Verteilte Systeme

Prof. Dr. Martin Becke

CaDS - HAW Hamburg

Version 0.9

Koordination

Algorithmen

- Einzelne Komponenten agieren unabhängig
- Dennoch gemeinsames Ziele
- Verschiedene Ansätze und Techniken

Koordination

Waves

- Klasse von Algorithmen in der Graphentheorie
- Ziel ist Verteilung von Informationen
- Es wird nicht nur zur Zustandsspeicherung genutzt, sondern auch es kann Aktionen triggern, oder Informationen an die Nachricht anhängen
- Exisitieren viele Varianten
  - Wellen gleichzeitig
  - Besondere Ausbreitung
  - Spezielle Techniken der Optimierung

Waves

Grundsätzlicher Ablauf

- Ein Startknoten sendet eine Welle an alle seine Nachbarn.
- Jeder Knoten, der die Welle erhält, führt eine bestimmte Aktion aus (zum Beispiel aktualisiert er einen Zähler oder speichert eine Information) und sendet die Welle an alle seine Nachbarn, die die Welle noch nicht erhalten haben.
- Dieser Prozess wiederholt sich, bis alle Knoten im Graphen die Welle erhalten und verarbeitet haben.

Waves

Anwendung

- Routing
- Synchronisation
- Identifikation von Ressourcen
- ...

Waves

Spezielle Klassen

- Traversal
- Tree
- Echo
- ...

Koordniation

Snapshot

- Erfassung eines globalen Zustands
- Ziel konsistenter Zustand
- Ein Schnappschuss erstellen und koordinieren
- Verschiedene Ansätze, beispielhaft
  - Chandy-Lamport-Algorithmus
  - Lai-Yang-Algorithmus
  - ...

Snapshot

Chandy-Lamport-Algorithmus

- Initiator speichert Zustand und sendet Marker-Nachrichten an direkte Kommunikationspartner
- Knoten erhält Marker
  - Neuer Marker : Speichert Zustand und sendet Marker aus, speichert alle Nachrichten von jedem Kanal, außer von dem initiierenden
  - Bekannter Marker : stoppt Aufzeichnung von Nachrichten von diesem Kanal
- Schnappschuss besteht aus dem gespeicherten Zustand jedes Prozesses und den gespeicherten Nachrichten
- Der Zustand kann eingesammelt werden und die Zustände entsprechen der Happens-before Relation

Snapshot

Lai-Yang-Algorithmus

- Farbiger Algorithmus
- Prozesse und Nachrichten sind Rot oder Weiß
  - Bei Initialisierung alle weiß
  - Bei Start kann beliebiger Knoten beginnen und Farbe auf Rot setzen (Speichert Zustand und sendet rote Nachrichten)
  - Bei roter Nachricht, Zustandsänderung auf Rot, Speicher Zustand und sendet rote Nachrichten
  - Terminiert, wenn alle Knoten rot sind und keine weißen Nachrichten mehr im System.
- Keine Synchronisation des Startpunktes notwendig
- Hoher Ressourcenverbrauch für das Tracking

Snapshot

Einsatz

- Deadlock-Erkennung
- Eventuelle Konsistenz
- Monitoring und Debugging
- Simulation von verteilten Systemen
- Konsistenz in verteilten Datenbanken
- Globales Checkpointing
- ...

Koordniation

Checkpointing

- Erfassung eines globalen Zustands
- UND das System zu einem konsistenten Zustand zurückzusetzen
- Möglicher Einsatz bei Fehlern im System
- Verschiedene Ansätze, beispielhaft
  - Peterson-Kearns Rollback Recovery-Algorithmus

Checkpoint

Peterson-Kearns Rollback Recovery-Algorithmus

- Jeder Prozess im System führt in regelmäßigen Abständen ein lokales Checkpointing durch
- Zusätzlich zum Checkpointing protokolliert jeder Prozess alle gesendeten und empfangenen Nachrichten
- Bei Fehler wird Prozess und abhängige Prozesse zurückgesetzt
- Nach dem Rollback werden alle Nachrichten, die nach dem letzten Checkpoint erneut gesendet
- Fortsetzung

Koordniation

Deadlock Detection

- Deadlock : Zwei oder mehr Prozesse warten unbegrenzt
- Situationen zu identifizieren
- und gegebenenfalls zu beheben
- Idee : Ressourcenzuteilungsgraphen des Systems überprüfen

Deadlock Detection

Ressourcenzuteilungsgraphen

- Graph besteht aus Knoten, die Prozesse und Ressourcen darstellen
- Kanten stellen die Beziehungen zwischen diesen Elementen da
  - « Prozess P besitzt Ressource R »
  - « Prozess P wartet auf Ressource R »
- Ein Deadlock tritt auf, wenn es in diesem Graphen einen Zyklus gibt

Deadlock Detection

Grundsätzliche Verfahren

- Wait-for-Graphs-Algorithmus
  - Setzt voraus, dass genaue und aktuelle Informationen über alle Prozesse und Ressourcen im System verfügbar sind
  - Nicht fehlertolerant
- Konsensbasierte Algorithmen
  - Unvollständige oder verzögerte Informationen über das System
  - Fehlertolerant