EPM
Elastic Persistent Memory

Florian Klein

Institut für Informatik
Abteilung Betriebssysteme
Heinrich-Heine-Universität Düsseldorf

12.06.2012

Gliederung

Zielsetzung und Eigenschaften

Zielsetzung

- Alle Daten permanent im RAM
- Gute Fehlertoleranz (Knotenausfälle, Netzwerkpartitionierung, etc.)
- Skalierbar (mehrere tausend Knoten)

Eigenschaften

- variable Konsistenz
- verteilte MetaDaten-Verwaltung
- Logging und FastRecovery
- konfigurierbar bzgl. des CAP-Dilemma

Einsatzmöglichkeiten und Anwendungsbeispiele

Einsatzmöglichkeiten

- Rechenzentren
- Cloud-Computing
- große Anwendungen mit hohem Datendurchsatz

Anwendungsbeispiele

- verteiltes Dateisystem über Fuse-Anbindung
- Tabellenverwaltung (ähnlich zu BigTable)
  - ACID-Kriterien über mehrere Zeilen
- Cache
  - Unterstützung von Timeouts und Read-Only-Einträgen
- Data-Store

Gliederung

Konsistenz und Replikation

unterstützte Konsistenzmodelle

- schwache Konsistenz
- starke Konsistenz
- ggf. transaktionale Konsistenz

Replikation

- feste oder dynamische Replikatanzahl
- Unterstützung von Erasure-Codes
- CAP-Dilema-Eigenschaften konfigurierbar

Fehlertoleranz und MetaDaten-Verwaltung

Fehlertoleranz

- Maskieren von Knotenausfälle
- Maskieren von Netzwerkpartitionen
- Verschmelzung von Netzwerkpartitionen
- lokales und entferntes Logging
- FastRecovery

MetaDaten-Verwaltung

- verteilt in Form einer DHT
- lokaler Cache

Daten-Verwaltung

Daten-Verwaltung

- in Form von Binärblöcken variabler Größe (max. 100 MB)
- Jeder Binärblock hat eine eindeutige ID (512 Bit)
- Event-Service (Update / Invalidate)
- PubSub-Service

Schnittstelle

create	erzeugt einen neuen Datenblock mit einer unbenutzen ID
get	Holt einen Datenblock (synchron oder asynchron))
put	Speichert bzw. Aktualisiert einen Datenblock
lock	Sperre auf einen Datenblock setzen (freiwillig oder advisory)
remove	löscht einen Datenblock

Gliederung

Aufbau

Kern

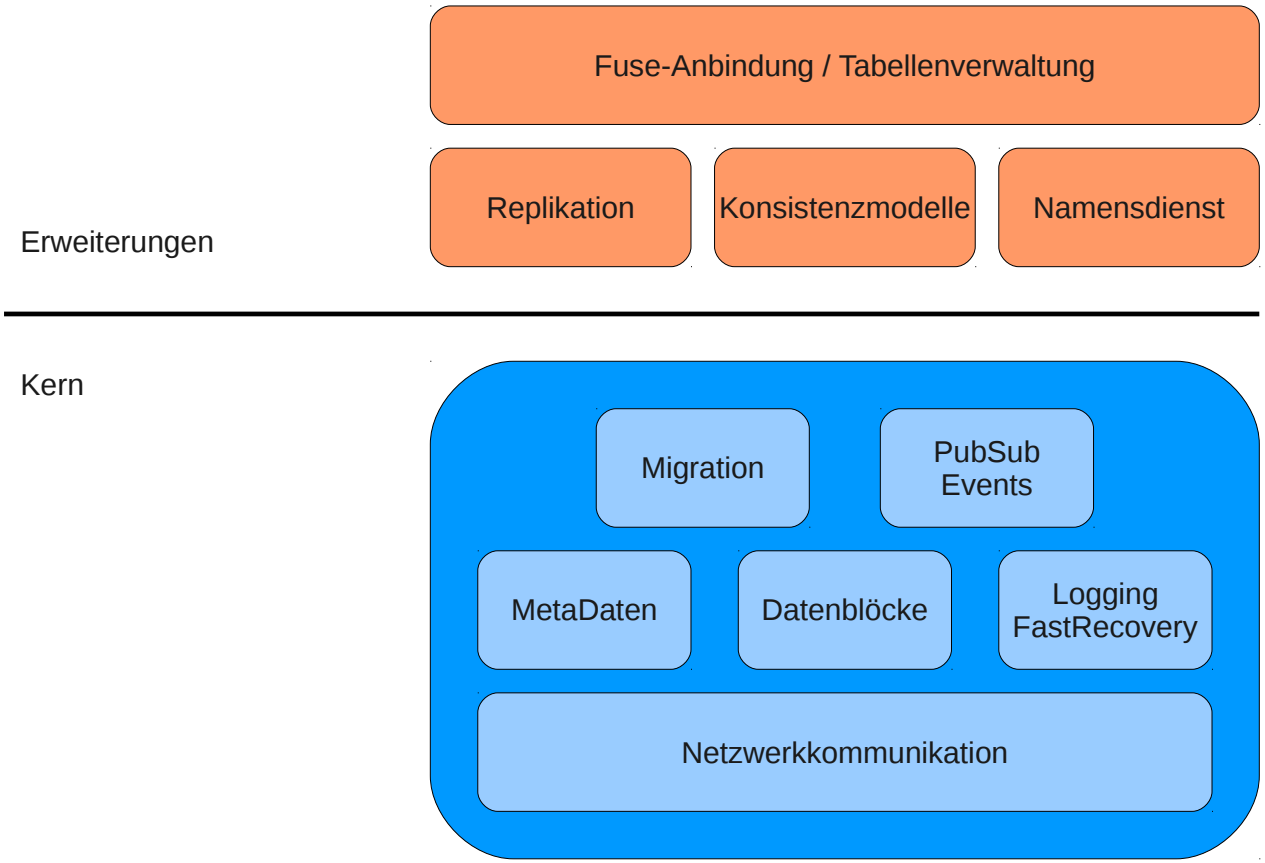
- Verwaltung der Datenblöcke und MetaDaten
- Datenmigration
- PubSub- und Event-Service
- Logging und FastRecovery
- Netzwerkkommunikation

Erweiterungen

- Replikation und Konsistenzmodelle
- Namensdienst
- Tabellenverwaltung
- Fuse-Anbindung

Aufbau

Überblick



Gliederung

Datenblock-Verwaltung

Struktur eines Datenblocks

- Eindeutige 512-Bit ID
- Binärdaten in Form eines byte-Arrays

Eigenschaften

- variable Größe der Datenblöcke (konfigurierbare maximal Größe)
- Eindeutige ID wird über einen Hash-Algorithmus erzeugt
- Synchrone und asynchrone Lesezugriffe
- Streaming API um unnötige Wartezeiten zu umgehen
- Lock-Mechanismus (freiwillige Sperre oder Advisory-Lock)

Datenblock-Verwaltung

Implementierung I

create - Erzeugt eine neue BlockID

- konfigurierbarer Hash-Algorithmus (default: SHA-512)
- Basis ist die lokale IP-Adresse und die aktuelle Zeit in Nanosekunden

get - Holt einen Datenblock anhand einer BlockID

1. Prüfe den lokalen Speicher
   a) Datenblock vorhanden $\rightarrow$ Bei Schritt 5 fortfahren
   b) Datenblock nicht vorhanden $\rightarrow$ Bei Schritt 2 fortfahren
2. Über die MetaDaten-Verwaltung den Ziel-Rechner finden
3. Anfrage an Ziel-Rechner stellen
4. Antwort des Ziel-Rechners abwarten $\rightarrow$ Datenblock entgegen nehmen
5. Datenblock zurückgeben

Datenblock-Verwaltung

Implementierung II

put - Speichert einen Datenblock

1. Prüfe den lokalen Speicher
   a) Datenblock vorhanden → Datenblock ersetzen
   b) Datenblock nicht vorhanden → Bei Schritt 2 fortfahren
2. Über die MetaDaten-Verwaltung den Ziel-Rechner finden
   a) lokale Rechner → Datenblock speichern (neuer Datenblock)
   b) anderer Rechner → Nachricht mit dem Datenblock an den Ziel-Rechner schicken

remove - Löscht einen Datenblock

1. Prüfe den lokalen Speicher
   a) Datenblock vorhanden → Datenblock löschen
   b) Datenblock nicht vorhanden → Bei Schritt 2 fortfahren
2. Über die MetaDaten-Verwaltung den Ziel-Rechner finden
3. Nachricht an den Ziel-Rechner schicken

Datenblock-Verwaltung

Implementierung III

lock - Sperrt einen Datenblock

1. Prüfe den lokalen Speicher
   a) Datenblock vorhanden → Datenblock sperren und bei Schritt 5 fortfahren
   b) Datenblock nicht vorhanden → Bei Schritt 2 fortfahren
2. Über die MetaDaten-Verwaltung den Ziel-Rechner finden
3. Anfrage an Ziel-Rechner stellen
4. Antwort des Ziel-Rechners abwarten → Datenblock entgegen nehmen
5. Datenblock zurückgeben

Die Sperre wird freigegeben, sobald der Datenblock mit `put` gespeichert wird.

MetaDaten-Verwaltung

Aktueller Stand

- ZooKeeper als zentrale Instanz
- ObjektID wird auf IP + Port des Zielrechners gemapped
- Verteilter Lock-Mechanismus

Aussicht

- Verteilte Verwaltung mit Hilfe einer DHT
- lokaler Cache für Anfragen

Migration

Aktueller Stand

- in die Datenblock-Verwaltung eingebunden
- Ablauf:
  1. Datenblock an den Ziel-Rechner übertragen
  2. MetaDaten-Verwaltung benachrichtigen
  3. lokalen Datenblock löschen
  4. PubSub-Service wird aktuell nicht informiert

Aussicht

- Eigenständige Komponente
- Datenblock während der Migration sperren
- automatische Lastverteilung durch Migrations-Algorithmus
- AKtualisierung des PubSub-Service

PubSub- und Event-Service

Aktueller Stand

- in die Datenblock-Verwaltung eingebunden
- mögliche Subscribearten: Update, Invalidate
- lokale Verwaltung für jeden gespeicherten Datenblock
- Bei einer Aktualisierung wird zu jedem der Subscriber eine Verbindung aufgebaut und eine Nachricht verschickt

Aussicht

- Eigenständige Komponente
- Gruppenkommunikation (beispielsweise jgroups)
- verteilte Verwaltung
- evtl. sogar aus dem Kern raus verlagern

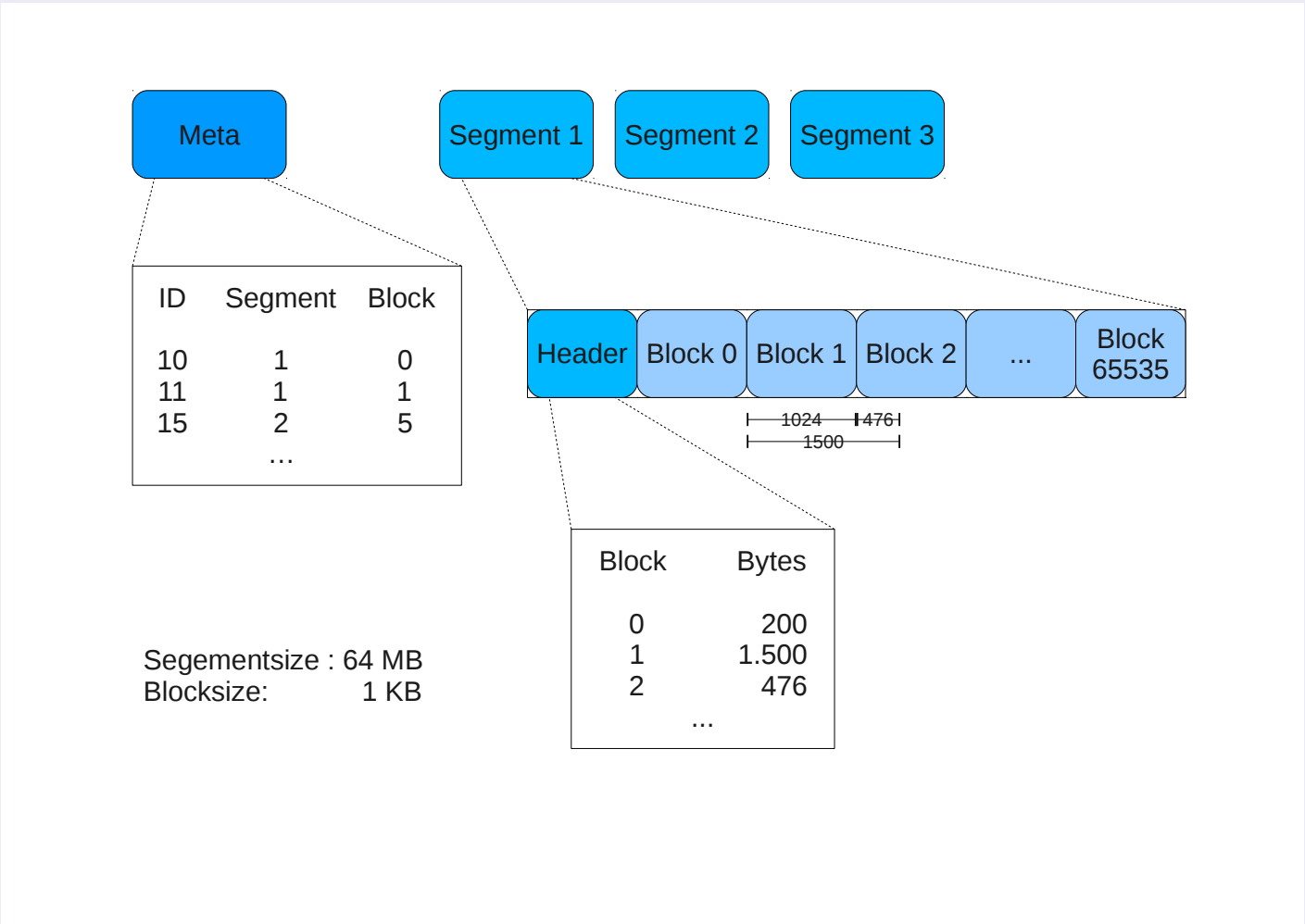
Logging

Eigenschaften

- Lokales Log
- Zusätzliche Backup-Logs auf weiteren Rechnern (default 3)
- Restart von lokalem Log möglich
- Log ist unterteilt in Segmente
  - Segmente sind wiederum unterteilt in Blöcke
  - Verwaltung der Segmente erfolgt mit Hilfe iner Hash-Tabelle
- zusätzlich zu den Datenblöcken wird eine Versionsnummer pro Datenblock gepflegt

Logging

Struktur



Logging

Funktionsweise I

Neuen Datenblock speichern

1. Finde eine Anzahl von zusammenhängenden Blöcken mit der Größe $\geq$ Größe des Datenblocks
   - Ist dies nicht möglich, wird ein neues Segment erstellt
2. Speicher den Datenblock in den ausgewählten Blöcken (Versionsnummer = 1)
3. Aktualisiere den Segement-Header
4. Aktualisiere die Hash-Tabelle mit den MetaDaten

Datenblock löschen

1. Lösche die entsprechenden Einträge im Segment-Header für jeden belegten Block
2. Aktualisiere die Hash-Tabelle mit den MetaDaten

Logging

Funktionsweise II

Datenblock ändern

1. Vergleiche die Größe des gespeicherten Datenblocks mit der neuen Größe
   a) die Größe ist gleich $\rightarrow$ die Blöcke mit den neuen Daten überschreiben
   b) die Größe hat sich verringert $\rightarrow$ die Blöcke mit den neuen Daten überschreiben und ggf. leere Blöcke freigeben
   c) die Größe hat sich erhöht $\rightarrow$ benachbarte Böcken überprüfen
      a) Genügend freie Blöcke vorhanden $\rightarrow$ die Daten in den Blöcken speichern
      b) Nicht genügend freie Blöck vorhanden $\rightarrow$ Die bisher belegten Blöcke freigeben und einen neuen Speicherplatz suchen (siehe neuen Datenblock speichern)
2. Versionsnummer inkrementieren
3. Aktualisiere den Segement-Header
4. Aktualisiere die Hash-Tabelle mit den MetaDaten

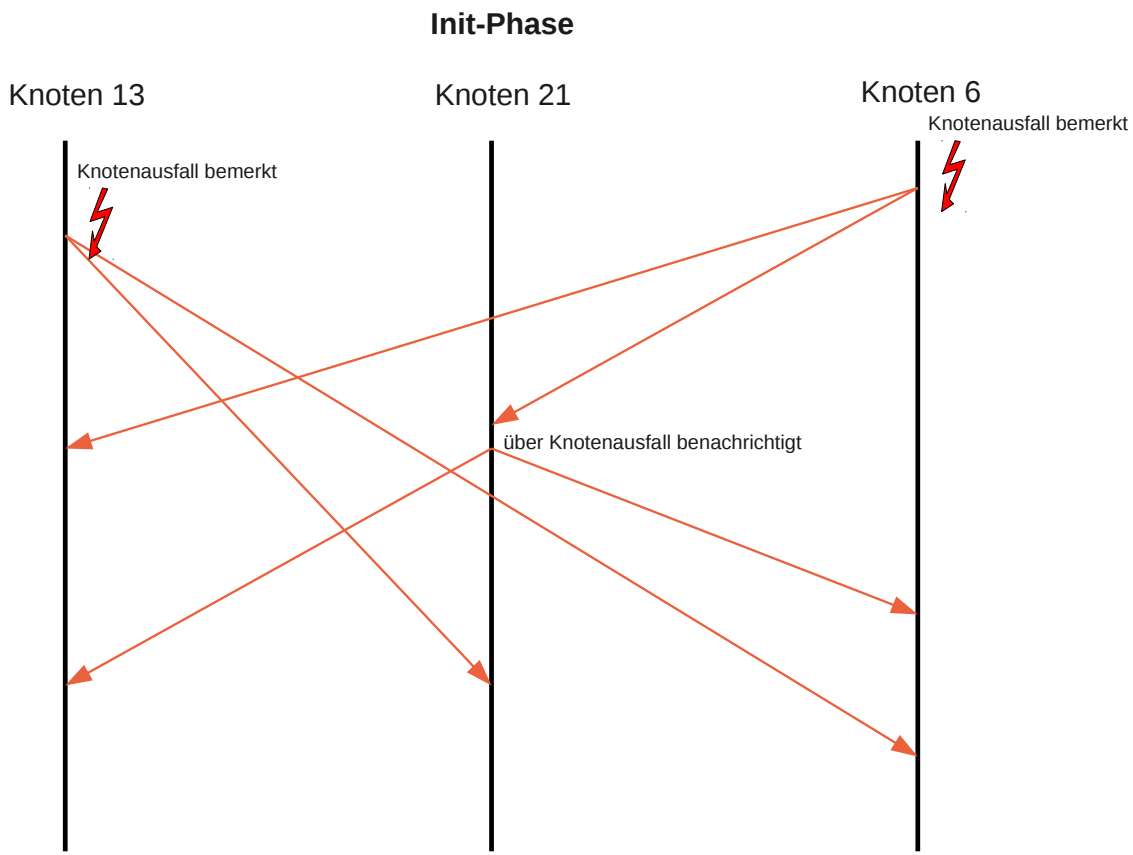
FastRecovery

Ablauf

1. ein Knoten fällt aus
2. einer der Backup-Knoten bemerkt den Ausfall und benachrichtigt die anderen Backup-Knoten
3. die Backup-Knoten initialisieren den Recovery-Algorithmus (Init-Phase)
4. die Backup-Knoten teilen die wiederherzustellenden Datenblöcke untereinander auf (Divide-Phase)
5. die Datenblöcke werden lokal wiederhergestellt (Recover-Phase)

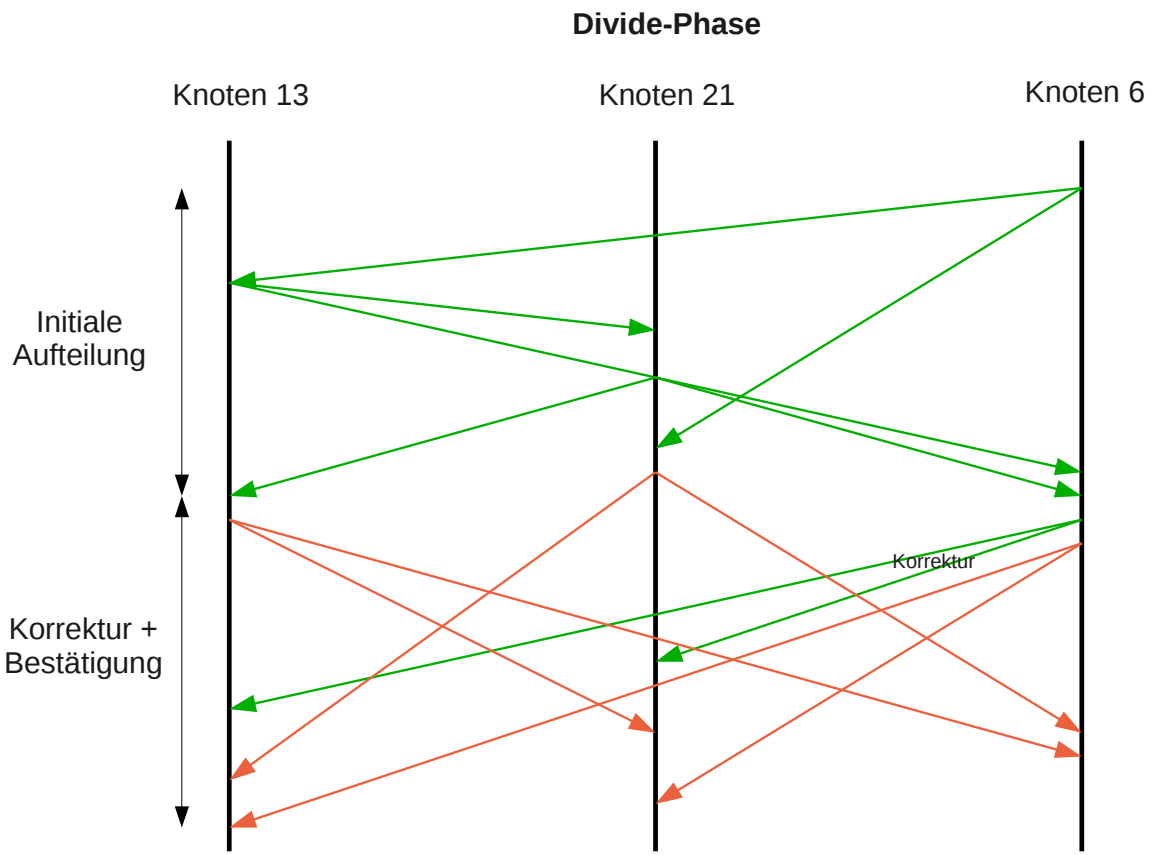
FastRecovery

Init-Phase



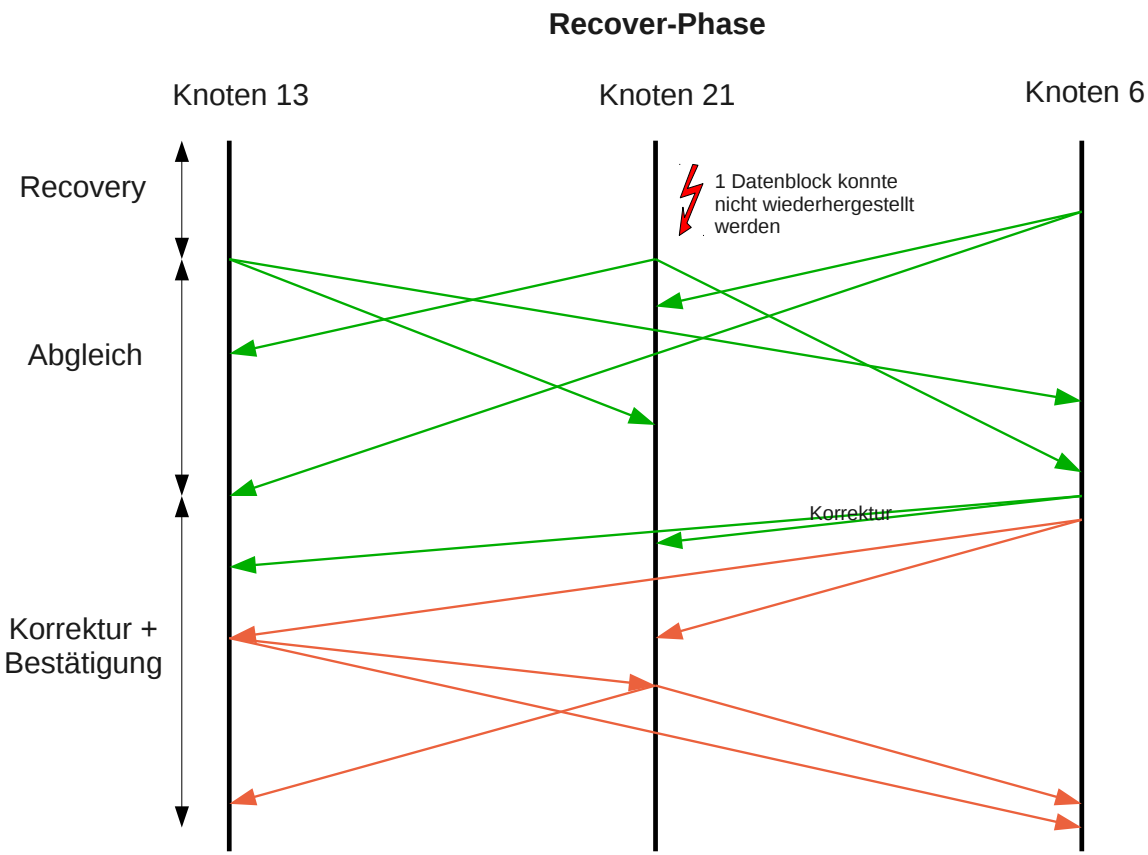
FastRecovery

Divide-Phase



FastRecovery

Recover-Phase



Netzwerk

Eigenschaften

- Java NIO
- TCP Sockets
- drei Nachrichtentypen: Message, Request und Response
- automatische Neuübertragung eines Requests nach einem Timeout
- Event-Listener-Model:
  - `MessageReceiver` werden über eingehende Nachrichten informiert
  - `RequestReceiver` werden über eingehende Requests und Responses informiert
  - `ConnectionLostReceiver` werden über unterbrochene Verbindungen informiert
- Aufbewahrung von nicht behandelten Nachrichten

Danksagung

Vielen Dank für Ihre Aufmerksamkeit

Für Fragen stehe ich gerne zur Verfügung