Technische Hochschule Ingolstadt
Fakultät für Elektrotechnik und Informatik

Zukunft in Bewegung

IT-Integrations- und Migrationstechnologien

Fehlerbehandlung in verteilten Systemen

Prof. Dr. Bernd Hafenrichter 26.11.2023

Motivation

- Fehlerfälle sind inhärent im Design von verteilten/integrierten Systemen enthalten
- Auch bei der Verwendung von Middleware-Technologien ist es notwendig über das Fehlerhandling nachzudenken.

- Partielle Ausfälle: eine Komponente fällt aus, wodurch ein Teil des Systems beeinträchtigt werden kann
- Ein wichtiges Ziel beim Entwurf verteilter Systeme: sie so aufzubauen, dass sie nach partiellen Ausfällen automatisch wiederhergestellt werden können
- Insbesondere sollte das System im Falle eines Fehlers akzeptabel weiterarbeiten, d. h. Fehler tolerieren

Motivation

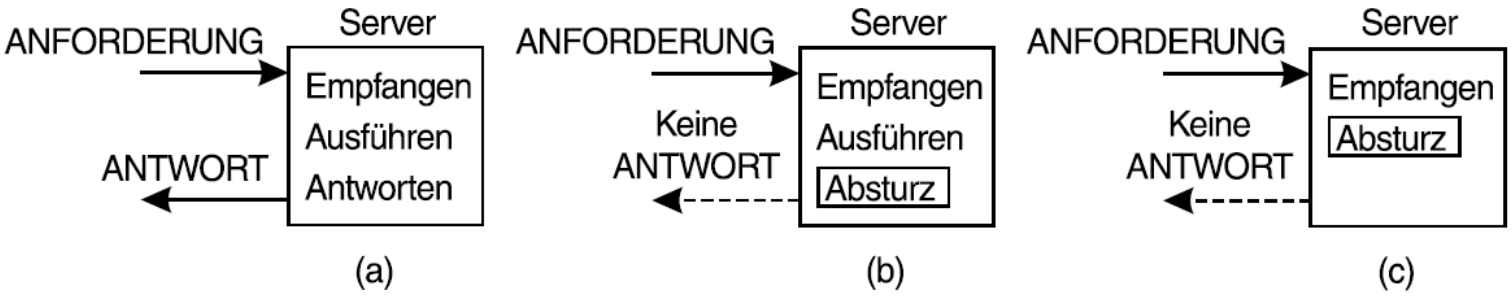
Mögliche Fehlertypen in verteilten Systemen

Fehlertyp	Beschreibung
Absturzfehler	Ein Server wurde unterbrochen, hat aber bis zu diesem Zeitpunkt korrekt gearbeitet
Auslassungsfehler	Server reagiert nicht auf Anfragen
Empfangsauslassung	Ein Server reagiert nicht auf eingehende Anforderungen
Sendeauslassung	Server sendet keine Nachrichten
Timing-Fehler	Die Antwortzeit eines Servers liegt außerhalb eines festgelegten Zeitintervalls
Antwortfehler	Die Antwort des Servers ist falsch
Wertfehler	Der Wert der Antwort ist falsch
Statusübergangsfehler	Der Server weicht vom korrekten Steuerfluss ab
Zufälliger Fehler/Byzantinischer	Ein Server erzeugt zu zufälligen Zeiten zufällige Antworten

Kommunikationsfehler

Grundsätzliches Problem bei verteilter Kommunikation:

- Ein Clients kann die Situationen (b) und (c) in Abb. nicht unterscheiden kann



Kommunikationsfehler – Absturz des Servers

- Beispiel: Ein Client möchte auf dem Server eine Funktion aufrufen die ein Bestätigungs-EMail verschickt.

- Randbedingungen Server (Strategie zum Senden der Antwortnachricht)
  - bevor er die Mail verschickt
  - nach dem verschicken der Mail

Kommunikationsfehler – Absturz des Servers

- Randbedingungen Client
  - Der Client wird über einen Absturz informiert, hat aber keine Information über den Status der Operation
  - Strategien im Fehlerfall:
    - niemals eine neue Anforderung absetzen
    - immer eine neue Anforderung absetzen
    - neue Anforderung wenn keine Bestätigung der Auslieferung
    - neue Anforderung wenn eine Bestätigung der Auslieferung

Kommunikationsfehler – Absturz des Servers

- Ereignisse: Senden der Durchführungsnachricht (M), EMail versenden (P), Absturz/Crash (C)
- Klammern: Ereignis passiert nicht, da bereits Absturz
- Fazit aus der Tabelle: Es gibt keine Kombination der Strategien, die für alle mögliche Abfolgen korrekt funktioniert

Client	Server					
	Strategie M→P			Strategie P→M		
Strategie zum erneuten Absetzen	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
Immer	DUP	OK	OK	DUP	DUP	OK
Nie	OK	NULL	NULL	OK	OK	NULL
Nur nach Bestätigung	DUP	OK	NULL	DUP	OK	NULL
Nur, falls keine Bestätigung erfolgt ist	OK	NULL	OK	OK	DUP	OK

OK = Mail wird einmal versandt, DUP = Mail wird zweimal versandt, NULL = Mail wird nicht versandt

Fehlerbehandlung in verteilten Systemen

Fehlermaskierung durch Redundanz:

Fehlertolerantes System muß Fehler vor anderen Prozessen Verbergen

Wichtigste Technik dazu: Redundanz

- Informationsredundanz: zusätzliche Prüfbits (z.B. CRC)
- zeitliche Redundanz: Wiederholung fehlerhafter Aktionen
- physische Redundanz: mehrfaches Vorhalten wichtiger Komponenten

Behandlung von Kommunikationsfehlern

Ausgangssituation:

Ein Client sendet einen Nachricht an einen Server und wartet auf einen Ergebnisnachricht. Diese bleibt jedoch aus

Einfache Lösung: nach Timeout die Anforderung wiederholen

Problem: Vielleicht ist der Server einfach zu langsam?

Risiko: Doppelte Ausführung möglich

Grundsätzlich können zwei Arten von Funktionen unterschieden werden

- Funktionen die den Status des Servers ändern (z.B. abbuchung)
- Funktionen die den Status des Servers nicht ändern (z.B. lesen eines Kontostands)

Behandlung von Kommunikationsfehlern

Definition Idempotenz:

- Als idempotent bezeichnet man Funktionsaufrufe, die immer zu den gleichen Ergebnissen führen, unabhängig davon, wie oft sie mit den gleichen Daten wiederholt werden. Idempotente Arbeitsgänge können zufällig oder absichtlich wiederholt werden, ohne dass sie nachteilige Auswirkungen auf den Computer haben.

Behandlung von Kommunikationsfehlern

Realisierung der Idempotenz:

- Jeder Methodenaufruf bzw. Nachrichtenaustausch wird mit einer eindeutigen Nummer versehen
- Der Server beobachtet die Nachrichten jedes Clients und verweigert wiederholte Ausführung gleicher Anforderung
- Aufwendig: der Server muss für jeden Client eine Administration bereitstellen

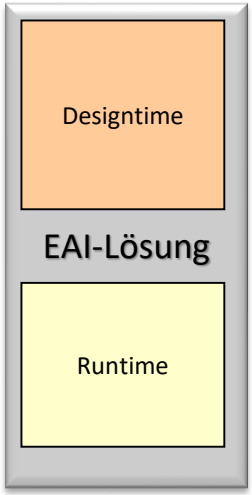
- Noch Besser: Operationen per Definition als Idempotenz definieren. Dadurch kann der Verwaltungsaufwand evtl. reduziert werden.

Behandlung von Kommunikationsfehlern über Middleware Technologie

- Für jede remote Operation sollte über einen Quality of Service nachgedacht werden:

Typ	Reaktion	Filterung von Duplikaten	Beschreibung
At-least-once	wiederholen	Nein	Die entfernte Prozedur wird bei einem empfangenen Duplikat wiederholt ausgeführt
at-most-once	wiederholen	Ja	Duplikate werden gefiltert, entweder komplette Ausführung des Auftrags, oder Fehlermeldung
exactly-once	wiederholen	Ja	Duplikate werden ebenfalls gefiltert. Weiterhin wird auch bei Ausfall des Systems die Ausführung des Auftrags über den Wiederanlauf hinaus gewährleistet.
Maybe	Nein	Nein	

Nicht-funktionale Anforderungen



Zuverlässigkeit & Robustheit

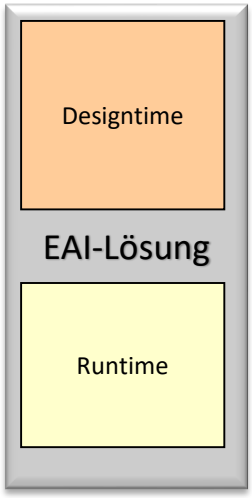
Fehlerhandling

- Konfigurations-Fehler
- Laufzeit-Fehler (permanent, sporadisch)
  - Kommunikationsfehler (falsches Datenformat, falscher Datentyp)
  - Verbindungs- und Netzwerkfehler (Empfänger nicht erreichbar)
- Absturz/Beenden der gesamten Integrationsinstanz

Reaktion auf Fehler

- Konfigurations-Fehler:
  - Erkennen der Fehler beim Start der Schnittstelle. Start abbrechen & Meldung ausgeben
- Absturz/Beenden:
  - Neustart & Recovery im letzten gültigen Systemzustand
- Verbindungs- & Netzwerkfehler:
  - Wiederholtes Ausführen der Aktion innerhalb definierter Grenzen
- Kommunikationsfehler
  - Aktuellen Datensatz suspendieren. Manueller Eingriff des Operators notwendig

Nicht-funktionale Anforderungen



Quality-of-Service (QoS)

Exactly-Once

- Garantierte Zustellung der Daten
- Zustellung erfolgt exakt einmal
- Persistente Speicherung des Verarbeitungszustandes ist notwendig
- Optimal für Batch-Betrieb

Best-Effort

- Keine Wiederholung bei Fehlern
- Fehler werden an Aufrufer zurückgegeben
- Ideal für Dialog-Betrieb