6 Verteilte Datenverwaltung

Ziel: *Zusammengehöriger* Datenbestand soll über mehrere Stationen *verteilt* werden,

z.B.
- **Fragmentierung**: in mehrere *Fragmente* aufgeteilt
- **Replikation**: in mehreren *Kopien* gespeichert
- **Mischformen** von Fragmentierung und Replikation

„zusammengehörig": gewisse **Konsistenz**-Eigenschaften, beschrieben durch **Invariante,** z.B. bei Replikation: „alle Kopien gleich"

Motivation: (Fragmentierung:) **Lastverteilung, Parallelarbeit** u.a.
(Replikation:) **Verfügbarkeit** – schnell und ausfallsicher

(z.B. verteilte Datenbank, verteiltes Dateisystem, verteilter Namensdienst, ...)

◈ *Nichtverteilter* Datenbestand:

- bestimmte Station/Prozess fungiert als zuständiger

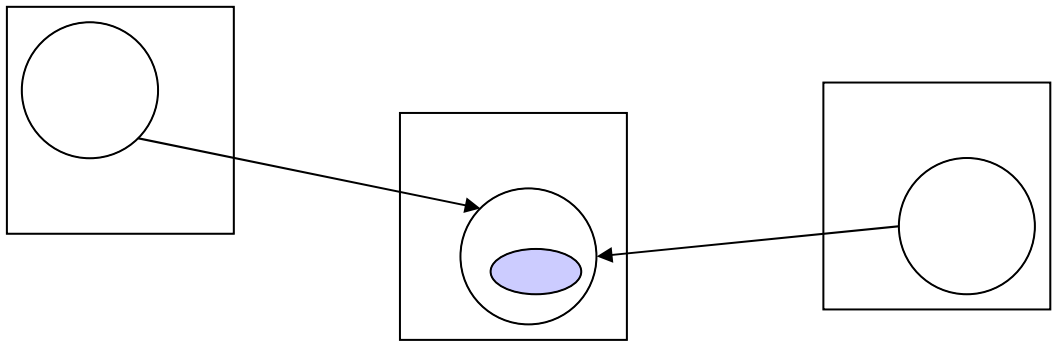
  Verwalter des Datenbestands

  (Träger, *host*, Anbieter, *server*),

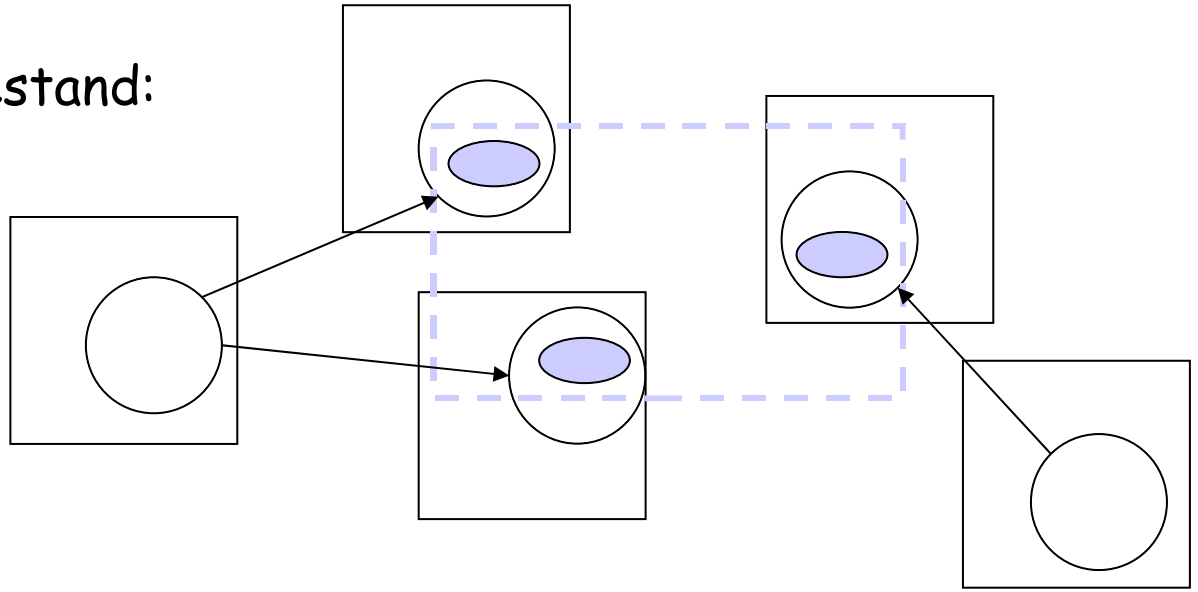
- nimmt von *Klienten*-Prozessen *(clients)* Nachrichten entgegen,

  die zum Lesen und/oder Modifizieren auffordern,

- antwortet gegebenenfalls mit gelesenen Werten.



◈ *Verteilter* Datenbestand:



Abstraktion: **Verteiltes Objekt**,

bestehend aus mehreren Teilen (Fragmenten/Kopien),

die auf mehrere Anbieter verteilt sind.

Operationen werden von Klienten ausgelöst

durch Nachrichten an Teil(e) des Objekts.

Konsistenz des verteilten Objekts ist bedroht durch

- lokale Nichtsequentialität (z.B. nebenläufiger Server)

  → Sperrsynchronisation/Transaktionen

- verteilte Nichtsequentialität

  → verteilter Ausschluss (→5.4)

  → verteilte Transaktionen (→6.3)

6.1 Replikation

bedeutet:
Ein **Datenobjekt** existiert in **mehreren Kopien**
auf verschiedenen Stationen *(copies, replicas)*

Vorteile:

- Kein Engpass für zugreifende Klienten
  → **Geschwindigkeit, Skalierbarkeit**
- Eventuell sogar **schneller lokaler** (oder „fastlokaler") Zugriff
- Gute **Verfügbarkeit** *(availability)* auch wenn eine Kopie
  (oder mehrere) nicht erreichbar/funktionsfähig
  (kein „*single point of failure*")
- **Fehlertoleranz** *(fault tolerance)* durch „Mehrheitsentscheidung"
  bei Fehlverhalten von Kopien

Nachteile:

- Die Kopien können **nicht** zu jedem Zeitpunkt (keine globale Zeit!) **identisch** gehalten werden.
  Ausnahme: replizierte Konstanten, replizierter Code.

- Schneller Lesezugriff – auf *eine* der Kopien – wird mit **aufwendigerem Schreibzugriff** - auf alle Kopien - erkauft.

Terminologie:

Objekte: programmiersprachliche Objekte
oder Seiten, Segmente, Dateien,
Tabellen in relationalen DB, ...

Verwalter: Prozess, der eine Objektkopie verwaltet;
auch Server oder **Replikatverwalter** *(replica manager)*

- (ist u.U. nichtsequentiell *(threaded)*)
- ist typischerweise für *mehrere* Objekte eines bestimmten Typs zuständig

Passive Replikation: Bei Änderung einer Kopie wird der neue Wert an alle anderen Verwalter geschickt.

Aktive Replikation: Bei Änderung einer Kopie wird der Operations-bezeichner samt Parametern an alle Verwalter geschickt, und diese führen die Operation lokal aus.

Entscheidungs-Kriterien:

- Objekt- vs. Parametergröße
- Rechenaufwand der Operation

6.1.1 Konsistenz replizierter Objekte

Definition von **Konsistenz** *(consistency)*:

Naive Definition: Kopien sind stets identisch

→ unerfüllbare Forderung

Sinnvolle Definition: unter Berücksichtigung von *Datenabstraktion:*
Objekt *verhält* sich so, als sei es *nicht repliziert*

→ immer noch starke Forderung –
für manche Anwendungen *unnötig stark*

Konsistenz z.B. erreichbar durch

- Sperren des gesamten verteilten Objekts *oder*
- verteilte Transaktion auf den Kopien

→ zu restriktiv und zu aufwendig

Stattdessen:

- lokale Sperrsynchronisation bei jeder Kopie
  (concurrency control)
- plus Konsistenzerhaltung mittels geeigneter *Rundrufe*
  (consistency control)

Wünschenswerte Eigenschaft 1: **Kausalitätstreue**:

Eine Folge *kausal abhängiger* Operationsaufrufe

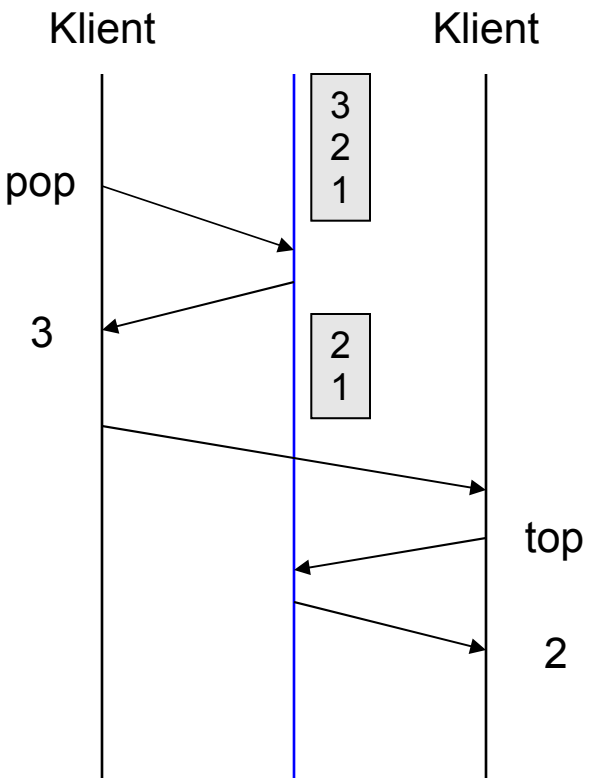
auf einem *replizierten* Objekt x

hat die gleichen Effekte/Ergebnisse wie eine entsprechende

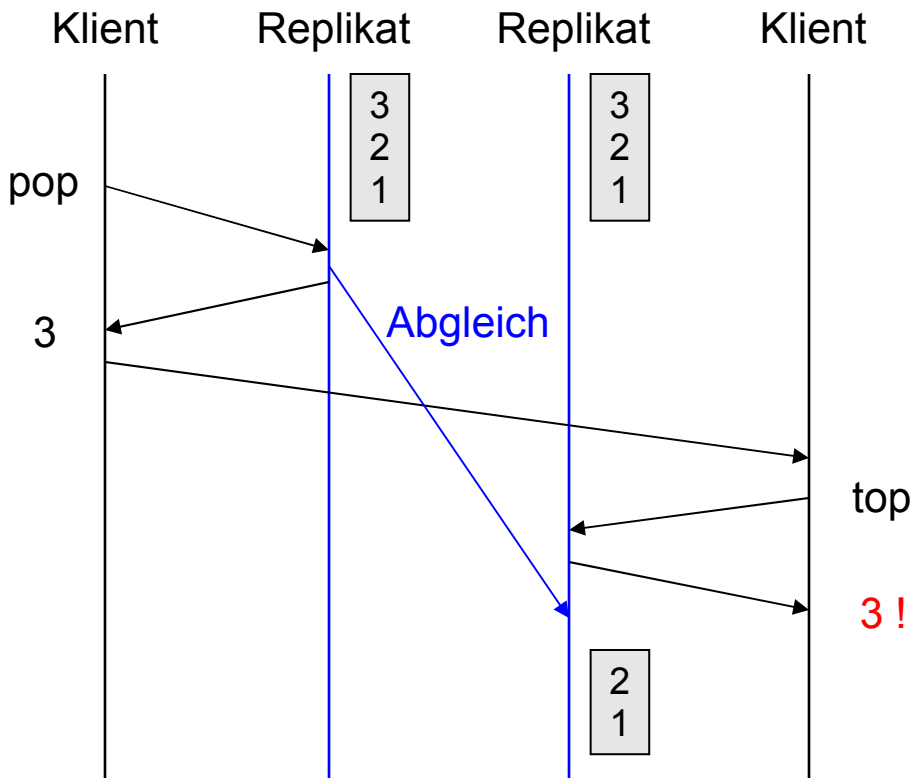
sequentielle Ausführung auf einem *nichtreplizierten* x

(wenn sonst keine weiteren Aufrufe im Spiel sind)

nicht repliziert



repliziert



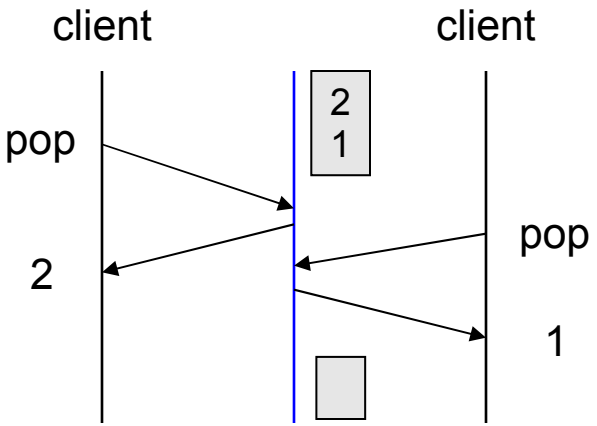
nicht kausalitätstreu

Wünschenswerte Eigenschaft 2: **Unabhängigkeitstreue**:

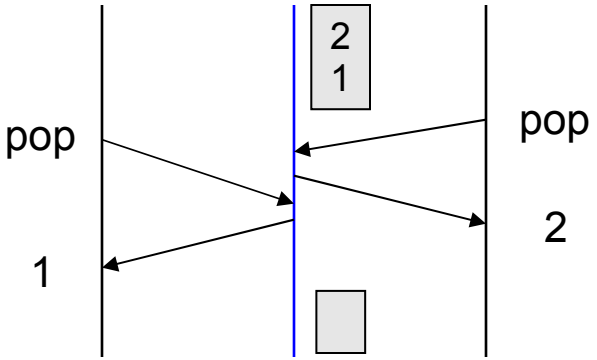
Effekt und Ergebnisse von *kausal unabhängigen* Operationsaufrufen auf einem *replizierten* Objekt x sind die gleichen wie bei einer *nebenläufigen* Ausführung auf einem *nichtreplizierten* x

(z.B. bei zwei Operationen a, b auf einem serialisierbaren Objekt x: „erst a, dann b" oder „erst b, dann a")

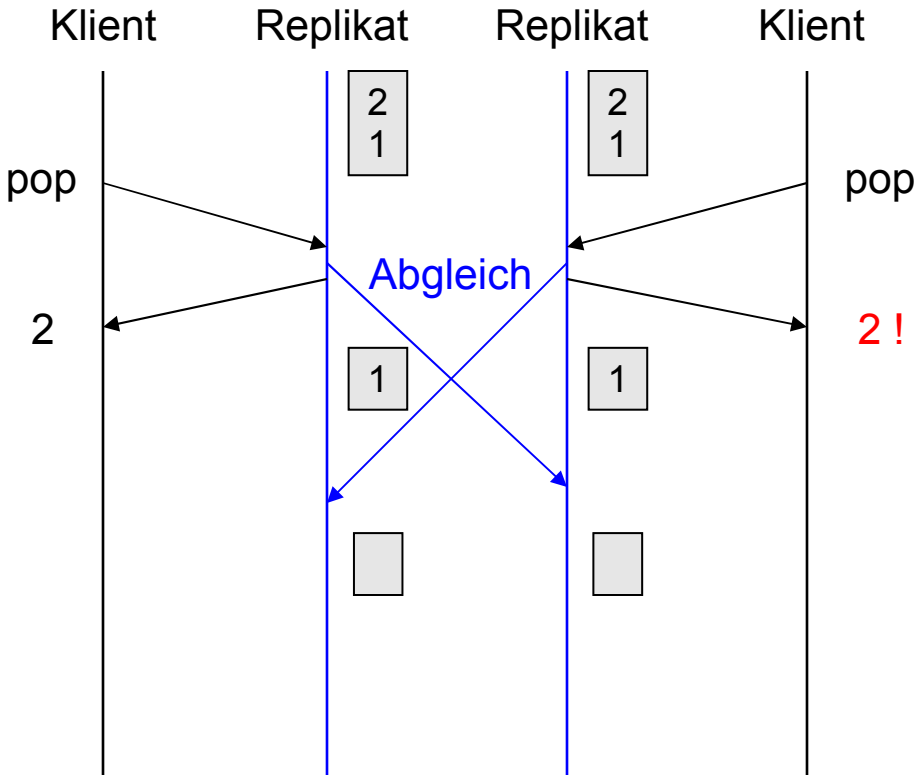
nicht repliziert



oder:



repliziert



nicht unabhängigkeitstreu