plattenspeicher liefert aber "looks Bad". Der Fehler kann durch wiederholtes Lesen vom Plattenspeicher behoben werden. -> entspricht der Kategorie *Fehler*.
Wird allerdings nach 'n' Wiederholungen immer noch keine korrekte Seite gelesen, so betrachtet man dies als *Ruin*. Das *erwünschte* Ergebnis eines Leseaufrufes ist natürlich, daß die zu lesende Seite korrekt gelesen wird.

Der Fall Eine Seite wird 'n'mal mit Ergebnis "looks Bad" gelesen impliziert, daß bei n-maligen Lesen einer Seite mit Ergebnis bad die Seite vom System als unbrauchbar behandelt werden muß - auch dann, wenn der eigentliche Zustand der Seite unklar ist.

c) Wie kann stabiler Speicher implementiert werden?

sorgfältiger Speicher
Auf einem realen, nichtflüchtigen und direkt zugreifbaren Speicher, z.B. Plattenspeicher oder batteriegepufferte Speicher, wird ein sorgfältiger Speicher aufgebaut. Der Unterschied zwischen dem realen Speicher und dem sorgfältigen Speicher besteht in den Operationen *lesen* und *schreiben*.
Beim sorgfältigen Lesen wird das Lesen gegebenenfalls mehrmals, bis zu einer vorgegebenen Schranke wiederholt, bis der Zustand *good* ist.
Beim sorgfältigen Schreiben wird nach jedem Schreiben die geschriebene Seite sofort wieder (einfach) gelesen und kontrolliert, ob der Zustand *good* ist. Dieser Vorgang kann ebenfalls bis zu einer vorgegebenen Schranke wiederholt werden.

stabiler Speicher
Dieser sorgfältige Speicher kann zu einem stabilen Speicher erweitert werden.
Dazu wird mit stabilen Seiten gearbeitet. Eine stabile Seite besitzt eine nicht verlustverwandte Schattenseite, d.h. eine zweite Seite mit gleichem Inhalt ist auf einem anderen physikalischen Medium abgespeichert, so daß nach dem physikalischen Verlust des einen Mediums immer noch das andere zur Verfügung steht.
Beim stabilen Lesen wird zuerst die Seite sorgfältig gelesen, wenn diese letztendlich den Zustand *bad* liefert, wird die Schattenseite sorgfältig gelesen.
Beim stabilen Schreiben wird zuerst die Seite sorgfältig geschrieben, anschließend wird die Schattenseite sorgfältig geschrieben.
Zu bestimmten Zeitpunkten wird eine Bereinigung vorgenommen. Diese erfolgt nach dem Systemstart, nach einem Zusammenbruch oder nach jedem Vielfachen des Verlustintervalls. Es wird angenommen, daß in einem Verlustinterval höchstens eine physikalische Seite von dem Paar (Seite, Schattenseite) verloren gehen kann. Bei der Bereinigung werden sowohl die Seite als auch die Schattenseite sorgfältig gelesen und miteinander verglichen. Falls genau eine den Zustand *bad* liefert, werde die Daten der anderen sorgfältig geschrieben. Falls beide den Zustand *good* liefern, aber verschiedene Daten enthalten, wird eine zufällig ausgewählt und sorgfältig geschrieben.

d) Beschreiben Sie das "Zwei Phasen Commit Protokoll"

Das Protokoll nimmt an, daß ein Teilnehmer der Transaktion sich als Koordinator verhält. Dieser könnte zum Beispiel anhand eines Wahlalgorithmus bestimmt werden, oder ist durch die Problemstellung implizit definiert. Desweiteren wird ein stabiler Speicher und ein Write-Ahead-Log vorausgesetzt.

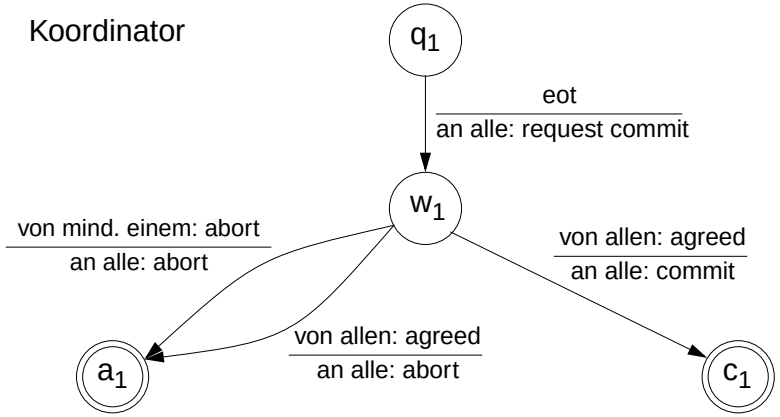
Das Protokoll läuft in 2 Phasen ab:

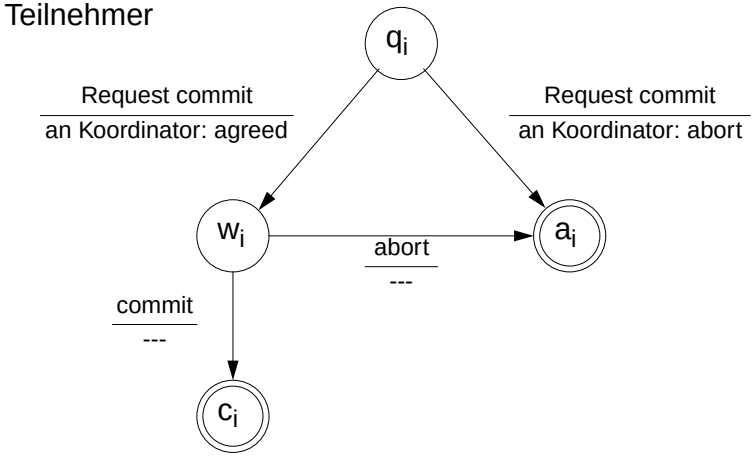
In der 1. Phase fragt der Koordinator bei allen an der Transaktion beteiligten Rechnern nach, ob sie bereit sind, die Transaktion ordnungsgemäß zu beenden. Jeder angesprochene Rechner teilt ihm seine individuelle Antwort mit. Nach dem Eintreffen aller Antworten, bzw. dem Ablaufen der dazugehörigen Timeouts, wird die Entscheidung gefällt, ob die Transaktion erfolgreich beendet oder abgebrochen wird.

Nun beginnt die 2. Phase. Sobald der "Commit"-Protokolleintrag beim Koordinator geschrieben ist, gilt die Transaktion als erfolgreich und alle Änderungen müssen geschrieben werden. Dazu werden an alle Commit-Nachrichten versandt, die Rechner schreiben einen Protokolleintrag und danach die Daten (Analog für Abort).

Die Zustände und die Verhaltensweisen der Teilnehmer kann durch Automaten geeignet dargestellt werden - siehe dazu nächste Seite.

Systemausfälle vor der ersten oder in der ersten Phase führen zum Abbruch der Transaktion. Systemausfälle in der zweiten Phase ändern nichts am Zustand der Transaktion (Commit/Abort) sondern zögern nur die Beendigung hinaus, da Wiederholungen stattfinden. Durch das Write-Ahead-Log kann der Zustand der vor dem Ausfall laufenden Transaktion(en) beim Wiederanlauf identifiziert und dementsprechend reagiert werden.





e) Welche Schwierigkeiten können bei diesem Protokoll auftreten und wie kann man sie lösen?

Das Zwei-Phasen-Commit-Protokoll gewährleistet eine globale Atomarität in einer verteilten Transaktion. Die große Schwierigkeit des Algorithmus liegt dabei im blockierenden Verhalten. Fällt der Koordinator aus, müssen die Teilnehmer der Transaktion bis zum Wiederanlauf des Koordinators warten.

Die Lösung dieses Problems ist die Einführung eines weiteren Zustandes in den Automaten, wodurch das Drei-Phasen-Commit-Protokoll entsteht. Allerdings hat dieses Protokoll das Problem, daß bei gleichzeitigen Auftreten von 2 Fehlern die Konsistenz nicht gewährleistet werden kann.

Zitat aus Tanenbaum [2]:

All in all, transactions offer many advantages and thus are a promising technique for building reliable distributed systems. Their chief problem is their great implementation complexity, which yields low performance.
These problems are being worked on, and perhaps in due course of time they will be solved.