

# Verteilte Systeme

Georeplikation

---

Sommersemester 2022

Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg  
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)



Lehrstuhl für Verteilte Systeme  
und Betriebssysteme



FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

## Georeplikation

Motivation

Konsistenzgarantien

Pileus

Platform for Nimble Universal Table Storage (PNUTS)

- Georeplikation
  - Platzierung von Replikaten an **verschiedenen geografischen Standorten**
  - Konsequenzen
    - Minimierung der Fehlerabhängigkeit zwischen Replikaten
    - Reduzierung der Distanz zwischen Client und Replikat
    - Höhere Latenzen zwischen den Replikaten
- Probleme bei der Bereitstellung **starker Konsistenz**
  - Gesteigerte Antwortzeit aufgrund eines mehrphasigen Einigungsprotokolls
  - Erhöhter Kommunikationsaufwand durch Austausch von Bestätigungen
  - Nichtverfügbarkeit von Replikaten, die keinen Kontakt zur Gruppe haben
  - Vergleiche: Uniformer zuverlässiger Multicast
- Herausforderungen
  - Welche **abgeschwächten Konsistenzgarantien** sind praktikabel?
  - Wie kann ein System unterschiedliche Konsistenzgarantien unterstützen?

## Georeplikation

Motivation

Konsistenzgarantien

Pileus

Platform for Nimble Universal Table Storage (PNUTS)

## ■ Beispiele typischer **Konsistenzgarantien aus der Praxis**

| Konsistenzgarantie                                                 | Beschreibung                                                                                        |
|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <i>Strong Consistency</i>                                          | Sichtbarkeit aller vorherigen Schreiboperationen                                                    |
| <i>Eventual Consistency</i><br>( <i>Letztendliche Konsistenz</i> ) | Sichtbarkeit einer Teilmenge der vorherigen Schreiboperationen                                      |
| <i>Consistent Prefix</i>                                           | Sichtbarkeit aller vorherigen Schreibaufrufe bis zu einem bestimmten Punkt in der Sequenz           |
| <i>Bounded Staleness</i>                                           | Sichtbarkeit aller Schreiboperationen bis zu einem bestimmten Zeitpunkt; eventuell beliebige danach |
| <i>Monotonic Reads</i>                                             | Sichtbarkeit einer monoton wachsenden Teilmenge der vorherigen Schreiboperationen                   |
| <i>Read My Writes</i>                                              | Sichtbarkeit aller vorherigen Schreiboperationen desselben Lesers                                   |

## ■ Literatur



Doug Terry

**Replicated data consistency explained through baseball**

*Communications of the ACM*, 56(12):82–89, 2013.

# Konsistenzgarantien im Vergleich

- Beispiel: Entwicklung des Spielstands in einem Baseball-Spiel

|                 |   |   |    |    |    |    |     |     |
|-----------------|---|---|----|----|----|----|-----|-----|
| Zeitpunkt [min] | 0 | 5 | 17 | 37 | 47 | 79 | 163 | 181 |
| Mannschaft A    | 0 | 0 | 1  | 1  | 1  | 2  | 2   | 2   |
| Mannschaft B    | 0 | 1 | 1  | 2  | 3  | 3  | 4   | 5   |

- Mögliche **Rückgabewerte von Leseaufrufen** zum Zeitpunkt  $t = 182$

[Annahme: Die Punkte jeder Mannschaft werden in separaten Datensätzen verwaltet, aber gemeinsam gelesen.]

|                                                                 |                                                                                              |
|-----------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <i>Strong Consistency</i>                                       | 2:5                                                                                          |
| <i>Eventual Consistency</i>                                     | 0:0, 0:1, 0:2, 0:3, 0:4, 0:5, 1:0, 1:1, 1:2, 1:3, 1:4, 1:5, 2:0, 2:1, 2:2, 2:3, 2:4 oder 2:5 |
| <i>Consistent Prefix</i>                                        | 0:0, 0:1, 1:1, 1:2, 1:3, 2:3, 2:4 oder 2:5                                                   |
| <i>Bounded Staleness</i><br>[Beispiel: max. 100 min alte Werte] | 2:3, 2:4 oder 2:5                                                                            |
| <i>Monotonic Reads</i><br>[Beispiel: 1. Aufruf bei 1:3]         | 1:3, 1:4, 1:5, 2:3, 2:4 oder 2:5                                                             |
| <i>Read My Writes</i>                                           | Offizieller Punktezähler: 2:5<br>Andere Leser: siehe letztendliche Konsistenz                |

- Offizieller Punktezähler bei Aktualisierung des Spielstands

```
public void incrementScore(String team) {  
    int oldScore = store.get(team);  
    store.put(team, oldScore + 1);  
}
```

⇒ **Read My Writes**

- Schiedsrichter nach der regulären Spielzeit: Entscheidung, ob Verlängerung nötig

```
public boolean isDraw() {  
    [scoreA, scoreB] = store.getAll("A", "B");  
    return (scoreA == scoreB);  
}
```

⇒ **Strong Consistency**

- Radioreporter bei periodischer Spielstandsmeldung

```
while(true) {  
    [scoreA, scoreB] = store.getAll("A", "B");  
    Report scoreA and scoreB;  
    Sleep for 30 minutes;  
}
```

**Consistent Prefix +**  
⇒ **(Monotonic Reads oder Bounded Staleness)**

⇒ **Konsistenzanforderungen sind abhängig vom Anwendungsfall**

## Georeplikation

Motivation

Konsistenzgarantien

Pileus

Platform for Nimble Universal Table Storage (PNUTS)

## ■ Problem

- Schwach konsistente Leseanfragen liefern oft stark konsistente Daten
- **Mehraufwand durch fehlendes Wissen** über die Aktualität von Daten

```
DATA data = weaklyConsistentGet([Schlüssel]);  
Display data;  
DATA latestData = stronglyConsistentGet([Schlüssel]);  
if(latestData != data) Display latestData;
```

## ■ Pileus

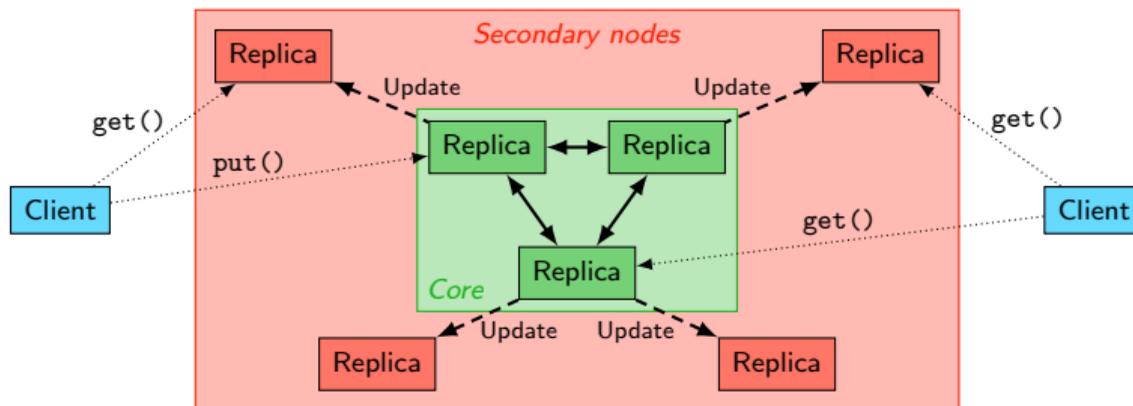
- Ansatz zur georeplizierten Speicherung von Schlüssel-Wert-Paaren
- Spezifizierung **konsistenzbasierter Dienstgütekriterien** durch den Client
- Anfrageabhängige Auswahl des Kontaktreplikats für Leseoperationen

## ■ Literatur



Douglas B. Terry, Vijayan Prabhakaran, Ramakrishna Kotla, Mahesh Balakrishnan et al.  
**Consistency-based service level agreements for cloud storage**  
*Proc. of the 24th Symposium on Operating Systems Principles (SOSP '13)*, S. 309–324, 2013.

- Systemkern
  - **Synchrone Replikation**
  - Bearbeitung von
    - Schreibanfragen
    - Leseanfragen mit starken Konsistenzanforderungen
- Sekundärknoten
  - **Asynchrone Übernahme der Daten** aus dem Systemkern
  - Bearbeitung von Leseanfragen mit schwachen Konsistenzanforderungen



# Konsistenzbasierte Dienstgütekriterien (SLAs)

## ■ Spezifizierung durch die **Client-Anwendung**

[Aufruf: `get(String key, SLA sla)`]

- Festlegung der gewünschten Konsistenz und einer oberen Latenzschranke
- Gewichtung der Kriterien nach „Nützlichkeit“ (*Utility*)

| Nr. | Konsistenzgarantie          | Latenz  | Nützlichkeit |
|-----|-----------------------------|---------|--------------|
| 1.  | <i>Strong Consistency</i>   | 150 ms  | 1.0          |
| 2.  | <i>Eventual Consistency</i> | 150 ms  | 0.5          |
| 3.  | <i>Strong Consistency</i>   | 1000 ms | 0.2          |

## ■ Auswertung durch die **Client-Bibliothek**

- Analyse des Verhaltens verschiedener Replikate
- Abschätzung von replikatspezifischen Wahrscheinlichkeiten
  - $P_C$                       Wahrscheinlichkeit für Einhaltung der Konsistenzgarantie
  - $P_L$                       Wahrscheinlichkeit für Bereitstellung der Latenz
  - $P_s = P_C \times P_L$  Wahrscheinlichkeit für Einhaltung des Dienstgütekriteriums  $s$
- **Auswahl des Kontaktreplikats**
  - Ziel: Maximierung von  $P_s \times s_{Utility}$  über alle  $s$
  - Bei Gleichstand: Wahl des Replikats mit der vermeintlich geringsten Latenz

- Einsatz von **Modifikationszeitstempeln für Datensätze**
  - Systemkern
    - Erzeugung von Zeitstempeln bei der Ausführung von Schreibanfragen
    - Weiterleitung von Änderungen in aufsteigender Reihenfolge ihrer Zeitstempel
  - Alle Replikate
    - Pro Datensatz: Verwaltung eines Zeitstempels  $T_D$  der letzten Änderung
    - $T_{max}$ : Zeitstempel der neuesten Aktualisierung
    - Antworten enthalten  $T_D$  und  $T_{max}$  [→ Abschätzung des Aktualitätsgrads der Daten möglich.]
- **Analyse des Replikatverhaltens** durch den Client
  - Messung der individuellen Umlaufzeiten zu einzelnen Replikaten
  - Verwaltung der neuesten bekannten  $T_{max}$  verschiedener Replikate
- Abschätzung von Konsistenzwahrscheinlichkeiten
  - Bestimmung eines erforderlichen **Mindestzeitstempels**  $T_{min}$  (Beispiele)
    - *Read My Writes* Zeitstempel der letzten Änderung des Clients
    - *Monotonic Reads* Zeitstempel der neuesten vom Client gelesenen Version
  - Vergleich von  $T_{min}$  und  $T_{max}$

## Georeplikation

Motivation

Konsistenzgarantien

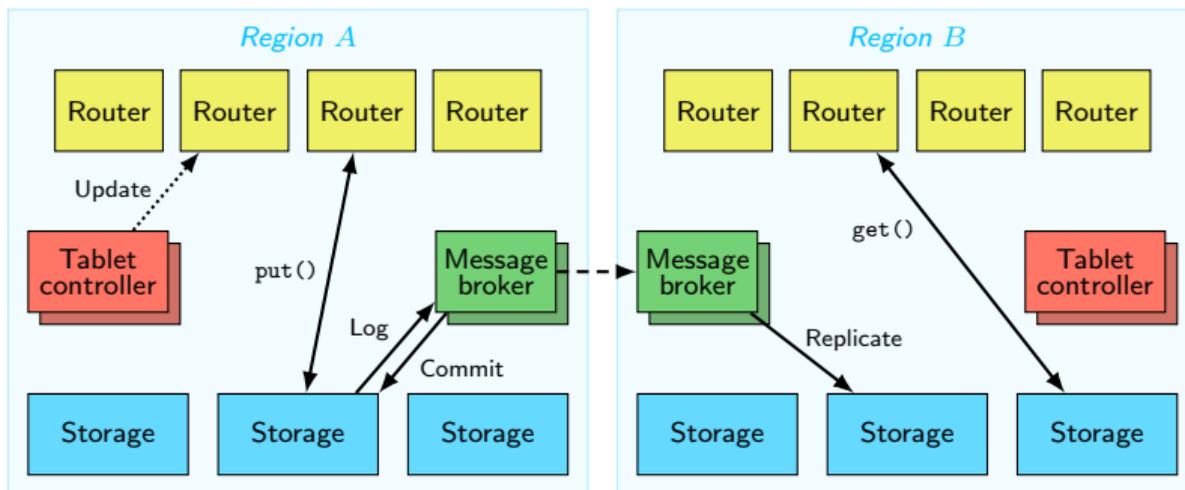
Pileus

Platform for Nimble Universal Table Storage (PNUTS)

# Platform for Nimble Universal Table Storage (PNUTS)

- Einsatzszenarien bei Yahoo (Beispiele)
  - Verwaltung von **Nutzerinformationen** (Profildaten, Zugriffsstatistiken,...)
  - Speicherung von Metadaten für Videos, Fotos und Email-Anhänge
- Verwaltung von **Schlüssel-Wert-Paaren in Tabellen**
  - Sortierung von Datensätzen
    - Aufsteigend nach Schlüssel oder Schlüssel-Hashwert
    - `scan()`: Zusätzliche Operation zum Iterieren über mehrere Datensätze
  - Speicherung von Tabellen aufgeteilt nach Abschnitten (*Tablets*)
- **Charakteristika**
  - Datensatzspezifische Primärreplikate
  - Alle Operationen mit hohen Latenzen werden asynchron ausgeführt
- Literatur
  -  Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein et al.  
**PNUTS: Yahoo!'s hosted data serving platform**  
*Proceedings of the VLDB Endowment*, 1(2):1277–1288, 2008.

- Replikation über **mehrere geografisch verteilte Regionen**
- Komponenten pro Region
  - Speicherknoten      Bearbeitung von Anfragen
  - Tablet-Controller    Zuordnung von Tablets zu Speicherknoten
  - Router                Weiterleitung von Anfragen an Speicherknoten
  - Message-Broker      Übermittlung von Nachrichten an **andere Regionen**



- Identifizierung des **zuständigen Speicherknotts** durch den Router
  - Router hält Kopie des Tablet-Controller-Zustands im Hauptspeicher
    - Information über Tablet-Grenzen
    - Abbildung von Tablets auf Speicherknotts
  - Überprüfung der Router-Entscheidung durch betroffenen Speicherknotts
  - Aktualisierung des Router-Zustands bei Fehlentscheidungen
- Identifizierung der **zuständigen Region** durch den Speicherknotts
  - Jeder Datensatz enthält Referenz auf die Region seiner Primärkopie
  - Weiterleitung von Schreibanfragen an entsprechende Region
  - Häufige Schreibanfragen von anderem Ort → Wechsel der Primärregion
  - Behandlung von Einfügeoperationen durch eine Default-Region
- **Ausführung einer Schreibanfrage** auf dem Speicherknotts
  - Übermittlung der Anfrage an den Message-Broker
  - Nach Message-Broker-Bestätigung: Lokale Bearbeitung der Anfrage