

Willkommen bei Verteilte Systeme!

Von Datenbanken über Webdienste bis zu p2p und Sensornetzen.



Heute: **Koordination – Reihenfolge, Uhren, Konfliktvermeidung**
„Hattest du das schon gesehen?“

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Wiederholung

Organisation: Projekte

- Konkrete Ziele entwerfen
- Planning Poker: 4, 8, 13 oder 20 Stunden.
 - online: atomic (Umfragen)
 - offline: Handzeichen (Faust = 10)
- Zielnoten nach Zeitschätzung: Ein Wochenende pro Person
- Ziele anpassen:
 - 13 Stunden für eine Person für 1,5
 - 20 für zwei Personen für 1,5

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Problem der Reihenfolge

Problem der Reihenfolge



- Repliziertes Bankkonto, 2 Rechenzentren (KA, FFM)
- Kunde in KA möchte 100 € einzahlen.
- Bänker in FFM möchte 5% Zinsen auf das Konto buchen.
- Beide Transaktionen zeitgleich.
- Werden in das jeweils andere Rechenzentrum repliziert.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Problem der Reihenfolge

Das Problem

- Je nach Reihenfolge der Messages anderer Kontostand.
- Solche Inkonsistenzen vermeiden!
- Wie lassen sich die Operationen ordnen?

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Uhren

Hardware Uhren

- Bestehen aus einem Quartz und 2 Registern.
- Der Quartz oszilliert in einer bestimmten Frequenz.
- counter-Register wird bei jeder Oszillation dekrementiert.
- Erreicht der Zähler 0, wird ein Interrupt abgesetzt.
 - Danach wird das counter-Register auf den Wert des holding-Registers gesetzt.
- Jeder Interrupt stellt einen tick dar.
- Die Software Uhr wird pro tick um 1 erhöht.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Synchronisation

Zeit-Synchronisation

Es existieren verschiedene Algorithmen:

- Cristians Algorithmus: Client-Server
- NTP: Weltzeit
- Berkeley: Clusterzeit

Wiederholung: Grundprobleme

- Einstieg:** Wie finde ich meinen Platz im Netz?
- Suche:** Wo gibt es, was ich brauche?
- Störungsresistenz:** Wie skaliert Gewünschtes besser als Unerwünschtes?
- Verbreitung:** Wie vermeide ich Flaschenhälse?
- Kommunikation:** Wie fließen Informationen durchs Netz?

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Ablauf heute

Ablauf heute

Koordination

- Reihenfolge ist relative
 - Timestamps als Lösung?
 - Uhren im Computer
 - Synchronisation von Uhren
 - Logische Uhren als Alternative
 - total geordneter Multicast
 - kausal geordneter Multicast
- Exklusiver Zugriff
- Wahlalgorithmen

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Problem der Reihenfolge

Reihenfolge 1

balance = 1000
balance = balance + 100
balance = balance * 1.05
return balance

- 1155.0
- Sicht aus Rechenzentrum KA.
 - Message des Kunden trifft zuerst ein.
 - Message des Bänkers danach.
 - Kontostand: 1'155 €.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Uhren + Synchronisation

Ziele Uhren + Synchronisation

- Sie kennen die Funktionsweise von Uhren in Computern
 - Sie kennen Methoden, um Uhren zu synchronisieren
- <https://xkcd.com/2867/>

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Uhren

Der internen Uhr vertrauen? Erfahrung

Kerberos

- "login failed: timed out after 5 minutes"
- log: password received 17442000 minutes after login.
- epoch ... (1970)

Admins mussten lokal an den Rechner und die Mainboard-Batterie tauschen.

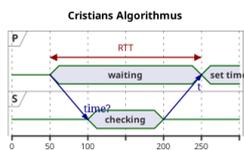
Ist ein Extremfall. Mikrowelle: Etwa +1 Minute pro Monat.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Synchronisation

Cristians Algorithmus



- P fragt Zeit von S an und startet timer.
- S liest die Zeit t und antwortet.
- P setzt seine Uhr auf $t + \frac{RTT}{2}$

Wiederholung: Implementierungen

	Einstieg	Suche
Gnutella	WebCache	Slow-Start + Keyword-Multicast
Kademlia	Suche nach eigener ID	xor-Hash-Hierarchie
BitTorrent	Tracker-URL	Kademlia / Tracker / Web
Freenet	Seed-Nodes suchen ID	Greedy Hash auf Small World
WebRTC	WebRTC Server	-
	Verteilung	Störung
Gnutella	Alt+NAlt, Range, Merkle-Tree	Heuristik/Credence
Kademlia	unterschiedlich	-
BitTorrent	Torrent	Wertung auf Tracker
Freenet	Chunk-Tree with Redundancy	Propagating Trust
WebRTC	-	-

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Ablauf heute

Ziele heute

- Sie verstehen, wie Operationen in einem verteilten System geordnet werden können
- Sie kennen die Funktionsweise von Uhren in Computern
- Sie kennen Methoden, um Uhren zu synchronisieren
- Sie kennen Alternativen zu synchronisierten Uhren
- Sie kennen Implementierungen für wechselseitigen Ausschluss (mutual exclusion) in verteilten Systemen
- Sie kennen grundlegende Wahlalgorithmen, um Knoten spezielle Rollen zuzuweisen

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Problem der Reihenfolge

Reihenfolge 2

balance = 1000
balance = balance * 1.05
balance = balance + 100
return balance

- 1150.0
- Sicht aus Rechenzentrum FFM.
 - Message des Bankers trifft zuerst ein.
 - Message des Kunden danach.
 - Kontostand: 1'150 €.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Uhren

Uhren

Computer verwenden 2 Arten von Uhren:

- time-of-day clocks
 - Gibt aktuelles Datum und Uhrzeit zurück.
 - Beispiel: `Java System.currentTimeMillis()`
- monotonic clocks
 - Geben eine Zahl zurück die monoton steigt.
 - Beispiel: `Java System.nanoTime()`

monoton: läuft nie rückwärts.

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Uhren

Clock Skew vs. Clock Drift

Wir vergleichen 2 Uhren:

- Clock Skew: Unterschied der Werte der Uhren
- Clock Drift: Unterschied zwischen der Frequenzen der Uhren
- Clock Skew != 0 => Uhren sind nicht synchronisiert sind
- Clock Drift != 0 => Clock Skew wird sich verändern
- Clock Skew zu UTC ca. 31 Sekunden pro Jahr
- Ursache: Unterschiede in der Frequenz des Quartz (auch bei baugleichen Uhren)
- Externe Einflüsse wie Temperatur
- => Wir müssen Synchronisieren!

Arne Babenhausenheide Verteilte Systeme 2: Koordination

Einstieg Reihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Synchronisation

PAUSE

--- PAUSE ---

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Übung Cristians Algorithmus

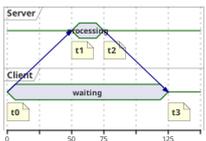
- Laufzeit Nachricht $P \rightarrow S$ und $S \rightarrow P$ jeweils 100 ms.
- S benötigt 10 ms für die Bearbeitung der Anfrage.
- $t = 500$ ms.
- Welche Zeit wird bei P eingestellt?

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP Berechnung



- Client startet Anfrage zu t_0 .
- Server empfängt Anfrage zu t_1 und sendet Antwort zu t_2 .
- Client empfängt Antwort zu t_3 .
- t_0 und t_3 jeweils in Client-Zeit, t_1 und t_2 in Server-Zeit.
- Offset berechnet sich: $offset = \frac{(t_1 - t_0) + (t_2 - t_3)}{2}$
- Offset wird verwendet, um die Zeit graduell anzupassen.

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP Übung

- Client stellt Anfrage bei $t_0 = 100$ ms.
- Server empfängt Anfrage bei $t_1 = 50$ ms und benötigt 10 ms zur Bearbeitung.
- Berechne das offset bei:
 - $t_{c \rightarrow s} = t_{s \rightarrow c} = 5$ ms
 - $t_{c \rightarrow s} = 5$ ms und $t_{s \rightarrow c} = 10$ ms
- Auf welche Zeit wird die Uhr des Client jeweils gestellt?

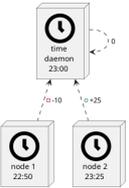
Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Berkeley Algorithmus - Schritt 2

Maschinen antworten mit ihrem offset.



Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Berkeley Algorithmus - Beobachtungen

- Bietet interne Synchronisation.
 - System wird nicht mit externer Uhr (UTC) synchronisiert.
- Solange das System intern einen einheitlichen Zeitbegriff verwendet, können Operationen geordnet werden.

Clusterzeit.

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Zusammenfassung Uhren + Synchronisation

Zusammenfassung Uhren + Synchronisation

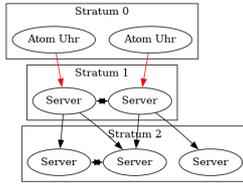
- wall time vs. monotonic clocks
- Uhren leiden under skew und drift.
 - \rightarrow müssen synchronisiert werden.
- externe Synchronisation: Cristians Algorithmus, NTP
- interne Synchronisation: Berkeley
- Uhren haben Unsicherheit

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP (Network Time Protocol): Diagramm



Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP graduelle Anpassung

- Was passiert bei offset < 0?
- Uhren sollten nicht rückwärts laufen!
- Bsp: jeder tick erhöht Software Uhr um 10 ms.
 - Idee: Verringerung des Inkrements, um Uhr schrittweise anzugleichen.
- Wird auch verwendet, um die Uhr vorwärts anzupassen.
- Graduelle Anpassung wird bei offsets > 128 ms nicht verwendet.

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP

Neben dem offset wird noch das delay berechnet:
 $delay = (t_3 - t_0) - (t_2 - t_1)$

Es werden 8 offset-delay Paare ermittelt und das Paar mit dem geringsten delay verwendet.

NTP erreicht Genauigkeiten von 1-50 ms.

30ms Verzögerung ist die Obergrenze für Interaktives!

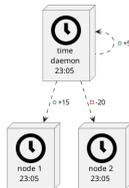
Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Berkeley Algorithmus - Schritt 3

Zeitserver berechnet Durchschnitt der Uhrzeiten und sendet offsets an Maschinen.



Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Spanner

- Nutzt timestamps in der Form $[T_{lower} T_{upper}]^1$
- „True Time Service“
- Kommt auf 6ms Genauigkeit
- Transaktionen werden verzögert, damit T_{upper} auf jeden Fall verstrichen ist.

¹Spanner-Beschreibung: <https://levelup.gitconnected.com/how-google-spanner-assigns-commit-timestamps-the-secret-sauce-of-its-strong-consistency-8bc1436142f6>

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Zusammenfassung Uhren + Synchronisation

PAUSE

--- PAUSE ---

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

NTP (Network Time Protocol): Ablauf



- Bestandteile des Systems werden in Strata unterteilt.
- Referenz-Uhren befinden sich in Stratum 0.
- Ein Server in Stratum n kontaktiert Server in Stratum $n - 1$ zur Synchronisation.
- Oft werden mehrere Server angefragt und die Ergebnisse statistisch behandelt (Mittel, Ausreißer).

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Niemals rückwärts! Erfahrung

- change password.
- error: attempt to login before password set.

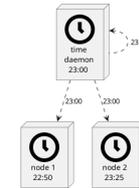
Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Berkeley Algorithmus - Schritt 1

Zeitserver sendet periodisch eigene Zeit an alle Maschinen im Netzwerk.



Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Berkeley Algorithmus - Übung

3 Rechner und ein Server im System.

Gebe die Nachrichten des Berkeley Algorithmus an. Welche Zeit wird im System eingestellt?

Server	11:00
Alice	10:55
Bob	11:15
Carol	11:10

Nr	von	an	Inhalt
1	Server	A,B,C	11:00
2	Server	Server	0
...			
9			

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Wall Time Timestamps - Fazit

- Uhren sind immer mit einem Fehler versehen.

Arne Babenhausen
Verteilte Systeme 2: Koordination

Einstieg
Reihenfolge ist relativ
Logische Uhren
Gegenseitiger Ausschluss
Abschluss

Synchronisation

Logische Uhren + Multicast ordnen

- Ziele:
- Sie kennen Alternativen zu wall time clocks.
 - Sie verstehen den Unterschied zwischen partieller und totaler Ordnung.
 - Sie verstehen Lamport clocks?
 - Sie kennen total geordneten Multicast mit Lamport clocks.
 - Sie verstehen Vector clocks.
 - Sie kennen kausal geordneten Multicast mit vector clocks.

Arne Babenhausen
Verteilte Systeme 2: Koordination

Lamport Uhren

- Die exakte Uhrzeit interessiert uns nicht
- Reihenfolge von Ereignissen
- ⇒ Zeit-Ordnung
- Timestamps sollen Kausalität berücksichtigen:
 - Wenn a kausal vor b passiert ist, dann $timestamp(a) < timestamp(b)$.
 - Tür wird geöffnet bevor man eintritt

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Umsetzung Lamport Clocks

Jeder Prozess P_i erstellt einen lokalen Zähler C_i und wendet folgende Regeln an:

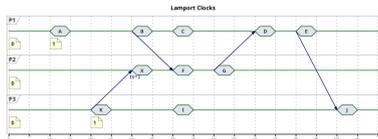
- Für 2 **sukzessive Ereignisse**, die in P_i stattfinden, wird C_i um 1 erhöht.
- Wenn eine Nachricht von P_j **gesendet** wird, erhält sie den timestamp $ts(m) = C_j$.
- Wenn eine Nachricht von P_j **empfangen** wird, setzt P_i C_i auf $\max(C_j, ts(m)) + 1$

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Schritt 2

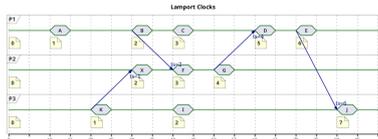


Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Ende



Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks, Bedeutung

- $A \rightarrow B \implies ts(A) < ts(B)$ aber
- $ts(A) < ts(B) \implies \{A \rightarrow B\} \text{ OR } \{A, B \text{ gleichzeitig}\}$

„Wenn es vorher war, dann ist der Zeitstempel kleiner.“

Aber nicht: „Hattest du hier meine Nachricht schon gesehen?“

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Vector Uhren - Umsetzung

- Jeder Prozess verwaltet einen Vektor von Integer Uhren.
- Bei N Prozesse hat jeder Vektor N Elemente.
- Ein Prozess i verwaltet einen Vektor $V_i[0 \dots (N-1)]$
- $V_i[i]$ ist die lokale Uhr des Prozesses i.
- Falls $V_i[j]=k$, dann weiß i, dass k Ereignisse in P_j stattgefunden haben.
- In jedem Knoten $O(N) \rightarrow$ Wir hätten gerne $O(\log(N))$ (haben wir aber nicht).

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Die Happens-Before Relation

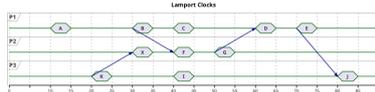
- Logische Beziehung zwischen 2 Ereignissen.
- Notation: $a \rightarrow b$: a ist vor b passiert.
- Regeln:
 - Innerhalb eines Prozesses $a \rightarrow b$, if $time(a) < time(b)$.
 - Wenn P_1 eine Nachricht m an P_2 sendet: $send(m) \rightarrow receive(m)$.
 - Wenn $a \rightarrow b$ und $b \rightarrow c$, dann auch $a \rightarrow c$ (Transitivität)
- Partielle Ordnung von Ereignissen.

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Beispiel Lamport Clocks

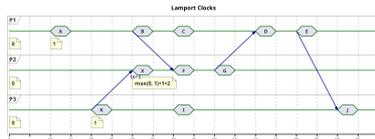


Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Schritt 3



Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks

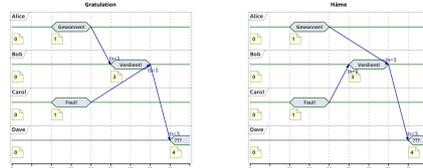
- Ein Paar von gleichzeitigen (concurrent) Ereignissen hat keinen kausalen Pfad.
- Lamport timestamps müssen bei gleichzeitigen Ereignissen weder geordnet noch ungleich sein.
- $A \rightarrow B \implies ts(A) < ts(B)$ aber
- $ts(A) < ts(B) \implies \{A \rightarrow B\} \text{ OR } \{A, B \text{ gleichzeitig}\}$

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport: Wann reicht das nicht?



Das Ziel ist nicht, die Wirklichkeit abzubilden, sondern eine Datengrundlage für lokale Entscheidungen zu haben.

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Vector Uhren - Verwaltung

- Vor Ausführung eines lokalen Ereignisses führt P_i die Anweisung $V_i[i] += 1$ aus.
- Wenn P_i eine Nachricht sendet, wird der timestamp der Nachricht auf V_i gesetzt, nachdem $V_i[i] += 1$ ausgeführt wurde.
- Beim Empfang einer Nachricht in P_j :
 - $V_j[i] += 1$
 - $V_j[j] = \max(V_m[j], V_j[j])$, for $j \neq i$

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Partielle Ordnung

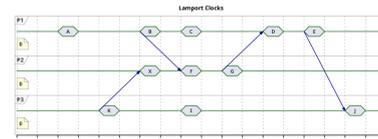
- Eine totale Ordnung erlaubt 2 beliebige Elemente miteinander zu vergleichen.
 - Für jedes Elementpaar kann die Aussage getroffen werden welches der Elemente größer ist.
 - Beispiel: natürliche Zahlen.
 - Eine Partiellen Ordnung kann nur einige vergleichen
 - Wir können nicht für alle Ereignispaare die Reihenfolge bestimmen.
- Gleichzeitig: „echte“ Reihenfolge unbekannt.

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Schritt 1

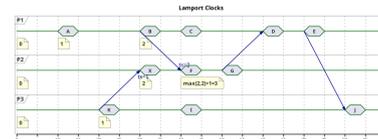


Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Schritt 4

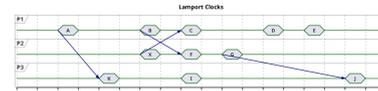


Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Lamport Uhren

Lamport Clocks - Übung



- Besteht ein kausaler Zusammenhang zwischen:
 - A, J
 - H, G
 - C, F
- Berechne die timestamps.

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Vektor Uhren

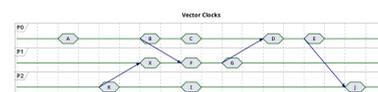
- Lamport Uhren:
 - $A \rightarrow B \implies ts(A) < ts(B)$ aber
 - $ts(A) < ts(B) \implies \{A \rightarrow B\} \text{ OR } \{A, B \text{ gleichzeitig}\}$
- Gibt es ein Verfahren, das $ts(A) < ts(B) \implies A \rightarrow B$ ermöglicht?

Arne Babenhauerheide Verteilte Systeme 2: Koordination

Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

Vector Uhren - Beispiel 1



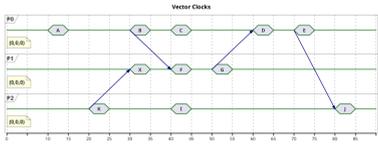
Selbe Ausgangssituation wie bei Lamport Clocks.

Arne Babenhauerheide Verteilte Systeme 2: Koordination

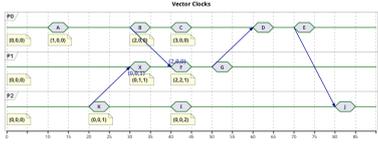
Einseitig Relihenfolge ist relativ Logische Uhren Gegenseitiger Ausschluss Abschluss

Vektor Uhren

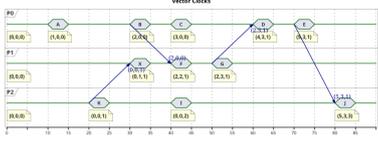
Vector Uhren - Beispiel 2



Vector Uhren - Beispiel 5



Vector Uhren - Gleichzeitigkeit



Paar	ts(1)	ts(2)	ts(1) < ts(2)	ts(2) < ts(1)	Folgerung
A, E	(1,0,0)	(5,3,1)	Ja	Nein	A kausal vor E
H, C	(0,0,1)	(3,0,0)	Nein	Nein	H und C gleichzeitig

Vector Uhren - Abschluss

Lampot Clocks:

- $A \rightarrow B \implies ts(A) < ts(B)$ aber
- $ts(A) < ts(B) \implies \{A \rightarrow B\}$ OR $\{A, B \text{ gleichzeitig}\}$

Vector Clocks:

- $ts(A) < ts(B) \implies A \rightarrow B$
- $\neg(ts(A) < ts(B)) \wedge \neg(ts(B) < ts(A)) \implies A, B \text{ gleichzeitig}$

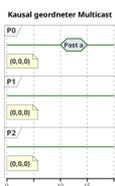
Kausal geordneter Multicast mit Vector Clocks

Mit wenigen Änderungen können Vector Clocks genutzt werden, um kausal geordnete Nachrichten sicherzustellen.

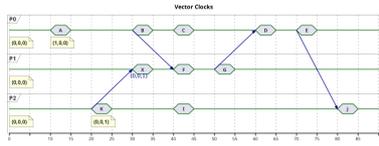
Folgendes Beispiel:

- Ein Bulletin Board Service.
- Benutzer treten Gruppen bei.
- Posts werden an alle Gruppenmitglieder gemulcastet.
- Könnte vollständig geordneten Multicast verwenden.
 - Wird aber nicht benötigt.

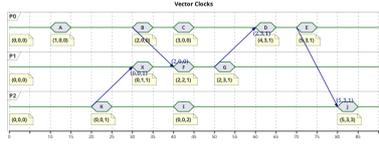
Kausal geordneter Multicast - Beispiel 1



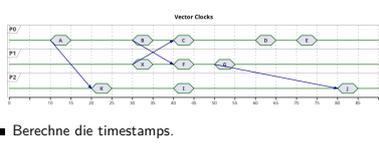
Vector Uhren - Beispiel 3



Vector Uhren - Beispiel 6



Vector Uhren - Übung



- Berechne die timestamps.
- Hängt J vielleicht kausal von A ab?
- Finden C und F gleichzeitig statt?

PAUSE

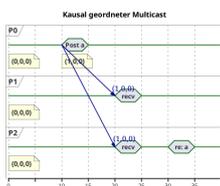
--- PAUSE ---

Bulletin Board - Anzeige

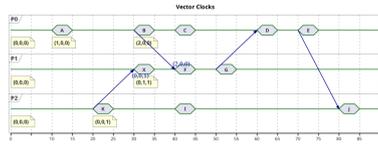
Betreff
Mach
Microkernels
Hurd; was: Microkernels
RPC Performance
Re: Mach

- Bei vollständiger Ordnung ist diese Liste bei jedem Benutzer in der gleichen Reihenfolge.
- Kausal geordneter Multicast erfordert nur, dass Reaktionen (Re: Mach) nach dem Post (Mach) angezeigt werden.
- Für die angezeigten Posts sind verschiedene Reihenfolgen möglich.

Kausal geordneter Multicast - Beispiel 2



Vector Uhren - Beispiel 4



Vector Uhren - kausale Abhängigkeit

Ereignis B hängt **vielleicht** kausal von A ab, wenn $ts(A) < ts(B)$.

$ts(A) < ts(B)$:

- für alle i : $ts(A)[i] \leq ts(B)[i]$ und
- es existiert mindestens ein Index k für den: $ts(A)[k] < ts(B)[k]$ gilt.

In diesem Fall gilt:

- A geht B kausal voraus.
- B hängt vielleicht kausal von A ab, da es Informationen von A geben könnte, die in B propagiert werden.

Vector Uhren, Bedeutung

Vector Clocks:

- $ts(A) < ts(B) \implies A \rightarrow B$
 - $\neg(ts(A) < ts(B)) \wedge \neg(ts(B) < ts(A)) \implies A, B \text{ gleichzeitig}$
- „Wenn der Zeitstempel kleiner ist, dann war es vorher.“

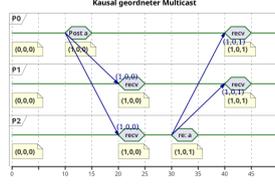
Kausal geordneter Multicast

- Vollständig geordneter Multicast stellt sicher, dass alle Nachrichten in gleicher Reihenfolge bearbeitet werden.
- Kausal geordnet bedeutet, dass Nachrichten, die sich gegenseitig beeinflussen könnten von allen Prozessen in gleicher Reihenfolge empfangen werden.

Kausal geordneter Multicast - Anpassungen

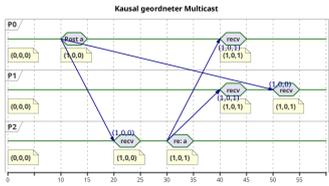
- Vector Clocks werden nur bei Empfang oder Senden einer Nachricht angepasst.
 - Beim Senden in P_i : $V_i[i] += 1$
 - Beim Empfang von m in P_j : für alle k : $V_j[k] = \max(V_j[k], V_m[k])$
- Eine Nachricht m (von P_x an P_j) wird erst von der Anwendung prozessiert wenn:
 - $V_m[k] = V_j[k] + 1$, m ist die nächste Nachricht, die $P_{-}\{i\}$ von $P_{-}\{k\}$ erwartet hat.
 - $V_m[x] \leq V_j[x]$ für alle $x \neq k$, P_j hat alle Nachrichten gesehen, die P_x gesehen hat als m gesendet wurde.

Kausal geordneter Multicast - Beispiel 3



Was passiert bei verzögerter Zustellung?

Kausal geordneter Multicast - Beispiel 4

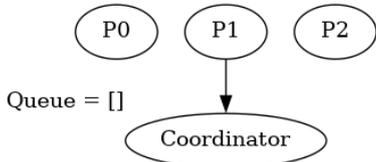


Gegenseitiger Ausschluss

- Ziele:
- Sie kennen die Grundlegenden Konzepte für Gegenseitigen Ausschluss.



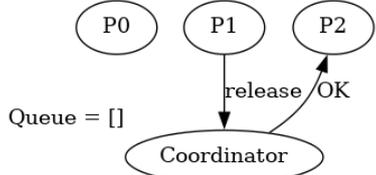
Zentralisierter Algorithmus - Happy Path



Falls die Ressource verfügbar ist, erhält der anfragende Prozess die Berechtigung.



Zentralisierter Algorithmus - Ressource wird frei



Sobald die Ressource wieder frei wird, gibt der Koordinator die Ressource an den ersten Prozess in der Queue.



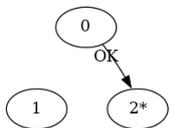
Verteilter Algorithmus

- Wenn ein Prozess, die Ressource benötigt sendet er eine Nachricht an alle Prozesse (auch sich selbst).
- Bei Erhalt so einer Nachricht:
 - Prozess hält Ressource nicht und möchte sie nicht: sendet OK.
 - Prozess hält Ressource: antwortet nicht.
 - Prozess möchte Ressource: vergleiche timestamp der Nachricht mit timestamp der eigenen Nachricht. Der niedrigere timestamp gewinnt.
- Prozess wartet Antworten aller Prozesse ab. Sobald er sämtliche OKs erhalten hat, verwendet er die Ressource.

Voraussetzung: Totale Ordnung der Nachrichten.



Verteilter Algorithmus - Gleichzeitiger Zugriff 3



- P0 benötigt die Ressource nicht mehr.
- P0 sendet OK an erste Anfrage in Queue.
- P2 erhält Zugriff.



Zusammenfassung Logische Uhren

- happens before Relation bestimmt eine partielle Ordnung.
- Lamport Clocks: Counter pro Prozess
 - Timestamps bilden totale Ordnung.
 - Vergleich von timestamps gibt keine Aussage zu Kausalität.
 - Total geordneter Multicast kann mit Lamport Clocks implementiert werden.
- Vector Clocks: Vector of Counter pro Prozess
 - Vergleich von timestamps gibt Aussage zu Kausalität.
 - Kausal geordneter Multicast kann mit vector clocks implementiert werden.



Gegenseitiger Ausschluss

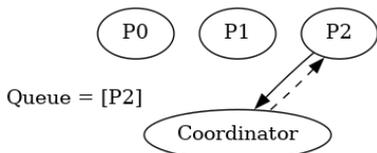
Das Problem: Einige Prozesse in einem verteilten System möchten exklusiven Zugriff auf eine Ressource.

- 2 Vorgehensweisen:
- Berechtigungsbasiert: Prozesse benötigen Berechtigung anderer Prozesse um auf Ressource zuzugreifen.
 - Tokenbasiert: Einzigartiges Token wird zwischen Prozessen weitergereicht. Wer das Token hält, hat Zugriff auf die Ressource.²

²Token-Basiert bis hin zu Funktionsargumenten: <https://fosdem.org/2022/schedule/event/spritelygoblins/>



Zentralisierter Algorithmus - Ressource belegt



- Koordinator hat Übersicht, ob Ressource momentan frei ist.

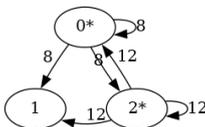


Zentralisierter Algorithmus - Fragen

- Wieviele Nachrichten werden ausgetauscht?
- Was passiert, wenn der Koordinator ausfällt?
- Was passiert bei vielen Anfragen?



Verteilter Algorithmus - Gleichzeitiger Zugriff



- P0 sendet Anfragen mit timestamp 8.
- P2 sendet Anfragen mit timestamp 12.



Verteilter Algorithmus - Bewertung

- Was passiert wenn ein Knoten ausfällt?
- Können wir den Algorithmus anpassen?
- Wieviele Nachrichten werden benötigt?



PAUSE

--- PAUSE ---

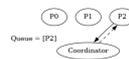


Zentralisierter Algorithmus

- Simuliert Vorgehen innerhalb einer CPU.
- Ein Prozess wird als Koordinator konfiguriert.
- Prozesse, die auf die Ressource zugreifen möchten, fragen dies beim Koordinator an.



Zentralisierter Algorithmus - Ressource belegt



- Koordinator hat Übersicht, ob Ressource momentan frei ist.
- Hier wird die Antwort an den anfragenden Prozess verzögert, bis die Ressource wieder frei wird.
- P2 wird dadurch geblockt.
- Der Koordinator speichert die Anfrage in einer Queue.

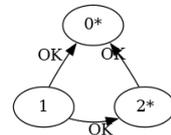


Zentralisierter Algorithmus - Bewertung

- Wieviele Nachrichten werden ausgetauscht?
- 3 Messages pro lock
- Was passiert, wenn der Koordinator ausfällt?
- System funktioniert nicht mehr
- Was passiert bei vielen Anfragen?
- Koordinator ist Flaschenhals



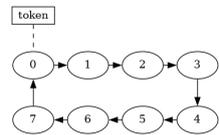
Verteilter Algorithmus - Gleichzeitiger Zugriff 2



- P1 sendet OK an beide Prozesse.
- P0 und P2 vergleichen timestamps.
 - P2 sendet OK.
 - P0 stellt P2s Anfrage in einer Queue ein.



Token Ring Algorithmus



- Prozesse werden in ringförmigen Overlay Netzwerk angeordnet.
- Erster Prozess erhält Token.
- Token stellt Berechtigung dar die Ressource zu verwenden.
- Wird Ressource nicht benötigt, wird Token weitergeleitet.



Token Ring Algorithmus - Bewertung

Vergleich

Zusammenfassung Gegenseitiger Ausschluss

- Was passiert wenn ein Knoten ausfällt?
- Wieviele Nachrichten werden benötigt?

Algorithmus	Nachrichten pro Ein/Austritt
Zentralisiert	3
Verteilt	2N-1
Token Ring	1, ..., ∞

- Mutex mit Koordinator
- Mutex verteilt
- Mutex Token Ring

Zusammenfassung

Reale Uhren:

- wall time vs. monotonic clocks
- Skew und Drift
- Synchronisieren: extern (Cristian, NTP), intern (Berkeley)

Logische Uhren:

- Lamport: Ein Zähler pro Knoten. „Wenn es vorher war, dann ist der Zeitstempel kleiner.“
- Vektor: N Zähler in jedem der N Knoten. Kausalität. „Wenn der Zeitstempel kleiner ist, dann war es vorher.“

Ausschluss: Koordinator oder verteilt ⇒ Zusätzliche Nachrichten.

Für koordinierte Projekte!



Verweise I

Bilder: