Lernzettel

Verteilte Systeme: Kommunikationsmodelle, Replikation, Konsistenzmodelle, Failover und Verfügbarkeit

Universität:	Technische Universität Berlin
Kurs/Modul:	Technische Grundlagen der Informatik
Erstellungsdatum:	September 19, 2025

Lernzettel: Verteilte Systeme: Kommunikationsmodelle, Replikation, Konsistenzmodelle, Failover und Verfügbarkeit

(1) Überblick. Verteilte Systeme bestehen aus mehreren Rechnern, die zusammenarbeiten, um eine Aufgabe zu erfüllen. Typische Herausforderungen sind Kommunikation, Koordination, Replikation von Daten, Konsistenz, Fehlererkennung und Failover.

(2) Kommunikationsmodelle.

- Nachrichtenaustausch (Message Passing): Sendern senden Nachrichten zu Empfängern, oft asynchron.
- Remote Procedure Call (RPC): Aufruf einer entfernten Funktion wie eine lokale Funktion; führt zu Synchronität oder Asynchronität je nach Implementierung.
- Publish/Subscribe: Produzenten publizieren Events, Konsumenten abonnieren Themen; Entkopplung über einen Broker.
- REST/HTTP und gRPC: Web-basierte oder binary RPC-Transportwege; oft synchron, aber asynchrone Muster möglich.
- Middleware und Messaging-Systeme: z. B. JMS, Kafka, RabbitMQ; unterstützen Persistenz, Acknowledgement und Delivery- Guarantees.

(3) Replikation.

- Zweck: Verfügbarkeit, Skalierung von Lesezugriffen und Redundanz.
- Replikationstypen:
  - Synchrone Replikation: Schreiboperationen werden bestätigt, wenn alle oder eine Mehrheit der Replikate aktualisiert sind; höhere Latenz, stärkere Konsistenz.
  - Asynchrone Replikation: Schreiben wird sofort bestätigt; Verzögerung bei der Replikation kann zu Inkonsistenzen führen.
- Architekturen:
  - Primär-/Sekundär (Master/Slave): Ein primäres Repository, mehrere Sekundärkopien.
  - Multi-Master (Leaderless): Mehrere Knoten können schreiben; Konfliktlösung nötig.
- Konfliktlösung: Versionierung, Vector Clocks, letzte Schreiberpriorität, oder Anwendungslogik.

(4) Konsistenzmodelle.

- Starke Konsistenz (Linearizability): Jede Operation wirkt sofort global konsistent; einfache Programmierung, höhere Latenz.
- Eventual Consistency: Replikate konvergieren irgendwann; hohe Verfügbarkeit und niedrige Latenz, aber zeitweise Inkonsistenz.
- Kausale Konsistenz: Ereignisse respektieren Ursache-Wirkung-Beziehungen; Konvergenz mit kausaler Ordnung.

- Bounded Staleness / Read-Your-Writes: Beschränkte Verzögerung oder konsistente Leseergebnisse nach einer Schreiboperation.
- CAP-Theorem-Bezug: In Netzwerken mit Partitionen kann kein System stark konsistent und hochverfügbar zugleich bleiben; Kompromisse je nach Anwendung.

(5) Failover und Verfügbarkeit.

- Failover-Strategien:
  - Hot Standby: Schnelles Umschalten auf einen dauerhaft bereiten Knoten.
  - Warm Standby: Bereits initialisiert, aber geringfügige Verzögerung beim Umschalten.
  - Cold Standby: Neuer Start notwendig; längste Reaktionszeit.
- Mechanismen:
  - Heartbeats, Timeouts und Leader Election.
  - Konsensusprotokolle (Paxos, Raft) zur Bestimmung eines gemeinsamen Zustands.
  - Quorum-basierte Entscheidungen (Mehrheitsbildung) bei Schreib-/Lesezugriffen.
- Verfügbarkeit und Metriken:
  - MTBF, MTTR, Verfügbarkeitsziele (Uptime) und SLA.
  - Tradeoffs zwischen Konsistenz, Verfügbarkeit und Partitionstoleranz (CAP).

(6) Fazit und Ausblick. Verteilte Systeme bieten Skalierbarkeit und Robustheit, erfordern jedoch sorgfältige Abwägungen zwischen Konsistenz, Verfügbarkeit und Kommunikationslatenz. Architekturen und Protokolle sollten an die konkreten Anwendungsziele angepasst werden.