Verteilte Systeme SS2002:

Zeit und Ordnung

Marcel Waldvogel

Übersicht

- *Physikalische und logische Zeit*
- *Lokale und globale Ordnung*
- *Lokale und globale Zustände*
- *Fehlersuche in einem verteilten System*

Zeit

- *Physikalisch*
- *Logisch*
  - *Ereignisse*
  - *Zustände*

Physikalische Zeit

- *Atomuhr (International Atomic Time)*
  - *Drift etwa 1 in 10^{13}*
- *GPS: 1μs*
- *Tag ungenau*
  - *Schaltsekunden*
  - *UTC (Coordinated Universal Time)*
- *Computerzeit*
  - *Offset*
  - *Drift*
    - *Quarz: 1 in 10^6 bis 10^8*

Zeitsynchronisation

- *Cristian (1989)*
  - *Einzelne Anfrage, probabilistisch*
- *Gusella und Zatti (Berkeley, 1989)*
  - *timed: Master-Slave (Polling)*
  - *Mittelwert der Messungen liefert Grundlage für individuelle Befehle zur Korrektur (nicht "aktuelle" Zeit: Fehlerminimierung)*
- *NTP*

NTP

- *Mills (1995)*
  - *Ein Uhrenbaum mit Alternativen*
  - *Korrektur von Frequenz (Drift) und Offset*
- *Statistik*
  - *Analyse, Umgehung von Netzwerkdelays*
  - *1-10ms Genauigkeit*
- *Modi*
  - *Multicast/Broadcast (LAN)*
  - *Client-Server ("Procedure Call")*
  - *Symmetrisch*

Logische Zeit

- *Reihenfolge von Ereignissen*
- *Einzelner Prozess*
  - *Lokale Reihenfolge einfach bestimmbar*
  - $a \rightarrow_i b$
- *Verteiltes System*
  - *Mehrere Prozesse* p_i, $i \in \{1, ..., N\}$
  - *Globale Reihenfolge unklar*
- *"Happened-Before"*

Happened-Before

- *Leslie Lamport (1978)*
  - *Temporale Logik ("Sometime" is Sometimes "Not Never")*
  - $L_A T_E X$
- *Definition*
  - *Wenn* $\exists\ p_i : a \rightarrow_i b$, *dann* $a \rightarrow b$
  - *Für eine Nachricht m gilt* $send(m) \rightarrow recv(m)$
  - *Transitivität: Wenn* $a \rightarrow b$ *und* $b \rightarrow c$, *dann* $a \rightarrow c$

Logische Uhr

- *Logical Clock*
- *Numerischer Wert (Ordnung) der "Happened-Before"-Relation*
- *Operationsweise*
  - *L_i ist logische Uhr im Prozess p_i*
  - *L_i wird in p_i vor jedem Ereignis inkrementiert*
  - *Jede versandte Nachricht m enthält $t=L_i$*
  - *Wenn p_j (m, t) empfängt, wird $L_j := max(L_j, t)+1$ (+1 für Timestamp des Empfangsereignisses)*

Logische Uhr

- *Nachteile*
  - *L(a) < L(b) bedeutet nicht $a \rightarrow b$*
- *Abhilfe*
  - *Vektoruhren (Mattern 1989, Fidge 1991)*
    - *Vektor von L_i aller beteiligten Prozesse*
  - *Matrixuhren (Raynal und Singhal 1996)*
    - *Eigene Vektoruhr plus Schätzung der Vektoruhren aller anderen Prozesse*

Globaler Zustand

- *Ist eine bestimmte Eigenschaft zu einem bestimmten Zeitpunkt gültig?*
  - *Garbage Collection*
  - *Erkennen von Deadlocks*
  - *Erkennen der Programmterminierung*
  - *Snapshot*
  - *Fehlersuche*
- *Schnitt: Konsistenz bezüglich Happened-Before*
  - *Für jeden Event b der im Schnitt liegt, liegt auch jeder Event a : $a \rightarrow b$ im Schnitt*

Erkennung von Zuständen

- *Garbage, Deadlock, Terminierung "stabile" Zustände*
- *Sicherheit ("safety"): Vom Startzustand aus kann kein unerwünschter Zustand erreicht werden*
  - *z.B. Deadlock*
- *Lebendigkeit ("liveness"): Für jeden vom Startzustand aus erreichbaren Zustand erreiche ich irgendwann einen erwünschten Zustand*
  - *z.B. Terminierung*

Snapshot

- Verteilte Erstellung eines konsistenten Schnittes (Chandy und Lamport 1985)
  - Nicht notwendigerweise zu einem bestimmten Zeitpunkt, aber konsistent
- Inhalt
  - Zustand der Prozesse
  - Zustand der Kommunikationskanäle (welche Nachrichten sind unterwegs)
- Verteilung von Tokens (Markern)

Snapshot: Annahmen

- Kanäle und Prozesse sind zuverlässig
  - Zuverlässige Kommunikation, korrekter Empfang jeder gesendet Nachricht, exactly-once
- Kanäle sind unidirektional und FIFO
- Prozesse und Kanäle sind stark verknüpft
  - Es existiert ein Pfad zwischen jedem beliebigen Paar
- Jeder Prozess kann jederzeit einen Snapshot einleiten
- Weiterarbeit auch während Snapshot

Snapshot: Algorithmus

- Verteilung von Markern
  - Aufforderung an den Empfänger, einen Snapshot einzuleiten
  - Erkennen, wann Nachbarn ihre Snapshots erledigt haben
- Arbeitsweise
  - Snapshot des lokalen Zustandes (falls nicht bereits geschehen)
  - Archivierung aller einkommenden Nachrichten auf allen Kanälen, bis dort ebenfalls ein Marker erscheint

Snapshot: Code

- Marker-Empfang p_i auf Kanal c:
  - p_i hat seinen Zustand noch nicht gesichert:
    - Prozesszustand jetzt sichern
    - Zustand von c ist {}
    - Beginne Log aller anderen einkommenden Kanäle
    - Sende je einen Marker über alle Kanäle
  - Sonst:
    - Log auf Kanal c ist jetzt vollständig