Einsendeaufgabe 3

Replikation und Konsistenz

Wintersemester 2025/2026
05.01.2026

EINGEREICHT BEI
FernUniversität in Hagen
Fakultät für Mathematik und Informatik
Kooperative Systeme
Verteilte Systeme
Prof. Dr. Christian Icking

EINGEREICHT DURCH
Emily Lucia Antosch
emilyluciaantosch@web.de
Hamburg

Contents

1. Aufgabe 3.1

Betrachten Sie die Ausführungsgeschichte der vier Prozesse P_1, P_2, P_3 und P_4:

1. Ist diese Ausführungsgeschichte **sequentiell** konsistent?
2. Ist diese Ausführungsgeschichte **kausal** konsistent?

Bitte begründen Sie Ihre Antwort.

Lösung 3.1 Lösung 1

1. Die Ausführungsgeschichte ist nicht sequentiell konsistent, da in P3, im Gegensatz zu P4, die Lesereihenfolge von x erst 1, dann 3 und dann 2 ist. Dies ist inkonsistent mit der Reihenfolge von P4, da hier eine Reihenfolge von 1, dann 2 und dann 3 gesetzt ist.
2. Die Ausführungsgeschichte ist auch nicht kausal konsistent, da wir keine Möglichkeit finden, die alle kausalen Ketten mit der Ausführungsgeschichte zu vereinen. Wenn wir eine Reihenfolge von x = nil, 1, 2, 3 wählen, dann brechen wir die Kausalkette in P3, die erst 3 und dann zwei liest, aber genügen der in P4. Wenn wir die Reihenfolge x = nil, 1, 3, 2 wählen, genügen wir der Kausalkette in P3, brechen aber die P4. Da wir keine anderen Reihenfolgen zulassen können (P2, P3, P4) lesen als erste Aktion $x = 1$, gibt es keine Kette an Aktionen, die allen kausalen Ketten genügt.

2. Aufgabe 3.2

In einem System von n Servern $S_1, ..., S_n$ mit replizierten Daten soll die client-zentrierte Konsistenz gewährleistet werden. Clients schicken Anfragen mit Lese- oder Schreiboperationen an das System, die jeweils von genau einem der Server bearbeitet und beantwortet werden.

1. Welche Operationen müssen im System verbreitet werden?
2. Die Verbreitung der Operationen soll mit Hilfe von einem Vektor-Uhr-System realisiert werden.
   - Was bedeuten die einzelnen Komponenten der Vektor-Uhr?
   - Wie kann eine verbreitete Operation eindeutig im System identifiziert werden?
   - Wie gibt ein Server einem anderen Server Operationen weiter?
   - Wie aktualisiert ein Server seine Vektor-Uhr, wenn er Operationen von einem anderen Server erhalten hat?
3. Für alle vier Arten der client-zentrierten Konsistenz geben Sie jeweils bitte an, welche Informationen ein Client bei Anfragen mitliefern muss und was ein Server sicherstellen muss, bevor er die Antwort zurückschickt.

Lösung 3.2 Lösung 2

1. Grundsätzlich müssen nur Schreiboperationen im System verbreitet werden, da Leseoperationen nichts an der Konsistenz des Systems selbst ändern und auch nur lokal auf dem Server ausgeführt werden.
2. Eine Vektor-Uhr zeigt an, zu welchem Zeitpunkt welche Aktion ausgeführt wurde. In einem System mit n Servern wird eine Vektor-Uhr mitgeführt, die n Komponente hat. Jeder Server führt dabei seine eigene Vektor-Uhr. Für den i-ten Server stellt VC[i] den Zeitwert der derzeitigen Aktion dar. Alle anderen Werte stellen für den selben Server die Menge der Aktionen dar, die von den anderen Servern gemacht wurden, von denen der Server i weiß. Eine Operation kann klar durch das Paar (`server_id`, `timestamp`) definiert werden. (Beispiel (A, 8), also Server A zum Zeitpunkt 8). Wenn ein Server S_i einem anderen Server S_j eine Aktion weitergeben will, dann muss dieser die Operation(en) selbst übergeben und den Zeitstempel für jede Operation. Diese Infos können dann dem Server S_j den nötigen Kontext geben, um die Operationen erfolgreich abzuschließen. Wenn S_j dann seine interne Zeit updaten will, dann nimmt er `max(ts_i, ts_j)` für jede Komponente, um die Kausalität zu updaten. Nach dem Abschließen der Aktion wird der Zeitstempel VC[j] + 1 gerechnet.

Lösung 3.2 Lösung 3

3. Für die vier Arten der client-zentrierten Konsistenz:
    - Monotones Lesen
        - Der Client muss die Menge an Schreiboperationen bereitstellen, die für die Leseoperation relevant sind
        - Der Server muss bereitstellen, dass alle Schreiboperationen durchgeführt worden sind. Falls dies nicht der Fall ist, muss der Server sich die relevanten Infos von anderen Services anfragen oder die Daten an einen Service weiterleiten
    - Monotones Schreiben
        - Der Client muss die Menge an Schreiboperationen bereitstellen, die von dem Client durchgeführt worden sind
        - Der Server muss bereitstellen, dass alle Schreiboperationen in der richtigen Reihenfolge durchgeführt worden sind. Falls dies nicht der Fall ist, muss der Server sich die relevanten Infos von anderen Services anfragen oder die Daten an einen Service weiterleiten. Falls eine weitere Schreiboperation durchgeführt werden musste, muss das in der Menge der Schreiboperationen zu sehen sein.
    - Read your Writes
        - Der Client muss die Menge an Schreiboperationen bereitstellen, die von dem Client durchgeführt worden sind.
        - Der Server muss bereitstellen, dass alle Schreiboperationen durchgeführt worden sind. Erst wenn das bereitgestellt worden ist, darf die entsprechende Leseoperation durchgeführt werden.
    - Writes Follow Reads
        - Der Client muss die Menge an Schreiboperationen bereitstellen, die für die Leseoperation relevant sind. Und die Menge an Schreiboperationen, die vom Client ausgeführt worden sind.
        - Der Server muss bereitstellen, dass alle Schreiboperationen durchgeführt worden sind. Erst danach dürfen eventuelle Schreiboperationen durchgeführt werden. Im Anschluss muss sowohl die Menge von Schreiboperationen, die für das Lesen relevant sind, geupdated werden, als auch die Menge aller Schreiboperationen.

3. Aufgabe 3.3

Abbildung 1 zeigt die Ausführung der Prozesse P0, P1 und P2, die durch Nachrichtenaustausch kommunizieren, dabei werden Nachrichten m_i, $i = 0, \ldots, 7$ gesendet und empfangen. Prozess P_i setzt unabhängig die Checkpoints C_{ij} mit $i = 0, 1, 2$ und $j = 0, 1, 2$ ohne Koordination. Jeder Prozess startet mit einem initialen Checkpoint C_{i0} mit $i = 0, 1, 2$.

1. Gibt es einen Dominoeffekt (domino effect), wenn Prozess P2 nach dem Senden von Nachricht m6 ausfällt und eine Wiederherstellung (rollback recovery) startet?
2. Wo liegt die Recovery line?

Lösung 3.3 Lösung 4

1. Ja, durch den Fehler nach m_6 gibt es einen Dominoeffekt, der sich solange durchzieht, bis das System zum Initalzustand zurückspringt. Da das Absenden von m_6 durch den Rollback vergessen worden ist, kommt die Nachricht für P_1 aus dem Nichts und ist inkonsistent. Daher muss auch hier ein Rollback passieren. Dadurch vergisst P_1 m_7 und für P_0 kommt diese auch aus dem Nichts und muss einen weiteren Rollback machen. Die Linie ergibt sich als:

$$C_{22} \to m_6 \to C_{12} \to m_7 \to C_{02} \to m_5 \to C_{21} \to m_4 \to C_{11}$$
$$\to m_3 \to C_{01} \to m_2 \to C_{20} \to m_1 \to C_{10} \to m_0 \to C_{00}$$

2. Dadurch ergibt sich auch direkt, dass die Recovery Line $\{C_{00}, C_{10}, C_{20}\}$ ist.

4. Aufgabe 3.4

1. In einer Gruppe von 3 Prozessen, P1, P2 und P3, werden fünf Multicast-Nachrichten an die ganze Gruppe gesendet, wobei die Nachrichten 1 und 4 von P1 kommen, Nachricht 2 von P2 und die Nachrichten 3 und 5 von P3.
   - Wie viele Auslieferungsreihenfolgen der Nachrichten gibt es hier beim atomaren Multicasting?
   - Wenn wir jetzt voraussetzen, dass P1 die Nachricht 1 vor 4 sendet und P3 die Nachricht 3 vor 5, welche Auslieferungsreihenfolgen sind nun beim FIFO-atomaren Multicasting zulässig?
   - Außerdem nehmen wir noch an, dass P1 die Nachricht 2 vor dem Absenden von 4 und P3 die Nachricht 1 vor dem Absenden von 3 ausgeliefert hat. Welche Auslieferungsreihenfolgen der Nachrichten sind noch zulässig beim kausalen atomaren Multicasting? Bitte geben Sie diese Auslieferungsreihenfolgen an.
2. Betrachten wir jetzt Abbildung 2, in der die Reihenfolge der Punkte auf einem Zeitstrahl die Reihenfolge des Absendens und Empfangens der Nachrichten darstellt. Prozesse P1, P2 und P3 führen den Algorithmus zum Erzwingen kausaler Kommunikation aus (siehe Note 6.4 ab Seite 321), um die Nachrichten der Anwendungsschicht in einer kausalen Ordnung auszuliefern.
   - Geben Sie zu jeder Nachricht den Zeitstempel beim Senden und bei der Auslieferung an.
   - In welcher Reihenfolge werden die fünf Nachrichten jeweils von P1, P2 und P3 ausgeliefert?
   - Realisiert der Algorithmus ein atomar kausales Multicasting?

Lösung 3.4 Lösung 5

1. Die Antworten lauten:
   - Hier gibt es $120 = 5!$ Auslieferungsreihenfolgen, wenn wir nur atomares Mutlicasting ohne weitere Constraints annehmen. (Die Menge an Möglichkeiten für 5 Elemente mit Reihenfolge)
   - Es gibt $30 = 10 \cdot 3$ Auslieferungsreihenfolgen, da wir für 1 vor 4, 10 Möglichkeiten haben, die Nachrichten zu platzieren. Für 3 vor 5 bleiben noch jeweils 3 Plätze offen, wodurch sich pro Möglichkeit für 1 vor 4, 3 weitere Möglichkeiten ergeben (Nachricht 2 hat nur noch eine Lösung).
   - Hier gibt es nach den Regeln 1 vor 4, 3 vor 5, 2 vor 4 und 1 vor 3 noch 9 Möglichkeiten:

$$\{1, 2, 3, 4, 5\} \quad (1)$$

$$\{1, 2, 3, 5, 4\} \quad (2)$$

$$\{1, 2, 4, 3, 5\} \quad (3)$$

$$\{1, 3, 2, 5, 4\} \quad (4)$$

$$\{1, 3, 2, 4, 5\} \quad (5)$$

$$\{1, 3, 5, 2, 4\} \quad (6)$$

$$\{2, 1, 3, 4, 5\} \quad (7)$$

$$\{2, 1, 3, 5, 4\} \quad (8)$$

$$\{2, 1, 4, 3, 5\} \quad (9)$$

Lösung 3.4 Lösung 6

2. Wir gehen nun alle Events durch (Startzustand (0,0,0)):
   - P2 sendet Nachricht 2
     - $m_2 = (0, 1, 0)$
   - P1 sendet Nachricht 1
     - $m_1 = (1, 0, 0)$
   - P3 bekommt Nachricht 1
     - $P_3 = (0, 0, 0)$
     - $m_1 = (1, 0, 0)$
     - $P_3 = (1, 0, 0)$
   - P3 sendet Nachricht 3
     - $m_3 = (1, 0, 1)$
   - P2 buffert Nachricht 3
     - $P_2 = (0, 1, 0)$
     - $m_3 = (1, 0, 1)$
   - P2 bekommt Nachricht 1
     - $P_2 = (0, 1, 0)$
     - $m_1 = (1, 0, 0)$
     - $P_2 = (1, 1, 0)$
   - P2 bekommt Nachricht 3
     - $P_2 = (1, 1, 0)$
     - $m_1 = (1, 0, 1)$
     - $P_2 = (1, 1, 1)$
   - P1 bekommt Nachricht 2
     - $P_1 = (1, 0, 0)$
     - $m_2 = (0, 1, 0)$
     - $P_2 = (1, 1, 0)$
   - P3 sendet Nachricht 5
     - $m_5 = (1, 0, 2)$
   - P1 sendet Nachricht 4
     - $m_4 = (2, 1, 0)$
   - P3 buffert Nachricht 4
     - $P_3 = (1, 0, 2)$
     - $m_4 = (2, 1, 0)$
   - P1 buffert Nachricht 5
     - $P_1 = (2, 1, 0)$
     - $m_5 = (1, 0, 2)$

Lösung 3.5 Lösung 7

- P2 bekommt Nachricht 4
  - $P_2 = (1, 1, 1)$
  - $m_4 = (2, 1, 0)$
  - $P_2 = (2, 1, 1)$
- P3 bekommt Nachricht 2
  - $P_3 = (1, 0, 2)$
  - $m_2 = (0, 1, 0)$
  - $P_3 = (1, 1, 2)$
- P3 bekommt Nachricht 4
  - $P_3 = (1, 1, 2)$
  - $m_4 = (2, 1, 0)$
  - $P_3 = (2, 1, 2)$
- P2 bekommt Nachricht 5
  - $P_2 = (1, 1, 1)$
  - $m_5 = (1, 0, 2)$
  - $P_2 = (1, 1, 2)$
- P1 bekommt Nachricht 3
  - $P_1 = (2, 1, 0)$
  - $m_3 = (1, 0, 1)$
  - $P_1 = (2, 1, 1)$
- P1 bekommt Nachricht 5
  - $P_1 = (2, 1, 1)$
  - $m_3 = (1, 0, 2)$
  - $P_1 = (2, 1, 2)$
- Die Reihenfolge pro Prozess lautet also:
  - $P_1 = \{1, 2, 4, 3, 5\}$
  - $P_2 = \{2, 1, 3, 4, 5\}$
  - $P_3 = \{1, 3, 5, 2, 4\}$
- Und damit ergibt sich auch eine kausale, aber nicht atomare Auslieferungsreihenfolgen, da sich keine totale Reihenfolge aus den Auslieferungsreihenfolgen der Prozesse ablesen lässt.