

Vortrag

Zuverlässige Koordinierung in Cloud-Systemen

Florian Heisig

florian.heisig@informatik.stud.uni-erlangen.de

Seminar

Ausgewählte Kapitel der Systemsoftware: Cloud Computing

Friedrich-Alexander-Universität Erlangen-Nürnberg

Koordinierung

- Viele parallel arbeitende Prozesse
- Koordinierung = Abstimmung der Aktivitäten verschiedener Prozesse
- Zugriff auf gemeinsame Ressourcen synchronisieren

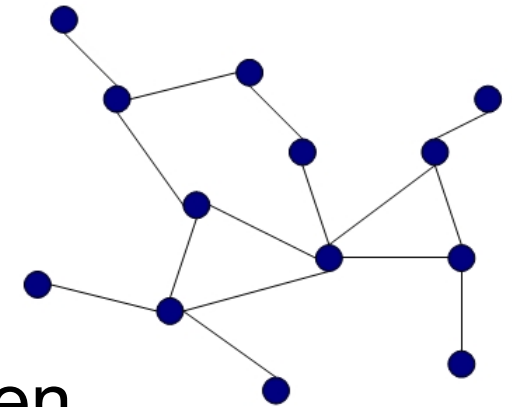
- Beispiele:
 - Locks
 - Producer-Consumer-Queues
 - Leader Election
 - Verwaltung von Gruppen

Koordinierung in der Cloud

- Strukturwandel im Rechenzentrum
 - Neuer Ansatz durch Cloud Computing
 - Günstige Commodity-Hardware in großer Stückzahl
 - Ausfälle bewusst einkalkuliert
- Anforderungen an Koordination
 - Hohe Skalierbarkeit
 - Zuverlässigkeit trotz Ausfällen
 - Konsistenz

Beispiel: Locks

- Lokaler Fall
 - Unmittelbarer Zugriff auf Lock
 - Effizient implementierbar
- Verteilter Fall
 - Kommunikation über Netzwerk
 - Nachrichten für Betreten und Verlassen
 - Teuer durch Latenzen und Übertragungsvolumen
- Cloud-System
 - Extrem viele Knoten im Netzwerk
 - Hohe Anforderungen an Algorithmus

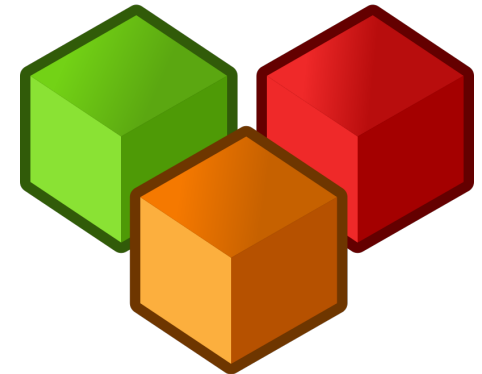


Eigenentwicklungen

- Vorteile
 - Direkte Einbettung in Anwendung
 - Beschränkung auf benötigte Funktionen
- Probleme
 - Zeitmangel in Projekten
 - Koordinierung keine Kernaufgabe
- Folge
 - Nicht-optimale / fehlerhafte Lösungen
 - Auswirkungen auf gesamte Anwendung

Externe Komponenten

- Integration einer Komponente
 - Generisch
 - Ausgereift
- Wiederkehrende Investition in Eigenentwicklung wird vermieden
- Vorhandene Lösungen:
 - **Apache ZooKeeper**
 - Chubby Lock Service (Google)



Quelle [1]

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Apache ZooKeeper

- Unterprojekt von Hadoop
- Generischer Koordinierungsdienst
- Speicherung von Metadaten (wenige KB)
- Einsatzzwecke
 - Konfiguration
 - Namensdienst
 - Koordinierung
 - Leader Election
 - Verwaltung von Gruppen



Quelle [2]

Gliederung

1. ZooKeeper

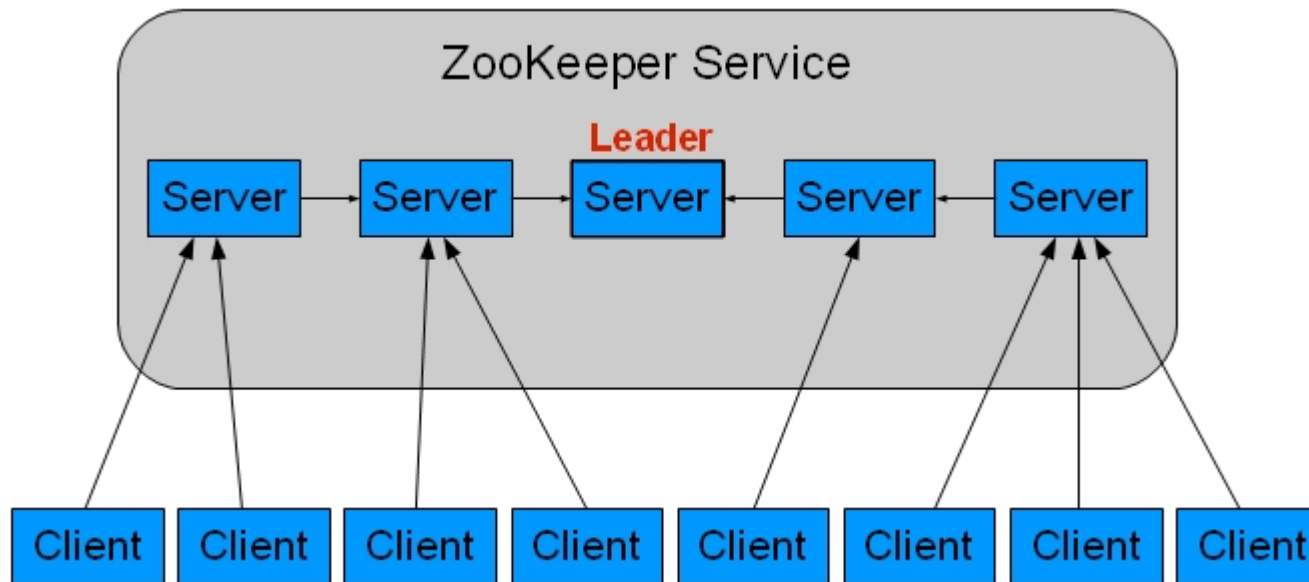
- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

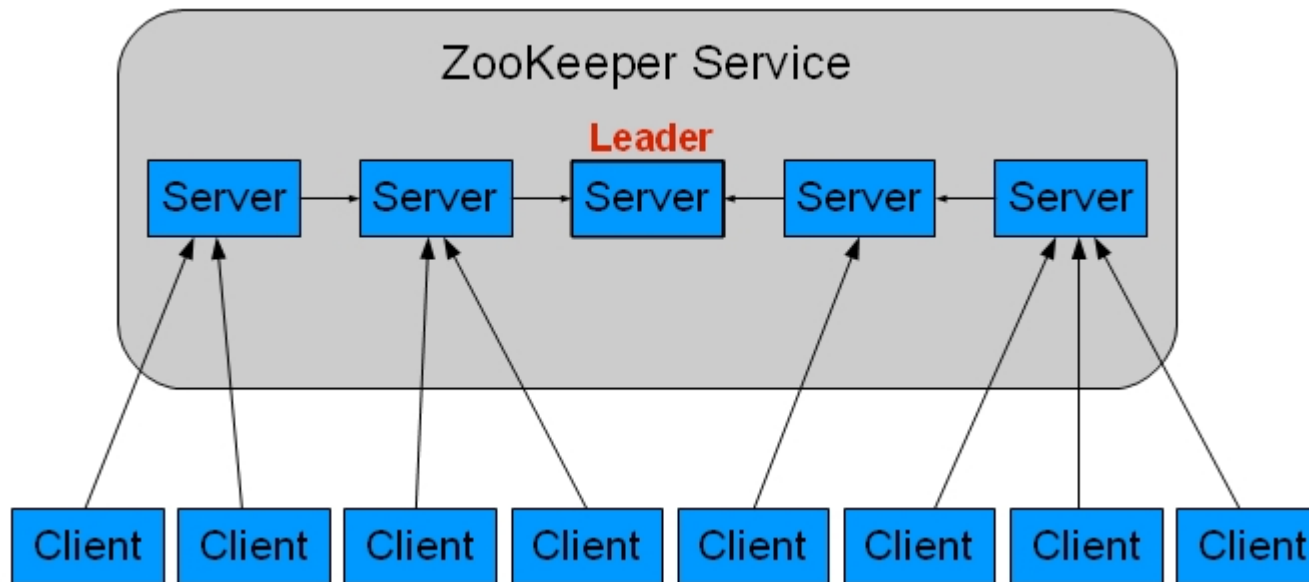
3. Fazit

Architektur Server-Seite



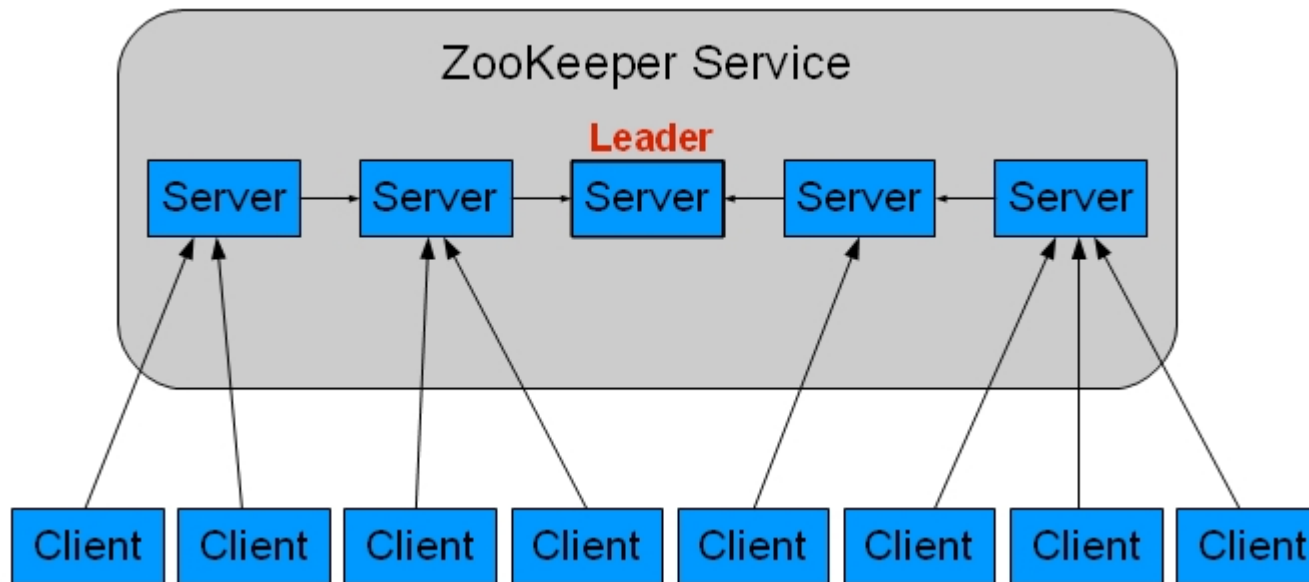
- Vollständiges Replikat auf jedem Server
 - Fehlertoleranz
 - Verfügbarkeit
- Daten im Arbeitsspeicher
 - Kurze Zugriffszeiten

Architektur Server-Seite



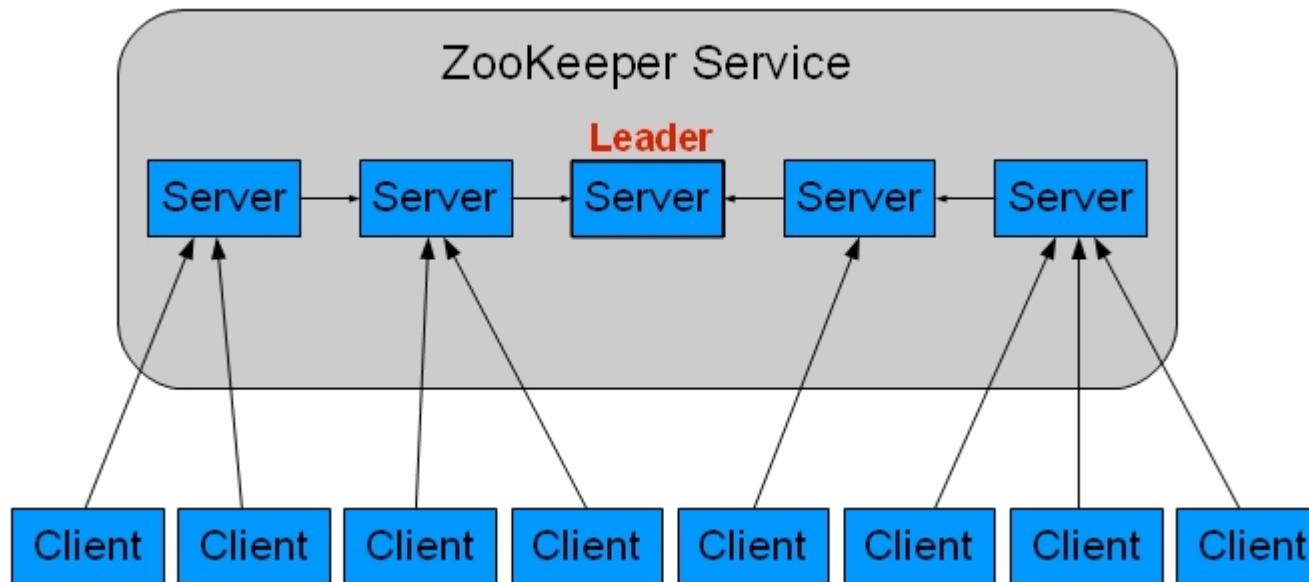
- Konsistenz durch Konsensalgorithmus
- Persistenz
 - Transaktionsprotokoll
 - Zustandssicherungen
- Leader definiert Ordnung auf Schreiboperationen

Architektur Client-Seite



- Client-Bibliothek zur Kommunikation
- Verbindung mit beliebigem Server
 - Sitzungsbasiert
 - Lastverteilung möglich

Architektur Client-Seite



- Schreiboperationen

- Weiterleitung an Leader
- Vollzieht Änderungen konsistent auf Replikaten

- Leseoperationen

- Beantwortung direkt durch Replikant

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

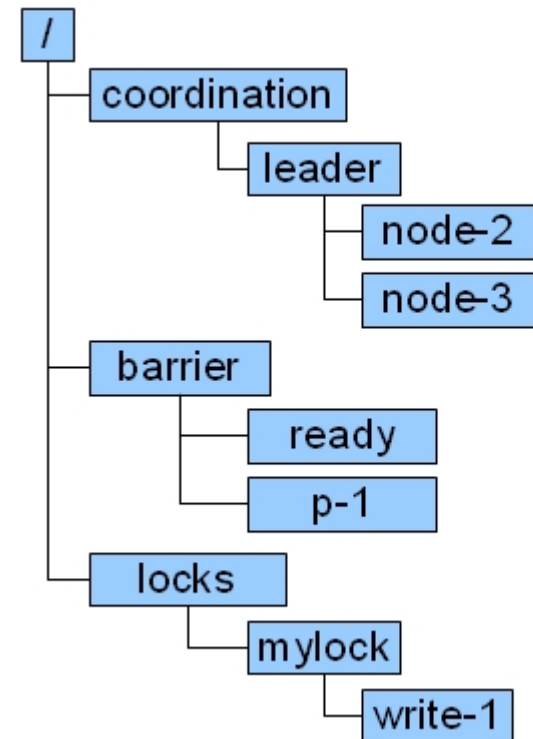
2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Datenmodell

- Hierarchische Datenstruktur
- Knoten speichern Daten und besitzen Unterknoten
- Keine partiellen Zugriffe
- Zusatzinformationen
 - Versionsnummern
 - Zeitstempel
 - Access Control Lists (ACLs)



Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- **Wichtige Konzepte**

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Konzepte: Ephemeral Nodes

- Vergängliche Knoten (Ephemeral Nodes)
 - Automatische Löschung bei
 - Sitzungsende
 - Client-Ausfall
 - Statusüberwachung von Clients
 - Automatisches Aufräumen der Daten

Konzepte: Watches

- Wächter (Watches)
 - Überwachung von Knoten
 - Automatische Benachrichtigung bei Änderungen
 - Einmalige Auslösung
 - Vermeidet ständiges Nachfragen

Konzepte: Conditional Updates

- Bedingte Änderungen (Conditional Updates)
 - Beispiel: Zähler inkrementieren
 - Daten lesen
 - Zähler erhöhen
 - Daten schreiben
 - Getrennte Schritte
 - Während der Ausführung vergeht Zeit
 - Netzwerklatenzen
 - Rechenzeit

Konzepte: Conditional Updates

- Problem

- Parallele Prozesse arbeiten mit den selben Daten
- Überlappung paralleler Operationen möglich
- Falsche Ergebnisse

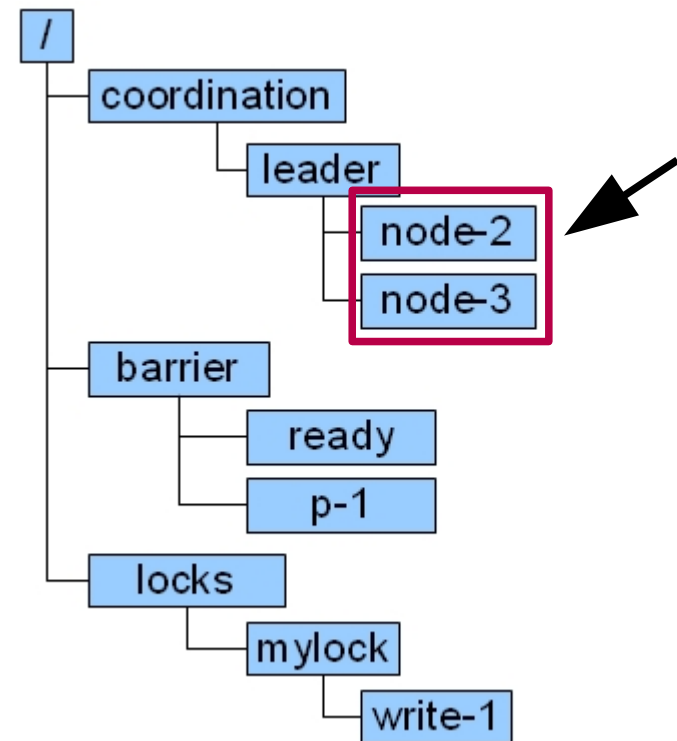
- Lösung

- Angabe der erwarteten Versionsnummer bei Schreib- und Löschooperationen
- Ausführung nur bei Übereinstimmung

Konzepte: Sequence Nodes

- Generierte Knotennamen (Sequence Nodes)
 - Anhängen von Sequenznummern
 - Rückgabe des erzeugten Namens

- Eindeutige Knotennamen
- Merkmal der Reihenfolge



Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Verwendung von ZooKeeper

- „Out of the box“
 - Namensdienst
 - Konfigurationsdatenbank
 - Verwaltung von Knoten
- Komplexere Koordinierungsfunktionen
 - Rezepte zeigen beste Umsetzung
 - Konventionen
 - Ziele
 - Kein aktives Warten
 - Keine Herden-Effekte

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Leader Election

- Häufigste Verwendung von ZooKeeper
 - Verwaltung einer Gruppe von Rechnern
 - Ausfallerkennung
 - Ein gemeinsamer Leader
 - Zentraler Organisator
 - IDs generieren
 - Arbeit verteilen
 - Aufgaben zuweisen
 - Nutzt ZooKeeper zur Speicherung von Metadaten
 - Speicherorte
 - Zugeteilte Aufgaben
 - Wertebereiche

Leader Election: Rezept

- Gemeinsames Verzeichnis */leader*
- Ablauf jedes Clients
 1. Vergänglichem Knoten „node-zz“ erstellen
 2. Unterknoten von */leader* abfragen
 3. Nächstkleinere Sequenznummer suchen
 - a) Keine kleinere → Leader
 - b) Sonst → Watch setzen
- Bei Ausfall wird Knoten automatisch gelöscht
 - Genau ein Client wird benachrichtigt
 - Leader beerben / neues Watch setzen

Sequenznummer

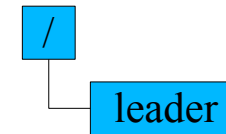


Leader Election: Beispielablauf

Knoten:

1

ZooKeeper Datenstruktur:



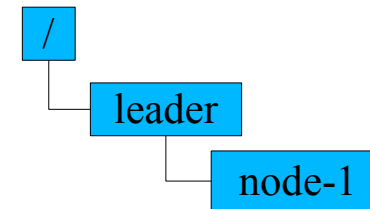
Client 1 verbindet sich

Leader Election: Beispielablauf

Knoten:

1

ZooKeeper Datenstruktur:



Erstelle Ephemeral Node als Repräsentant

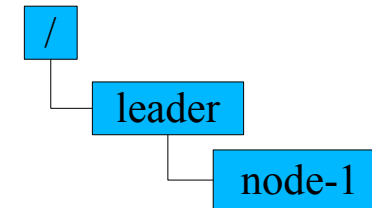
Leader Election: Beispielablauf

Knoten:

Leader

1

ZooKeeper Datenstruktur:



Client 1 wird zum Leader, da er die kleinste Sequenznummer hat

Leader Election: Beispielablauf

Knoten:

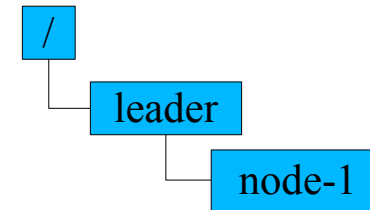
Leader

①

②

Client 2 verbindet sich

ZooKeeper Datenstruktur:



Leader Election: Beispielablauf

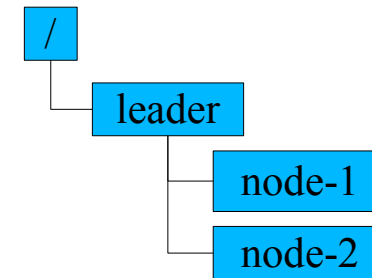
Knoten:

Leader

1

2

ZooKeeper Datenstruktur:



Erstelle Ephemeral Node als Repräsentant

Leader Election: Beispielablauf

Knoten:

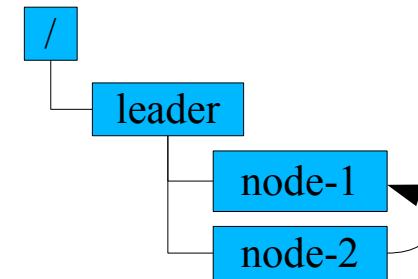
Leader

1

2

Setze ein Watch auf node-1

ZooKeeper Datenstruktur:



Leader Election: Beispielablauf

Knoten:

Leader

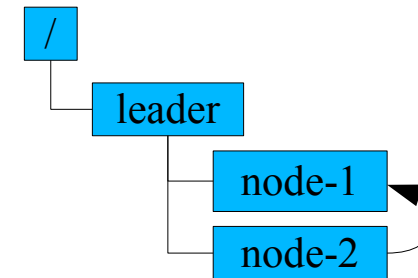
1

3

2

Client 3 verbindet sich

ZooKeeper Datenstruktur:



Leader Election: Beispielablauf

Knoten:

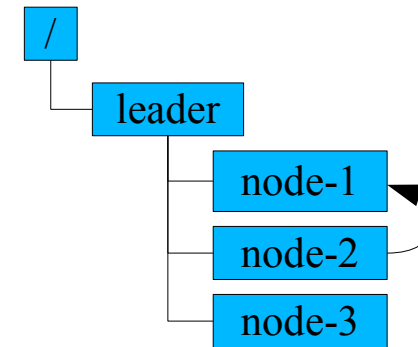
Leader

1

3

2

ZooKeeper Datenstruktur:



Erstelle Ephemeral Node als Repräsentant

Leader Election: Beispielablauf

Knoten:

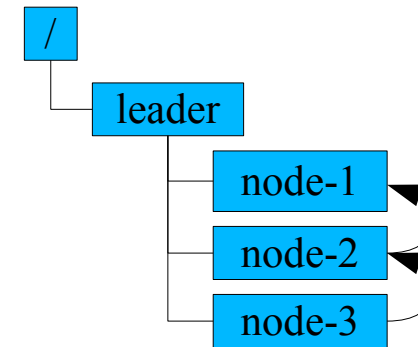
Leader

1

3

2

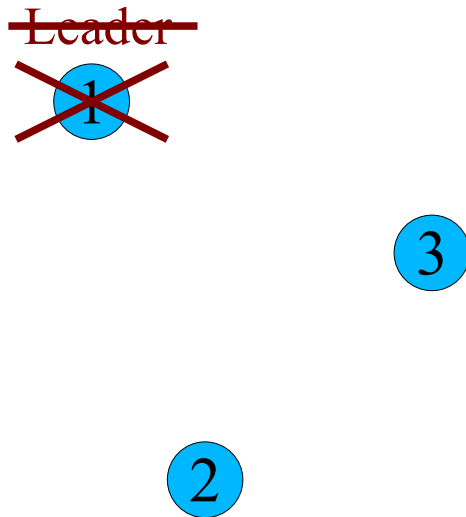
ZooKeeper Datenstruktur:



Setze ein Watch auf node-2

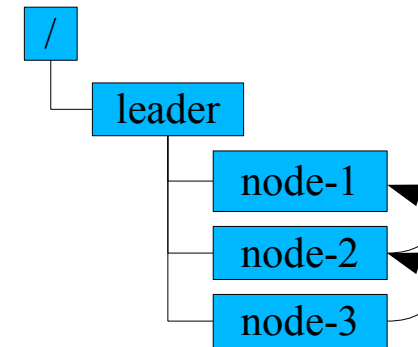
Leader Election: Beispielablauf

Knoten:



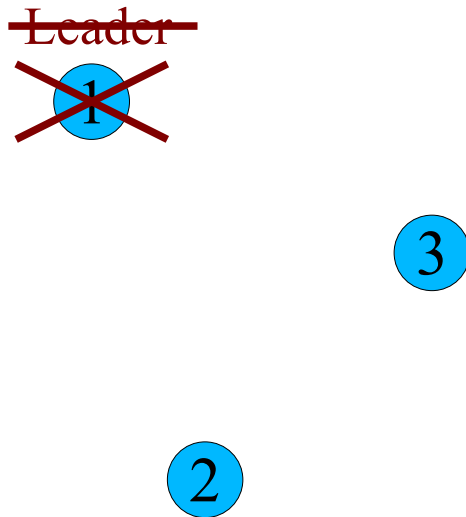
Ausfall des Leaders

ZooKeeper Datenstruktur:

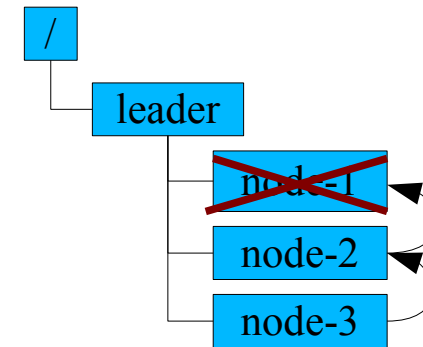


Leader Election: Beispielablauf

Knoten:



ZooKeeper Datenstruktur:



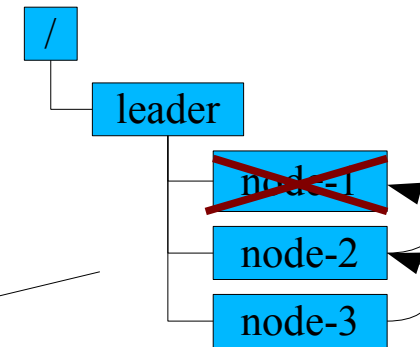
Automatische Löschung von node-1

Leader Election: Beispielablauf

Knoten:



ZooKeeper Datenstruktur:



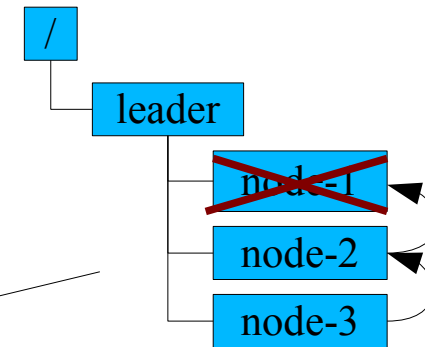
Automatische Benachrichtigung von Client 2

Leader Election: Beispielablauf

Knoten:



ZooKeeper Datenstruktur:



Client 2 ersetzt den Leader

Anwendungsfälle in der Cloud

- Katta
 - Verteilte Speicherung großer Datenbank-Indizes
 - Sharding (horizontale Partitionierung)
 - Master verwaltet die replizierten Shards auf den Slave-Rechnern
 - Aufgabe von ZooKeeper
 - Wahl des Masters
 - Verwaltung der Slaves und der Shards

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Wichtige Konzepte

2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Doppelte Barrieren

- Barrieren
 - Synchronisierung einer Gruppe paralleler Prozesse
 - Auf Eintreffen aller Prozesse warten
 - Gemeinsam in weiteren Ablauf starten
- Einsatz
 - Iterative Verfahren
 - Abläufe mit Phasen
- Doppelte Barrieren
 - Gemeinsamer Ein- und Austritt eines Bereichs

Doppelte Barrieren: Rezept

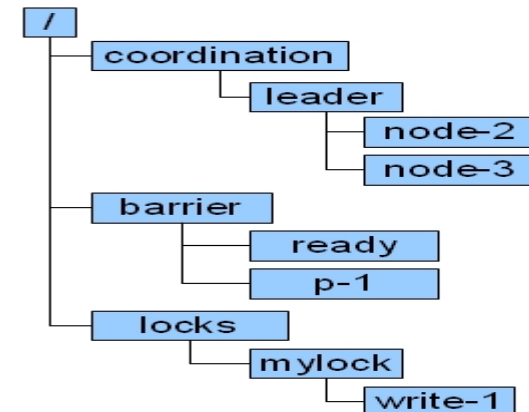
- Gemeinsamer Pfad: */barrier*
- Anzahl der Prozesse: x
- Eintritt in Barriere
 - Alle Prozesse registrieren sich unter */barrier*
 - Letzter Prozesse erzeugt */barrier/ready*
 - Alle Prozesse werden benachrichtigt
 - Betreten Barriere

Rezept

- Eintrittsprozedur

1. Setze ein Watch auf */barrier/ready*
2. Erstelle einen vergänglichen Knoten „p-zz“ als Repräsentant unter */barrier*
3. Frage Unterknoten von */barrier* ab
 - a) x Prozesse registriert → erstelle */barrier/ready*
 - b) Sonst warte auf Benachrichtigung von */barrier/ready*
4. Passiere erste Barriere

Sequenznummer

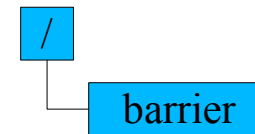


Doppelte Barrieren: Beispielablauf

x=3

Knoten:

ZooKeeper Datenstruktur:



Noch kein Prozess hat Barriere erreicht

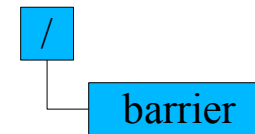
Doppelte Barrieren: Beispielablauf

x=3

Knoten:

①

ZooKeeper Datenstruktur:



Prozess 1 erreicht die Barriere

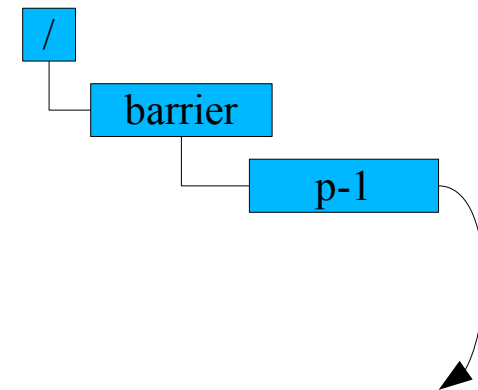
Doppelte Barrieren: Beispielablauf

$x=3$

Knoten:

1

ZooKeeper Datenstruktur:



Prozess 1 registriert sich und setzt ein Watch auf den ready-Knoten

Doppelte Barrieren: Beispielablauf

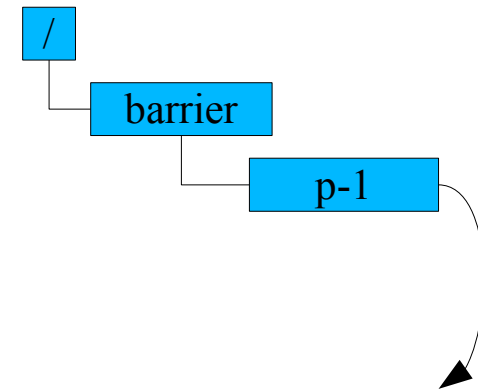
$x=3$

Knoten:

1

2

ZooKeeper Datenstruktur:



Prozess 2 erreicht die Barriere

Doppelte Barrieren: Beispielablauf

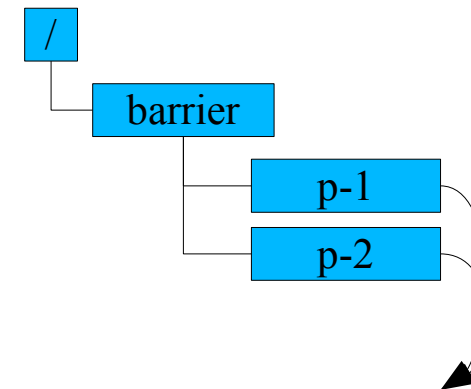
x=3

Knoten:

1

2

ZooKeeper Datenstruktur:



Prozess 2 registriert sich und setzt ein Watch auf den ready-Knoten

Doppelte Barrieren: Beispielablauf

x=3

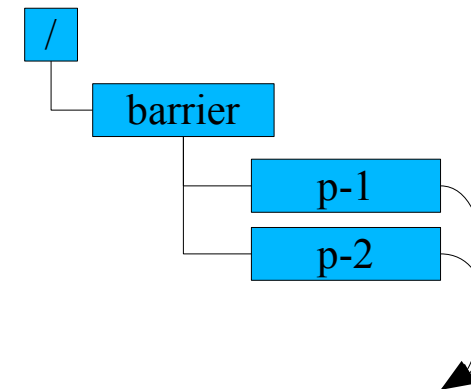
Knoten:

1

3

2

ZooKeeper Datenstruktur:



Prozess 3 erreicht die Barriere

Doppelte Barrieren: Beispielablauf

x=3

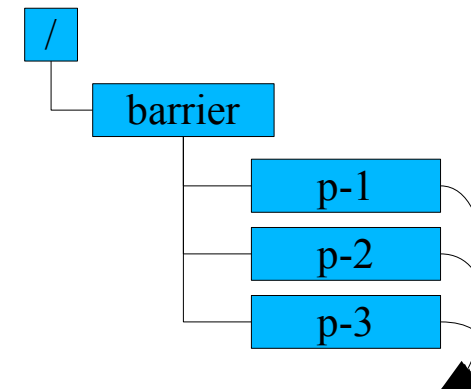
Knoten:

1

3

2

ZooKeeper Datenstruktur:

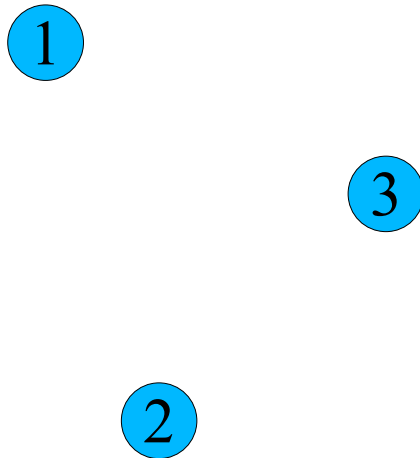


Prozess 3 registriert sich und setzt ein Watch auf den ready-Knoten

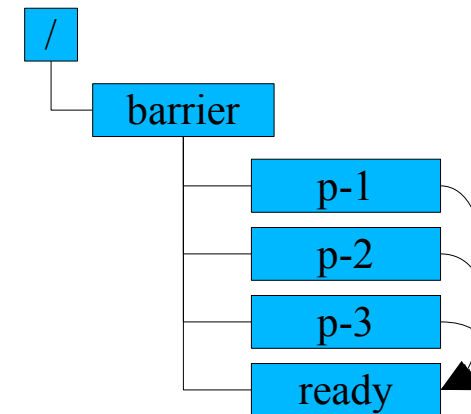
Doppelte Barrieren: Beispielablauf

$x=3$

Knoten:



ZooKeeper Datenstruktur:



Prozess 3 erkennt, dass $x=3$ Prozesse registriert sind
→ Erstellt ready-Knoten

Doppelte Barrieren: Beispielablauf

x=3

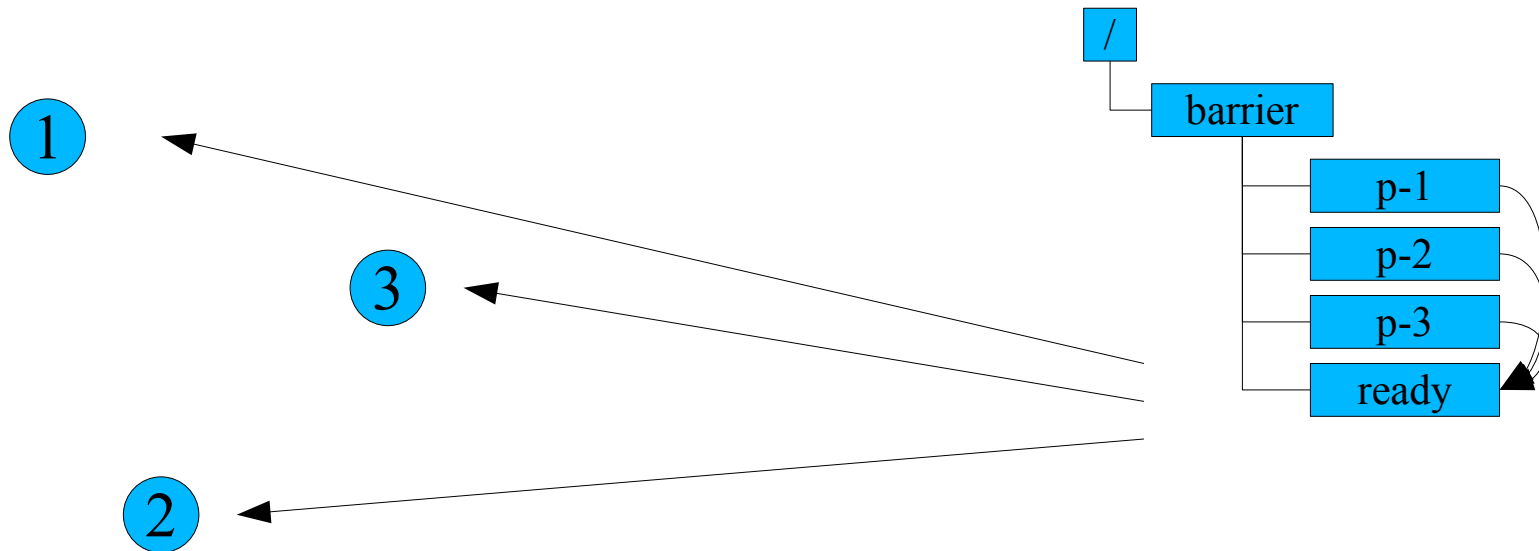
Knoten:

1

3

2

ZooKeeper Datenstruktur:



Alle Prozesse werden benachrichtigt und passieren die Barriere

Doppelte Barrieren: Rezept

- Prozesse durchlaufen parallel den Bereich innerhalb der Barriere
- Austritt aus Barriere
 - Wenn alle Prozesse ihren Knoten gelöscht haben
 - Kein Blockieren durch ausgefallene Prozesse
 - Ausfall → Ephemeral Node verschwindet automatisch
 - Ausnahme: kleinster Knoten
 - Erst zum Schluss löschen
 - Dient als Signal
 - Vermeidung von Herden-Effekten

Doppelte Barrieren: Rezept

- Austrittsprozedur

1. Frage Unterknoten von */barrier* ab

- a) Keine Knoten

- Verlasse Barriere

- b) Nur eigener Knoten

- Löschen und Verlassen

- c) Eigener Knoten hat kleinste Nummer

- Überwache den höchsten Knoten

- d) Sonst

- Lösche den eigene Knoten und überwache den kleinsten

2. Wiederhole Ablauf bei Benachrichtigung

Doppelte Barrieren: Beispielablauf

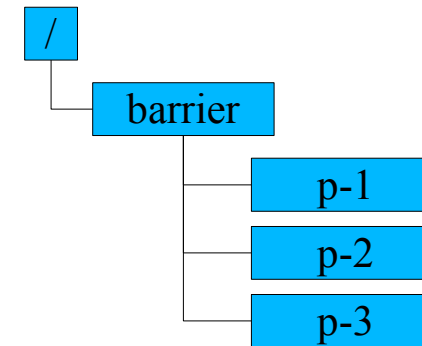
Knoten:

1

3

2

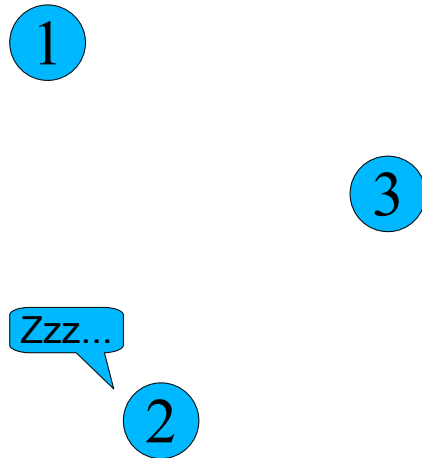
ZooKeeper Datenstruktur:



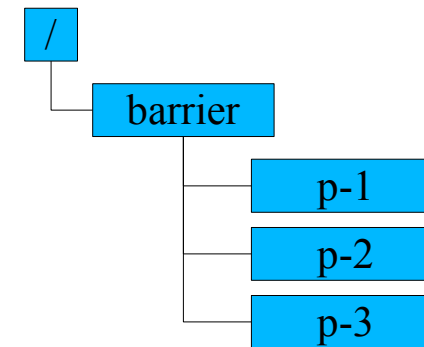
Prozesse führen Ablauf innerhalb der doppelten Barriere aus

Doppelte Barrieren: Beispielablauf

Knoten:



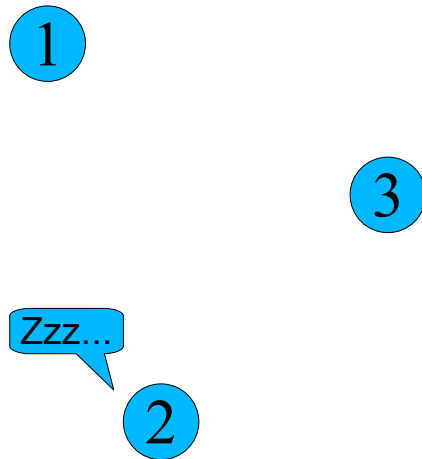
ZooKeeper Datenstruktur:



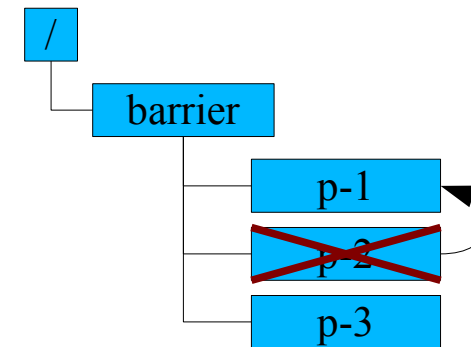
Prozess 2 erreicht das Ende der Barriere

Doppelte Barrieren: Beispielablauf

Knoten:



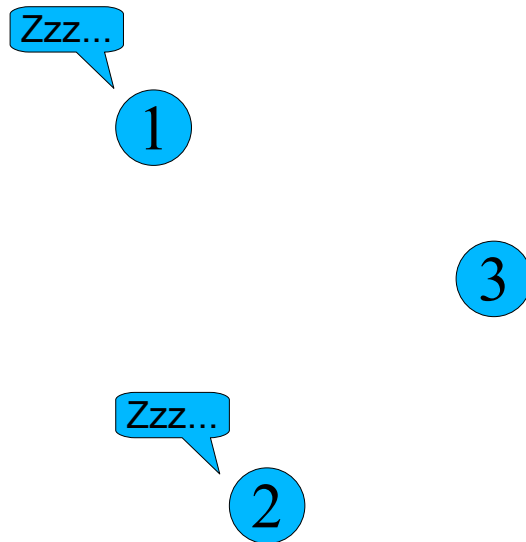
ZooKeeper Datenstruktur:



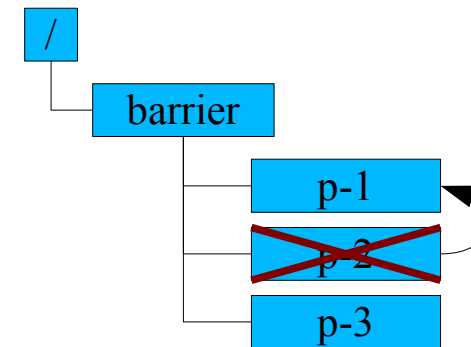
Prozess 2 löscht p-2 und setzt ein Watch auf den kleinsten Knoten

Doppelte Barrieren: Beispielablauf

Knoten:



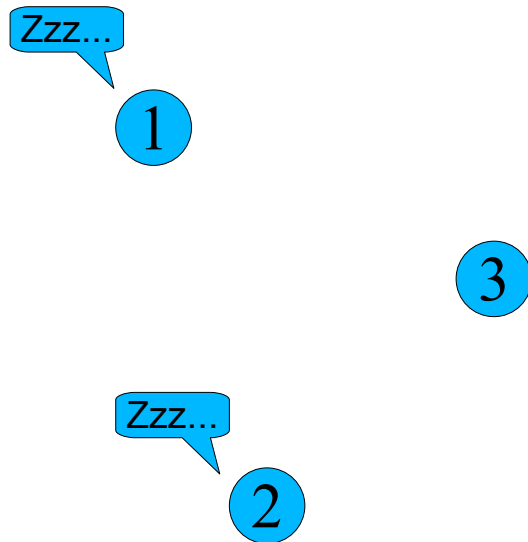
ZooKeeper Datenstruktur:



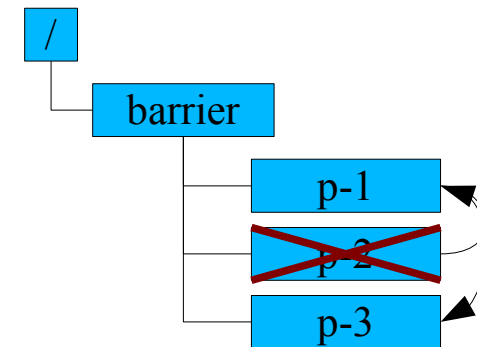
Prozess 1 erreicht das Ende der Barriere

Doppelte Barrieren: Beispielablauf

Knoten:



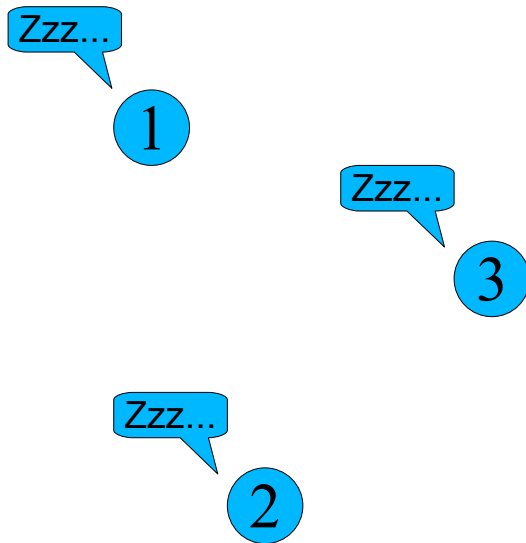
ZooKeeper Datenstruktur:



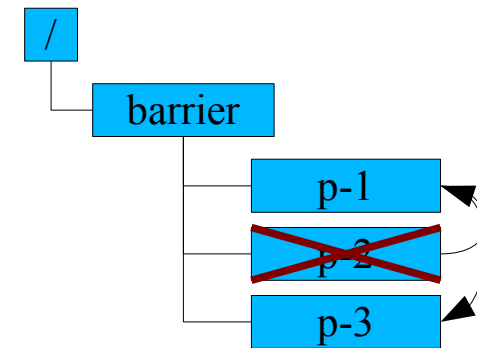
Prozess 1 setzt ein Watch auf den höchsten Knoten

Doppelte Barrieren: Beispielablauf

Knoten:



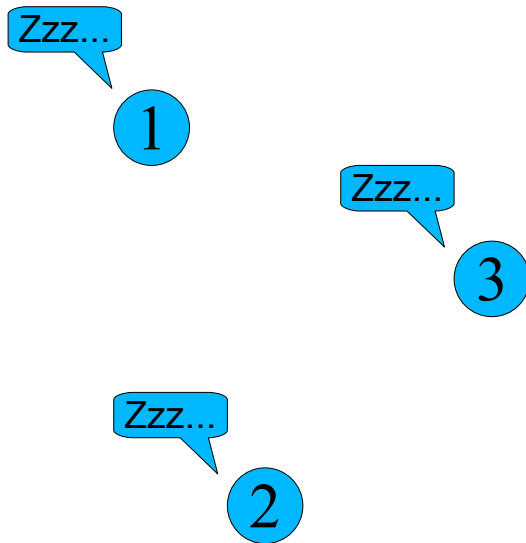
ZooKeeper Datenstruktur:



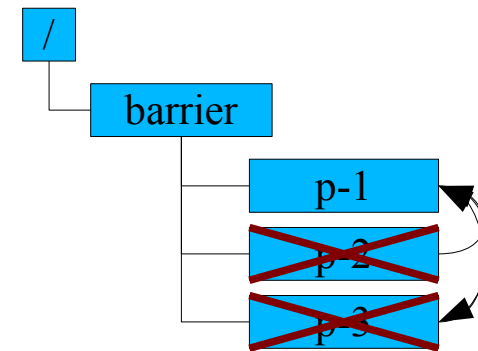
Prozess 3 erreicht das Ende der Barriere

Beispielablauf

Knoten:



ZooKeeper Datenstruktur:

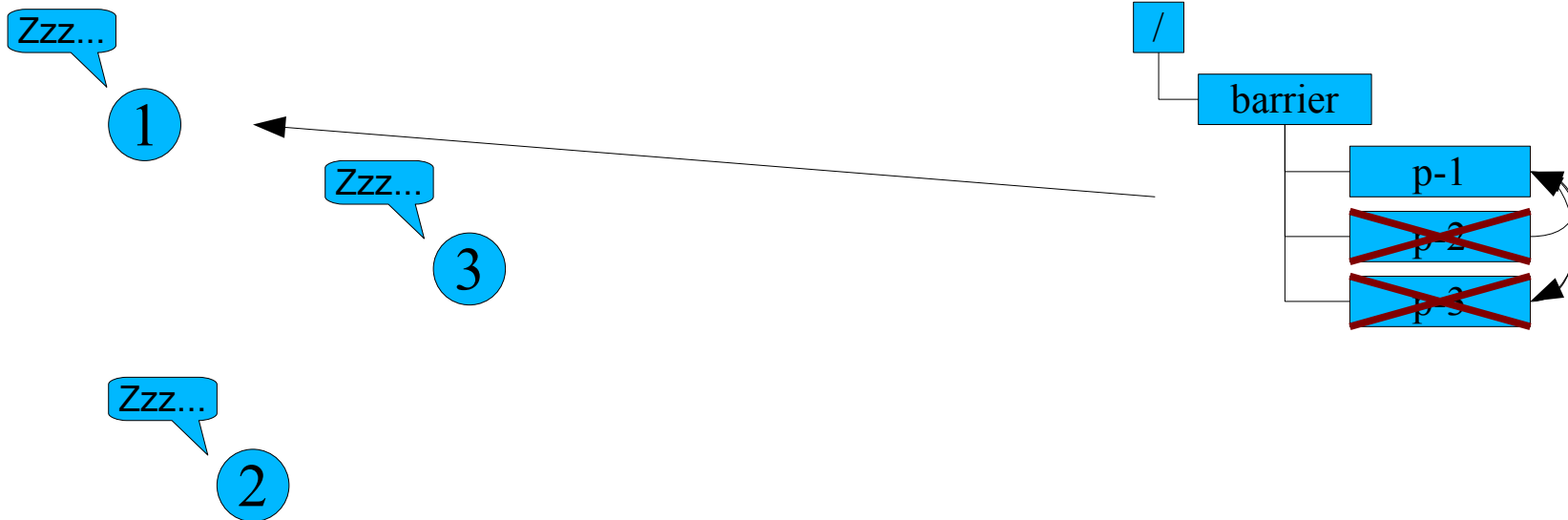


Prozess 3 löscht p-3 und setzt ein Watch auf den kleinsten Knoten

Beispielablauf

Knoten:

ZooKeeper Datenstruktur:

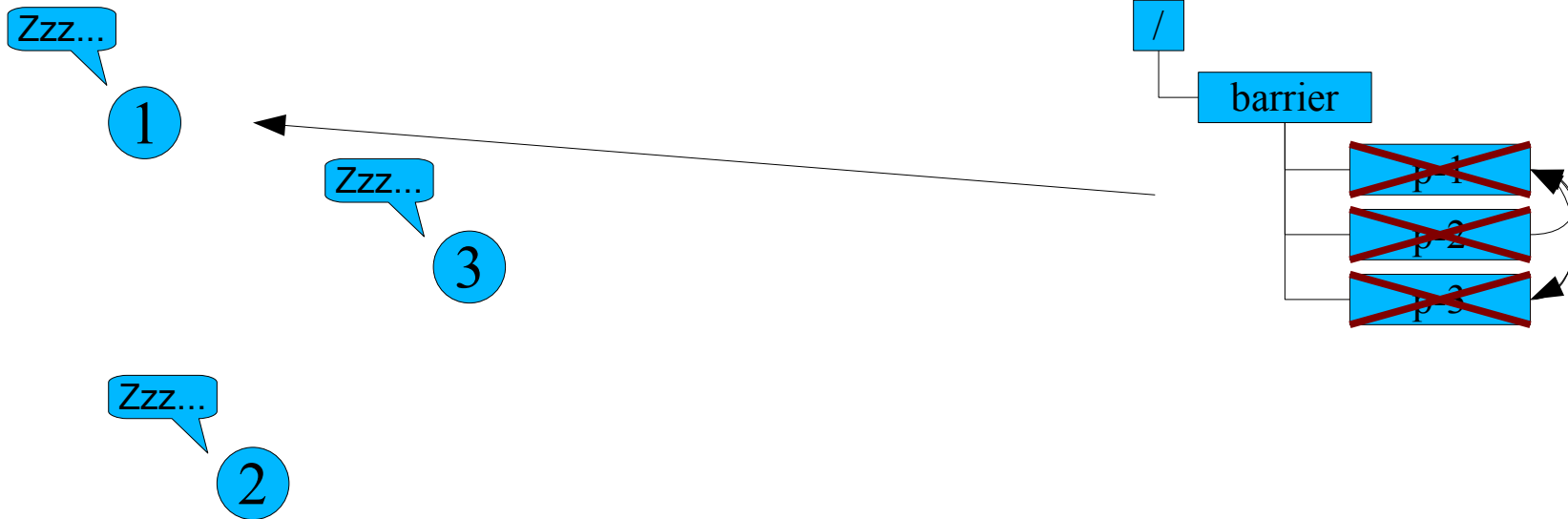


Prozess 1 wird benachrichtigt und erkennt, dass nur noch p-1 übrig ist

Beispielablauf

Knoten:

ZooKeeper Datenstruktur:

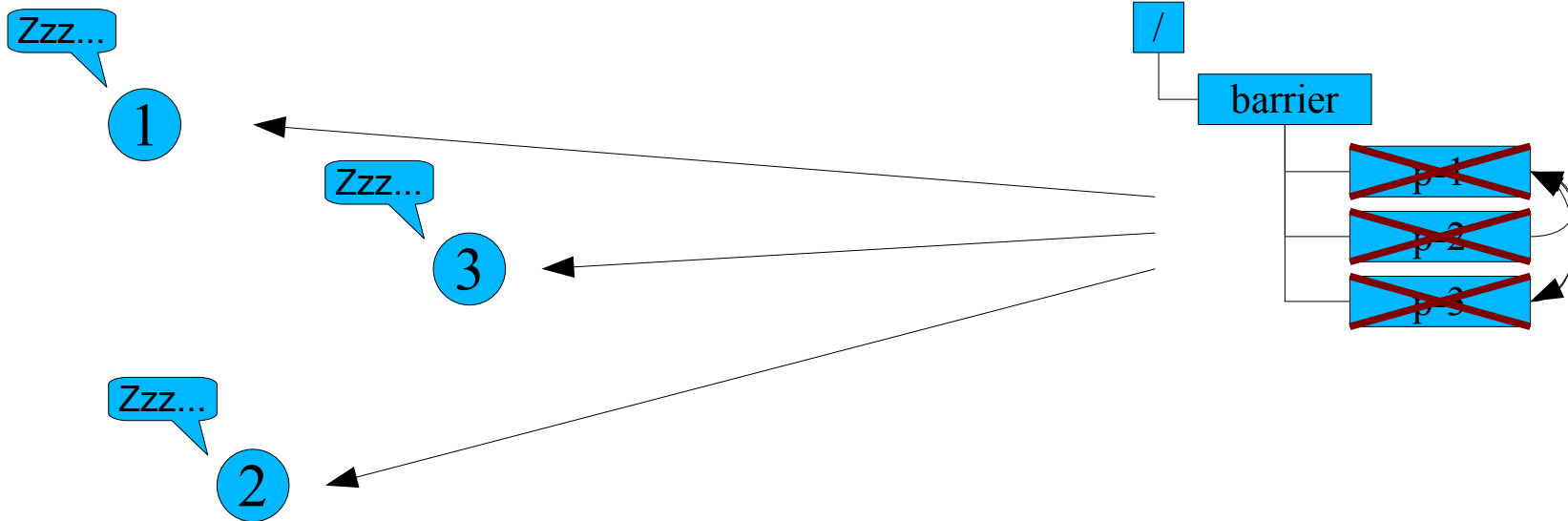


Prozess 1 löscht p-1

Beispielablauf

Knoten:

ZooKeeper Datenstruktur:

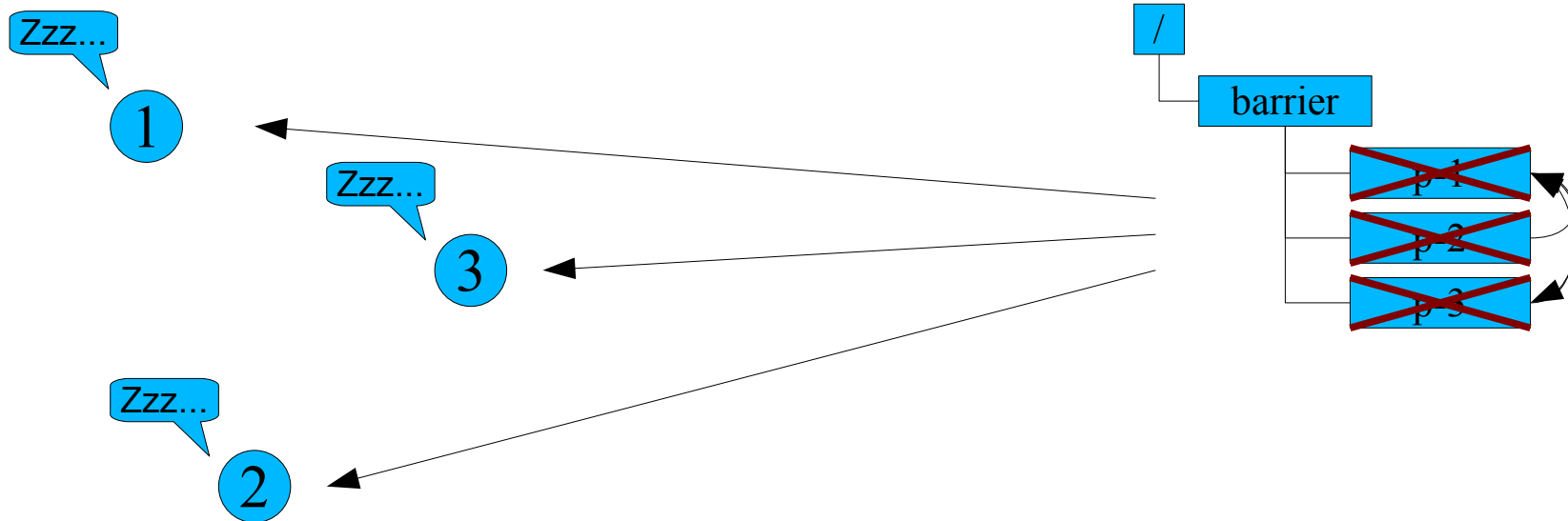


Die restlichen Prozess werden automatisch benachrichtigt

Beispielablauf

Knoten:

ZooKeeper Datenstruktur:



Alle Prozesse können die Barriere verlassen

Anwendungsfälle in der Cloud

- MapReduce
 - Parallele Verarbeitung großer Datenmengen
 - Getrennte Phasen
 - Map: Eingabedaten → Schlüssel-Wert-Paare
 - Sortierphase: Sortieren der Zwischenergebnisse
 - Reduce: Verarbeiten der Zwischenergebnisse
 - Bauen aufeinander auf
 - Kein Reduce vor Abschluss des Sortierens

Anwendungsfälle in der Cloud

- Übergänge zwischen Phasen
 - Durch Barrieren synchronisiert
 - Von ZooKeeper bereitgestellt
- Einsatz bei Yahoo
 - Internet-Suchmaschine
 - Index-Berechnungen mit MapReduce
 - Verwendete Cluster:
mehr als 10.000 Prozessorkerne

Gliederung

1. ZooKeeper

- Architektur
- Datenmodell
- Garantien
- Wichtige Konzepte

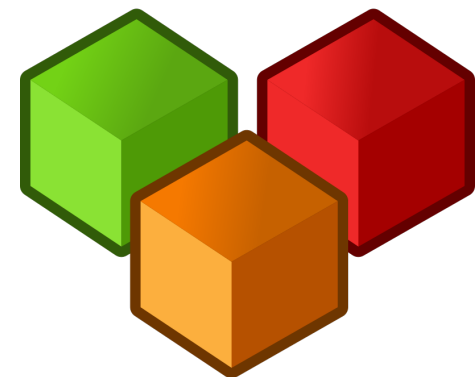
2. Verwendung von ZooKeeper

- Leader Election
- Doppelte Barrieren

3. Fazit

Fazit

- Koordinierung als Komponente
 - Erspart Eigenentwicklungen
 - Meist hinreichend getestet und daher verlässlich
 - Nachteile
 - Investitionen in Einarbeitung/Schulung
 - Einrichtung des Dienstes
 - Keine angepasste Lösung



Quelle [1]

Fazit

- Koordinierung mit ZooKeeper
 - Ausgereift
 - Zuverlässig und einfach
 - Vielfältige Anwendungsmöglichkeiten
 - Enge Bindung an Yahoo
 - Begrenzte Skalierbarkeit
 - Arbeitsspeicher der Server begrenzt Knotenanzahl
 - Leader als Flaschenhals für Schreiboperationen



Quelle [2]

Ende

Vielen Dank für die Aufmerksamkeit!

Fragen oder Anmerkungen?

Quellen

- [1] <http://www.openclipart.org/detail/16983>
- [2] <https://svn.apache.org/repos/asf/hadoop/zookeeper/logo/>