Karlsruher Institut für Technologie

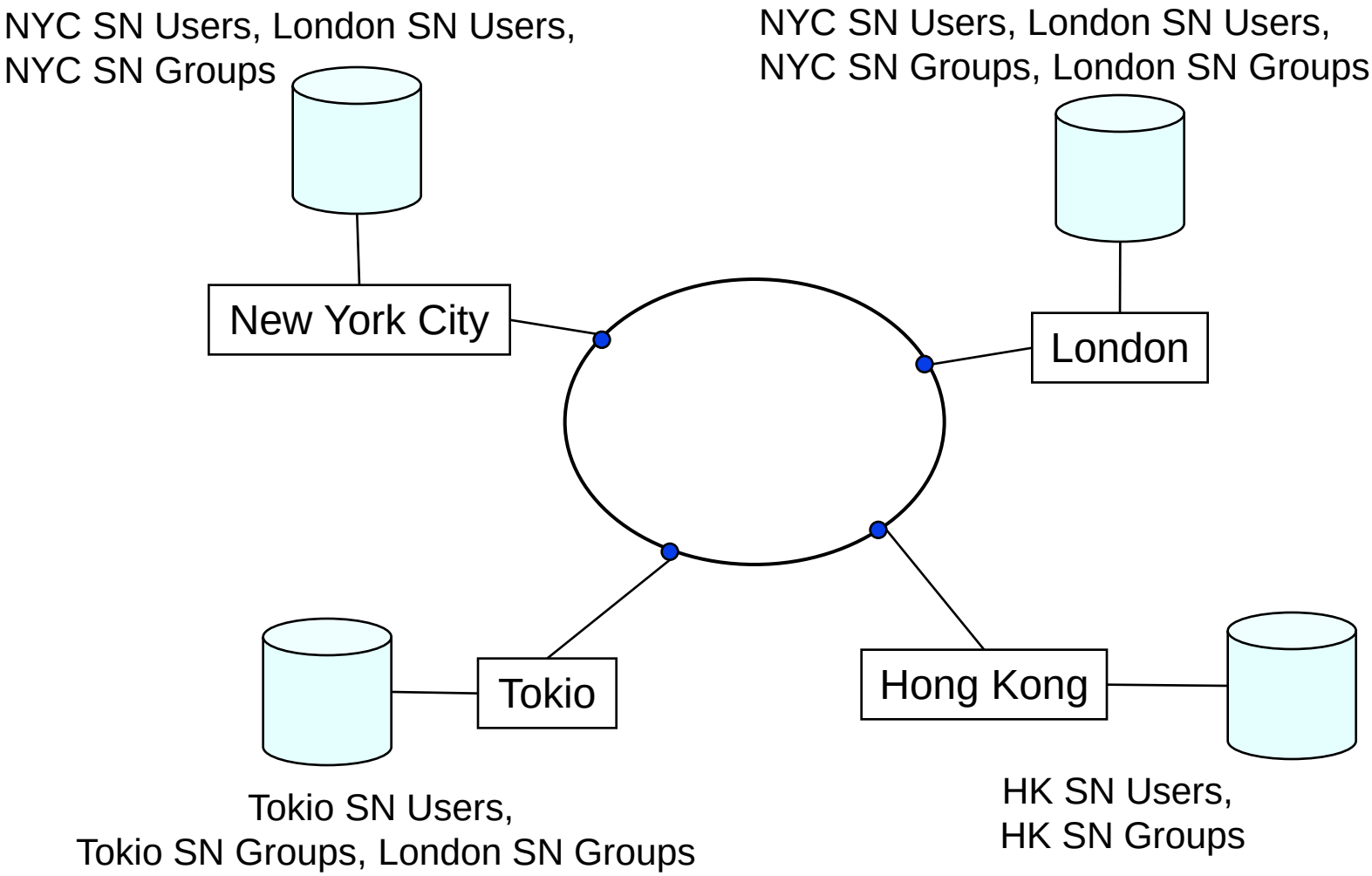
Datenbanksysteme

Kapitel 10: Konsistenz in Cloud-basierten Systemen

Lehrstuhl für Systeme der Informationsverwaltung, Fakultät für Informatik

Photo by Les Haines

Verteilung – Fragmentierung vs. Replikation



Verteilung – Vorteile

- Auf den ersten Blick,
  gilt sowohl für Fragmentierung als auch für Replikation:
  - Last (gemeint ist: Leselast) auf mehrere Knoten verteilt.
    Unverzichtbar für weltweit präsente Akteure
    (Social Networks, Online Handel).
  - Höhere Lokalität des Zugriffs. → Beschleunigung.
- Außerdem höhere Ausfallsicherheit.

- Für das Folgende nicht wesentlich,
  ob Anordnung der Daten transparent
  (typisch für Cloud-Infrastrukturen)
  oder nicht.

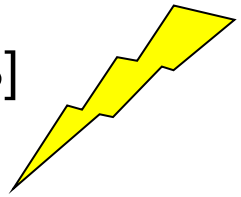
Verteilte Transaktionen – Beispiel

- T1: „SN User WB wechselt von Tokio nach London.“
  Operationen:
  - r_1[WB] (aus Tokio-Fragment)
  - $delete_1$[WB] (aus Tokio-Fragment)
  - w_1[WB] (in London-Fragment,
    auf Knoten NYC und London)
- T2: „Gibt es mehr SN User in Tokio oder London? Neuen User FK der ‚kleineren‘ Stadt zuordnen.“
  Operationen:
  - r_2[SNUSER] (aus Tokio-Fragment)
  - r_2[SNUSER] (aus London-Fragment,
    entweder Knoten NYC oder London)
  - w_2[FK] (ins Tokio- oder London-Fragment;
    im London-Fall auf Knoten NYC und London).

Erläuterungen zum Beispiel

- Operationen konfligieren,
  System muss Anordnung explizit festlegen (z. B. mit 2PL)
  oder zumindest überwachen.
- Anordnung der Transaktionen
  muss auf allen Knoten gleich sein.

- Tokio: $r_2[SNUSER] \rightarrow delete_1[WB]$
  London: $w_1[WB] \rightarrow r_2[SNUSER]$



- Es reicht nicht, wenn jeder Knoten für sich
  Serialisierbarkeit sicherstellt.

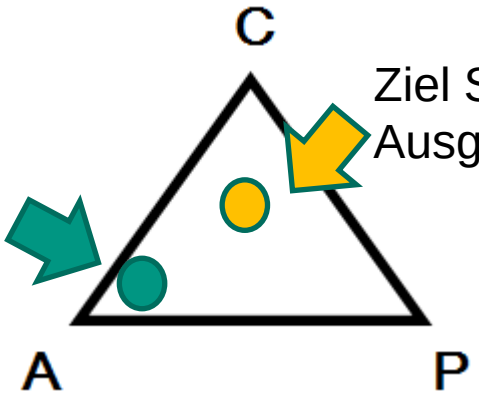
Probleme im verteilten Fall

- **Notwendigkeit von Koordination steht Skalierbarkeit im Weg.**
  - Anordnungsentscheidungen müssen widerspruchsfrei sein.
    Lässt sich nicht verteilen, ein Knoten muss sie fällen.
  - Skalierbarkeit schwierig bis unmöglich – ein Knoten muss alle Transaktionen kennen
    (und Informationen zu ihnen verwalten).
- **Mangelnde Verfügbarkeit.**
  - T1 und T2 greifen auf drei Knoten zu.
  - Jeder Knoten muss verfügbar und erreichbar sein, damit Transaktion erfolgreich.
  - Wahrscheinlichkeit, dass von n Knoten alle verfügbar
    $= 1 – (1 – p)^n$
    (p – Ausfallwahrscheinlichkeit eines Knotens)
    Z. B. p=0,01, n=20 $\Rightarrow$ WS=0,18
  - I. Allg. muss verteilte Transaktion auf langsamsten Knoten warten.
    Nachfolgende Transaktionen warten ebenfalls.

CAP Theorem

- ‚consistency', ‚availability', ‚partition tolerance'.
- ‚Konsistenz' bedeutet *single system images* von allen Datenobjekten.
  Anders formuliert: Man bekommt stets letzte Version jedes Datenobjekts zu sehen, egal auf welchem Knoten.
- CAP Theorem – sinngemäß in etwa:
  Wenn Netzwerkpartitionen möglich, sind hohe Verfügbarkeit und Konsistenz des Datenbestands i. Allg. unvereinbar.

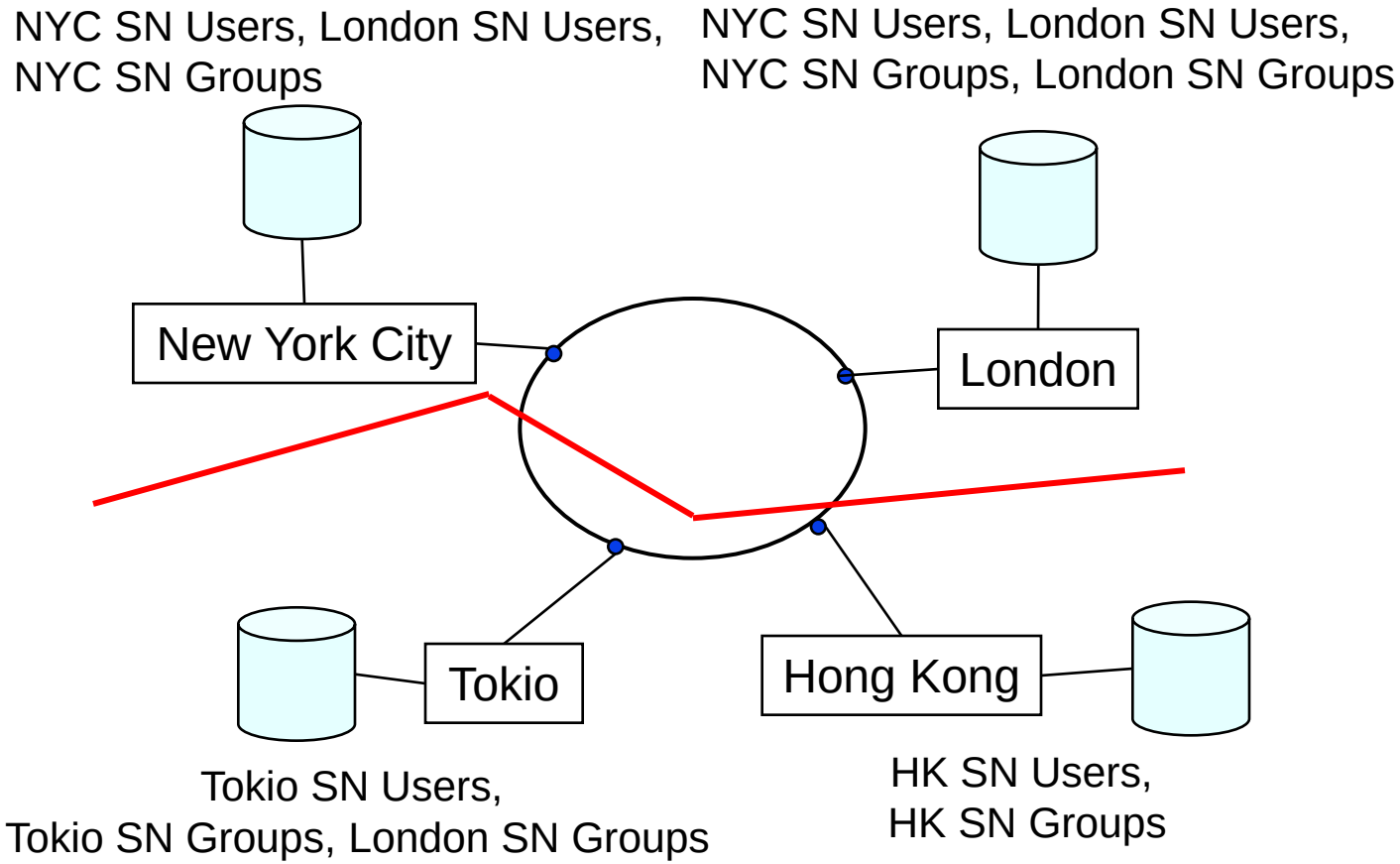
Ziel System 1: Hohe Verfügbarkeit = schnelle Antwortzeiten



Ziel System 2: Ausgewogene Mischung

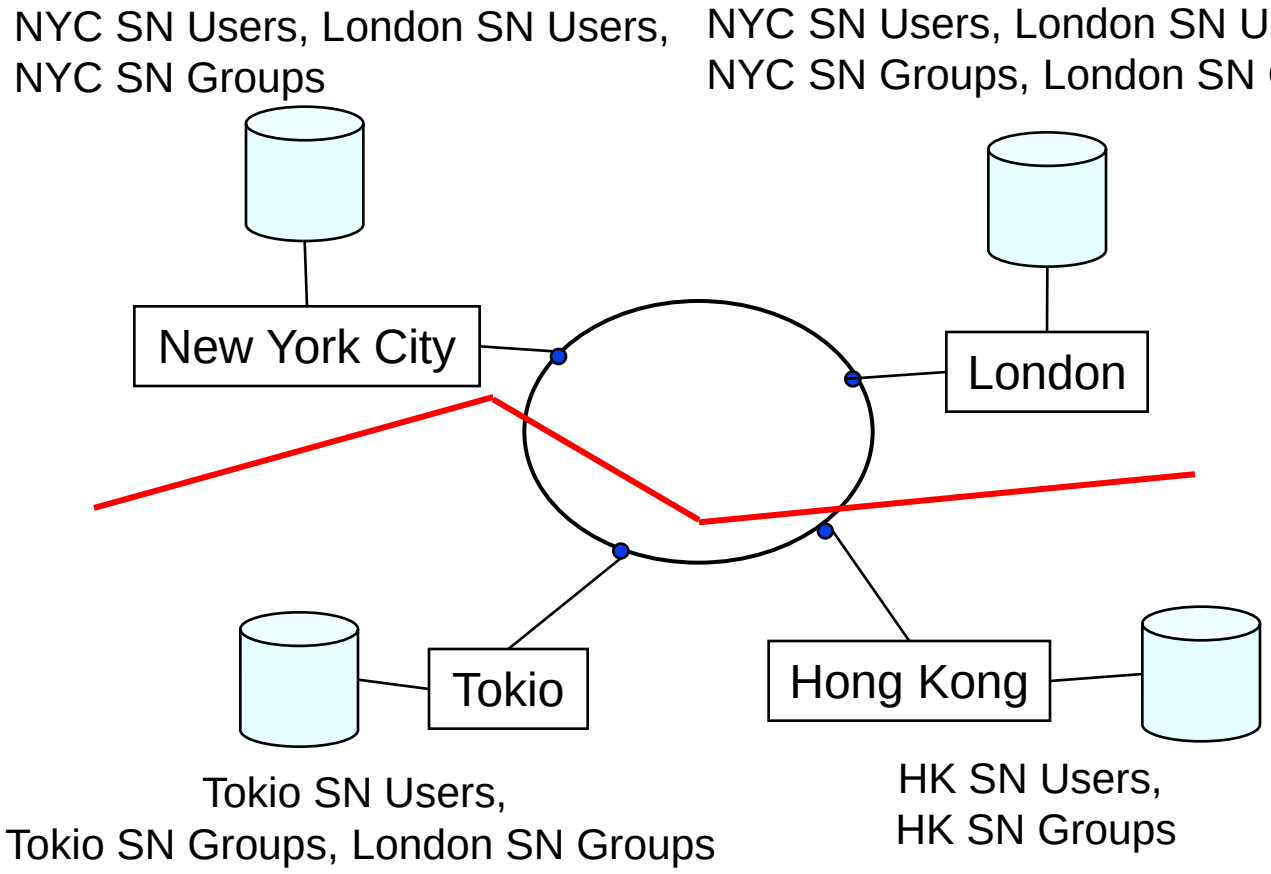
Fortsetzung Beispiel (1)

- Verfügbarkeit ohne Konsistenz – Illustration.
- Konsistenz i. Allg. nur, wenn System komplett verfügbar.



Fortsetzung Beispiel (2)

- Konsistenz nicht nur Thema bei Replikation, auch bei Fragmentierung.



Eventual Consistency (1)

- Takeaway:
  Bisher verwendeter Konsistenzbegriff wohl zu streng.

- Eventual Consistency – Definition:
  Wenn ab einem bestimmten Zeitpunkt keine Updates mehr, dann werden irgendwann alle Lesezugriffe auf ein Datenobjekt den gleichen Wert zurückliefern.

Eventual Consistency (2)

- Definition:
  Wenn ab einem bestimmten Zeitpunkt keine Updates mehr, dann werden irgendwann alle Lesezugriffe auf ein Datenobjekt den gleichen Wert zurückliefern.

- Prominentes Konzept;
  Werner Vogels von Amazon beispielsweise hat es propagiert.
- Diskussion: Sehr schwaches Konzept.
  - ‚irgendwann' kann nach allen für uns interessanten Zeitpunkten sein. Bis dahin ist alles möglich.
  - Keine Aussage dazu, welcher Wert zurückgeliefert wird.
- Keine *Safety* („nothing bad happens"), nur *Liveness* („es gibt Fortschritt").

Eventual Consistency (3)

- Alternative Definition daher:
  „… dann werden irgendwann alle Lesezugriffe den zuletzt geschriebenen Wert zurückliefern.“

facebook

E-Mail oder Telefon

Passwort

Anmelden

Angemeldet bleiben

Passwort vergessen?

Klemens Böhm
ist bei Facebook.

Um dich mit Klemens zu verbinden, registriere dich noch heute für Facebook.

Registrieren Anmelden

Klemens Böhm

Falscher Klemens Böhm? Bitte versuche es erneut

Klemens Böhm Suche

Andere Personen mit dem Namen Klemens Böhm

Klemens Böhm

Andere Personen mit einem ähnlichen Namen

Eventual Consistency – Fortsetzung Diskussion

- Warum sinnvoll?
- Beispiel:
  - Infrastruktur für Social Networks.
  - User schreiben Posts (z. B. Tweets).
  - Verteilte Verwaltung dieser Daten, z. B. wie im Beispiel zuvor.
  - Jetzt: Netzwerkpartition – grundsätzlich möglich:
    - Vorübergehend keine Postings möglich.
    - Rückmeldung an Benutzer erst, wenn alle relevanten Partitionen erreicht wurden.
  - Wohl besser:
    Follower bekommen Post zu sehen, sobald möglich – i. Allg. zu unterschiedlichen Zeitpunkten.
- Wie effizient zu implementieren?
  Thema weiterführender Vorlesung.

Photo by thinkpanama

Weitere Probleme ohne Datenbanken

– Folie aus einleitendem Kapitel –

- Isolation
  - Beispiel, „Bank-Szenario“:

Nummer	Inhaber	Stand
	Klemens	5000
	Gunter	200

  - Überweisung – zwei Elementaroperationen.
    - Abbuchung(Klemens, 500),
    - Einzahlung(Gunter, 500).
- *Isolation* – keine inkonsistenten Zwischenzustände werden sichtbar.
- Transaktionen.
  Programm oder Folge von Kommandos, die in Interpreter eingegeben werden.

Beispiel, Geldautomat

- Menge der Geldautomaten weltweit
  ist hochgradig verteiltes System.
- Angenommen, keine Konsistenz im klassischen Sinn.
  - Partitionierung des Geldautomat-Netzes.
  - Zwei zeitgleiche Abhebungen
    an unterschiedlichen Automaten.
    (Angenommen, mehrere Karten für ein Konto
    bzw. neue Technik ohne Karte.)
- Sehr schlimm – gemäß bisheriger Argumentation.
- Aber: Konto wird ‚nur' überzogen.
  Bank kann damit umgehen.

Beispiel, Geldautomat – Fortsetzung

- Angenommen, Geld wäre wirklich weg…
- Angenommen, das passiert sehr selten.
  (Z. B. jedes 10^8-te Mal, außerdem Limit bei Abhebung.)
- D. h. Kosten sind quantifizierbar.
- Anderer Fall – starke Konsistenz,
  keine Abhebungen während Partitionierung.
  - Implementierung und Betrieb
    von System mit starker Konsistenz wäre teuer.
  - Unzufriedene Kunden.
    (Lässt sich ebenfalls in Geld ausdrücken.)
- Takeaway: Alles ist relativ.
  Entwurfsentscheidung, anwendungsspezifisch.

Photo by Álvaro Ibáñez

Anderes Beispiel: Online-Handel

- Artikel kann von Hubwagen fallen,
  nachdem er bestellt wurde, vor Auslieferung.
  Erzwingen, dass Wirklichkeit
  mit Datenbankinhalt übereinstimmt,
  i. Allg. sowieso nicht möglich.
- Online Händler weiß, wie er damit umgehen muss
  (Entschuldigungen, Präsente usw.).
- Gleiches Prozedere im Prinzip anwendbar bei Inkonsistenz.
  (Nicht vorhandener Artikel wird bestellt,
  und Bestellung wird bestätigt.)

Kompensationen

- Konsistenzverletzungen ziehen i. Allg. sogenannte *Kompensationen* nach sich. (Entschuldigungen, Präsente, Geld wieder eintreiben usw.)
- Entsprechende Prozesse müssen entworfen und realisiert werden („Entschuldigungsdesign“). Manchmal trivial/einfach, i. Allg. jedoch nicht.
- Jedoch: In manchen Fällen ist Anordnung der Operationen egal. Es muss nie kompensiert werden.
- Beispiel:
  - Berechnung, die kommutativ und assoziativ ist.
  - Obwohl sie aus Lese- und Schreiboperationen besteht.
  - Egal, in welcher Reihenfolge Argumente daherkommen.

Schlussbemerkungen

- Klassische starke Konsistenz im verteilten Fall (z. B. Cloud-Infrastrukturen) teuer bzw. gar nicht erreichbar.
- Schwächere Konsistenzbegriffe – oft ausreichend (aber nicht immer).
- Aktuelle Forschung:
  - Wie möglichst nah an klassischen Konsistenzbegriff herankommen – ohne jene Nachteile?
  - Wie sehen effiziente Implementierungen aus?

Mögliche Prüfungsfragen, beispielhaft

- Geben Sie die Probleme mit dem klassischen, starken Konsistenzbegriff im verteilten Fall in eigenen Worten wieder.
- Bekommt man mit Eventual Consistency irgendeine Form von Safety? Begründen Sie Ihre Antwort.
- Warum kann man im Bank-Kontext in manchen Situationen doch auf starke, klassische Konsistenz verzichten?
- Geben Sie – alternativ zu dem Beispiel in der Vorlesung – ein weiteres Beispiel für eine Folge von Operationen, deren Anordnung egal ist.

Literatur

- Peter Bailis, Ali Ghodsi: Eventual Consistency Today: Limitations, Extensions, and Beyond. ACM Queue. Volume 11 Issue 3, März 2013.