# Verteilte Systeme

Prof. Dr. Martin Becke

CaDS - HAW Hamburg

Version 0.9

# Inhalt

# Allgemein

## Probleme für Algorithmen

- Effizienz
- Konsensprobleme
- Koordinationsprobleme
- Replikations- und Konsistenzprobleme
- Fehlertoleranz- und Wiederherstellungsprobleme
- Lastverteilungs- und Ressourcenverwaltungsprobleme

# Paralleles Rechnen

## Parallele Algorithmen

- Leicht : « embarrassingly parallel »
  - Matrixoperationen
  - Sortieralgorithmen
  - Grafikverarbeitung
  - ...
- Unterstützt durch Frameworks (z.b. MapsReduce [1])

1. Siehe Script

# Paralleles Rechnen

## Parallele Algorithmen

- Nicht alle Algorithmen gut für die Parallelisierung geeignet
- Beispiel ggT
- Strategie kann sein Granularität zu ändern

# Algorithmenstrukturn

## Koordination

- Leistung
- Zuverlässigkeit
- Konsistenz
- Sicherheit

One-to-Many-Kommunikation, Baumstrukturen, Flooding oder Gossip :

- Zentralisierte Algorithmen
- Dezentralisierte Algorithmen

# Algorithmen

## Konsens

- Viele Systeme auf hochverfügbare und korrekte Systeme angewiesen
- Einzelne Kerne, CPUs oder Systeme dem nicht gewachsen
- Redundanz hilfreich ? Zwingend Zusammenarbeit/ Kooperation !

# Konsens

## Problem

- Es existiert eine Gruppe
- Teilnehmer kommunizieren mit Nachrichten
- Teilnehmer oder Nachrichten können fehlerhaft sein

# Konsens Fallbeispiel

## Anwendungsbeispiel

- Verteiltes System von zwei Ampeln, das den Verkehr an einer Baustelle reguliert
- Jede Ampel arbeitet autonom
- Koordiniert sich mit anderer Ampel über Nachrichten

# Konsens

## Anforderungen

- Einigkeit
- Integrität
- Terminierung
- Validität

# Konsens

## Follow the Leader

- Ein Knoten wird zu Leader gewählt
- Reduziert Komplexität
- Paxos nutzt alternativen Multi-Leader Ansatz
- P2P Architekturen können auch Leaderless funktionieren

# Konsens

## Bekannte Lösungen

- Paxos (Mutter)
- RAFT (Einfacher)
- ZAB (Zookeeper)

# Konsens

## Paxos

- Leslie Lamport 1990
- Mutter aller Konsens-Algorithmen
- Arbeitet mit Rollen : Proposer oder Acceptors
- Berüchtigt für seine Komplexität

2

2. Weitere Details im Script

# Konsens

## Raft

- Diego Ongaro und John Ousterhout (2013/2014)
- Alternative zu komplexen Paxos
- Konzentriert sich auf drei Aufgaben
  - Leader Election
  - Log Replication
  - Safety

3

3. Weitere Details im Script

# Konsens

## ZAB

- Speziell für ZooKeeper entworfen und optimiert
- Zuverlässige und geordnete Zustandsaktualisierung in einer Replikationsgruppe
- Im Wesentlichen ein Zwei-Phasen-Commit-Protokoll mit einer Führungsfindungsphase
- Toll für Namensdienst, Konfigurationsmanagement, etc
- Aber schlecht auf großen Datenmengen, bei vielen Schreibvorgängen, etc

4

4. Weitere Details im Script

# Konsens

## Konsens mit Crash Failures

- Unmöglichkeit von 1-Crash-Konsens (fundamentale Beweis 1985)
- Als FLP-Unmöglichkeitstheorem bezeichnet
- Unterscheidung Nachricht verzögert oder Empfänger abgestürzt

Für ein asynchrones verteiltes System mit nur einem möglicherweise fehlerhaften Prozess (d.h. einem Prozess, der abstürzen kann) kann es keinen deterministischen Algorithmus geben, der immer zu einem Konsens führt [5]

5. Ein asynchrones System ist ein System, in dem es keine festen Obergrenzen für die Zeit gibt

# Konsens

## FLP-Unmöglichkeitstheorem

- Teilsynchrone Systeme (Asynchron ist kaum durchzuhalten)
- Randomisierte Algorithmen (Randomized Consensus Algorithmen-Gruppe)
- Praktische Konsensalgorithmen (Paxos, Raft oder ZAB)
- Konsens durch teilweise Synchronität
  - Bracha-Toueg Crash Consensus
  - Chandra-Toueg Unreliable Failure Detectors

# Konsens

## Unreliable Failure Detectors

- Fehlerdetektoren müssen nicht perfekt sein (False positives)
- Beispiel ist : Timeout-basierter Fehlerdetektoren
- Alternativen sind Heartbeat-Mechanismus, Quorum-basierte Ansätze

# Konsens

## Konsens mit Byzantine Failures

- Basis weiter Byzantinische Generäle Problem
- Mehr als zwei Drittel der Knoten müssen korrekt sein
- Möglicher algorithmischer Ansatz Practical Byzantine Fault Tolerance (PBFT)