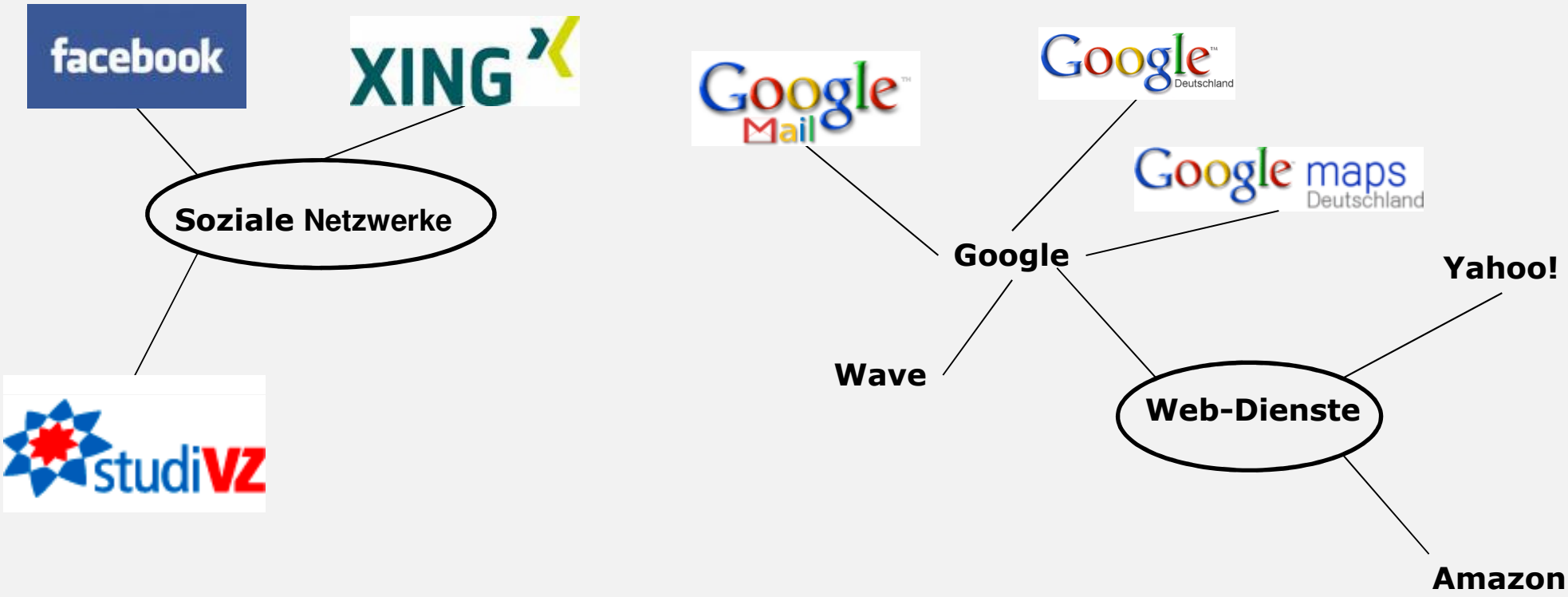
Parallele Cloud-DBS: Aufbau und Implementierung

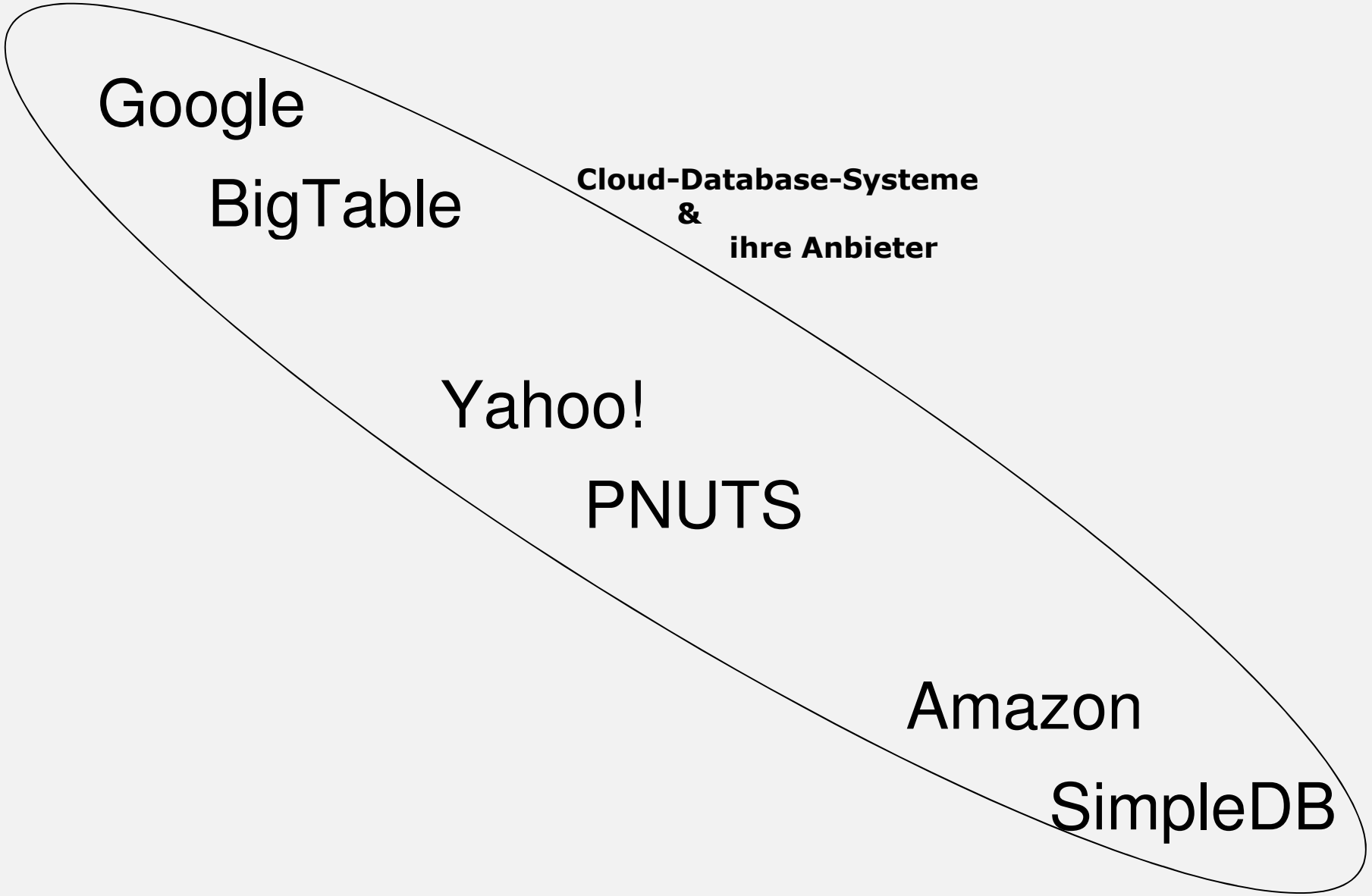
Markus Weise

Inhalt:

1. Einleitung
2. Google‘s Bigtable
3. Yahoo!‘s PNUTS
4. Zusammenfassung
5. Quellen

Wir leben in einer Informationsgesellschaft, in der massenweise Daten anfallen:

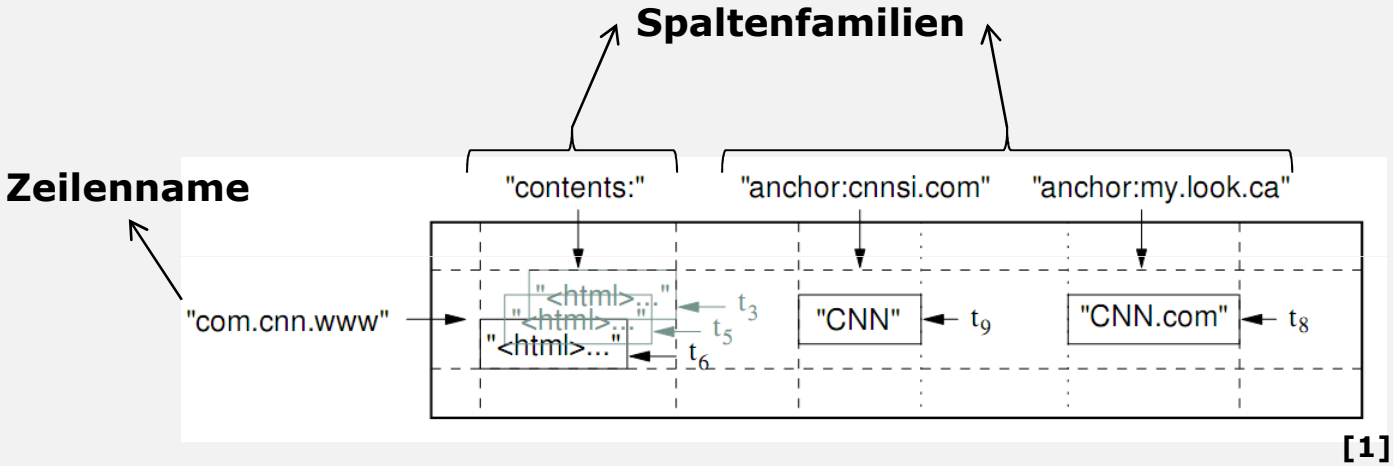



Google
BigTable
Cloud-Database-Systeme
&
ihre Anbieter
Yahoo!
PNUTS
Amazon
SimpleDB

- **BigTable:** → verteiltes Datenbanksystem, speziell für den Umgang mit sehr großen Datenmengen (Petabyte)
- BigTable wird von über 60 Google-Apps genutzt.
- Daten werden über Zeilen und Spalten (Strings) indiziert.
- Daten werden als Byte-Array abgelegt.
- Speicherung der Daten innerhalb von Google File System (GFS)

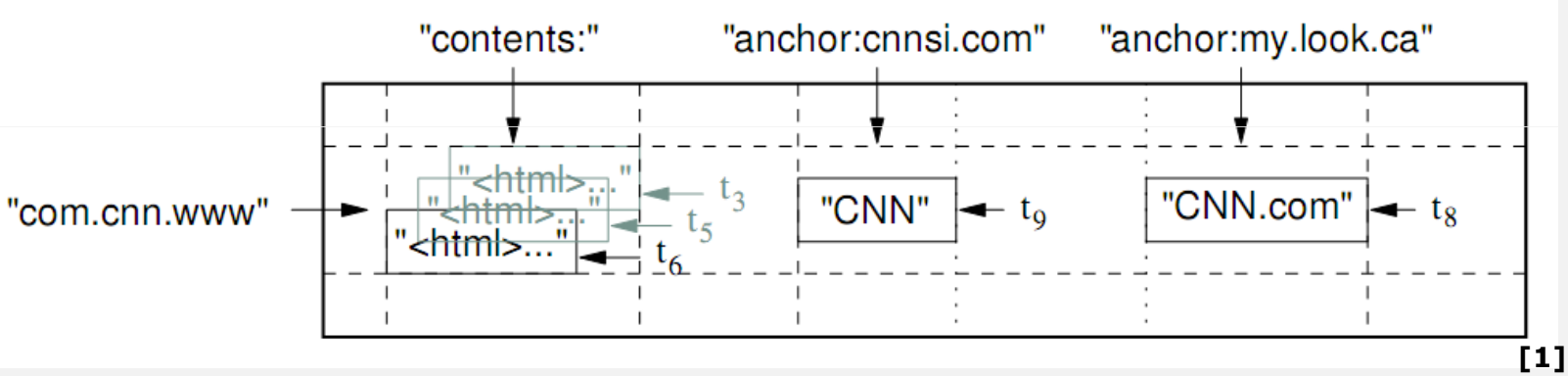
- **Datenmodell**
  - Indizierung der Daten über:

    Zeile(String), Spalte(String), Zeitstempel(int64)
  - Daten als byte-array gespeichert



[1]

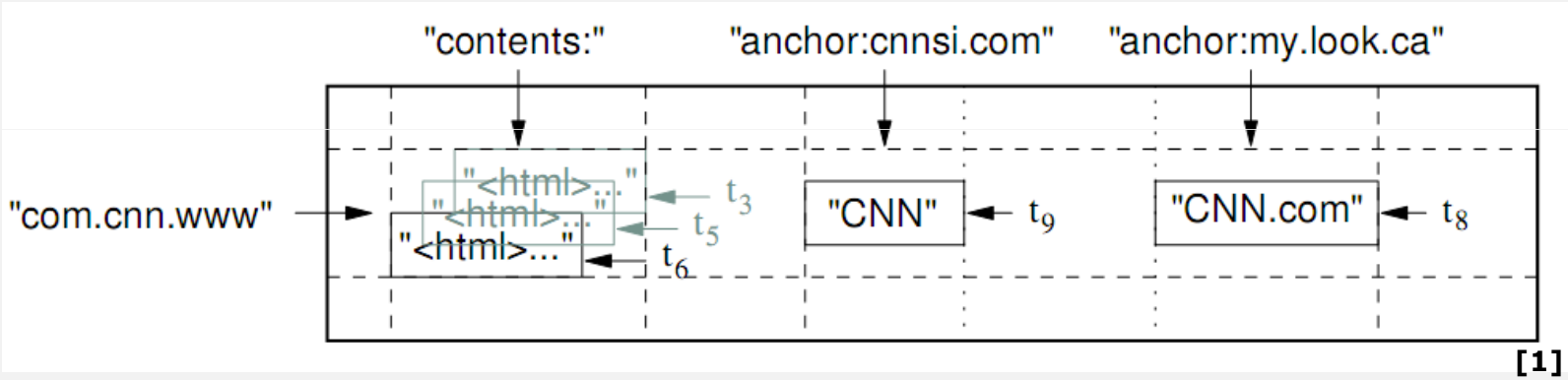
- **Datenmodell** – Zeilen
  - Strings mit maximal 64kB Länge
  - Lese-/Schreibzugriffe auf eine Zeile sind atomar
  - Alphabetische Sortierung innerhalb der Tabelle
  - Aufsplitten der Tabelle in Zeilenbereiche →Tablets



[1]

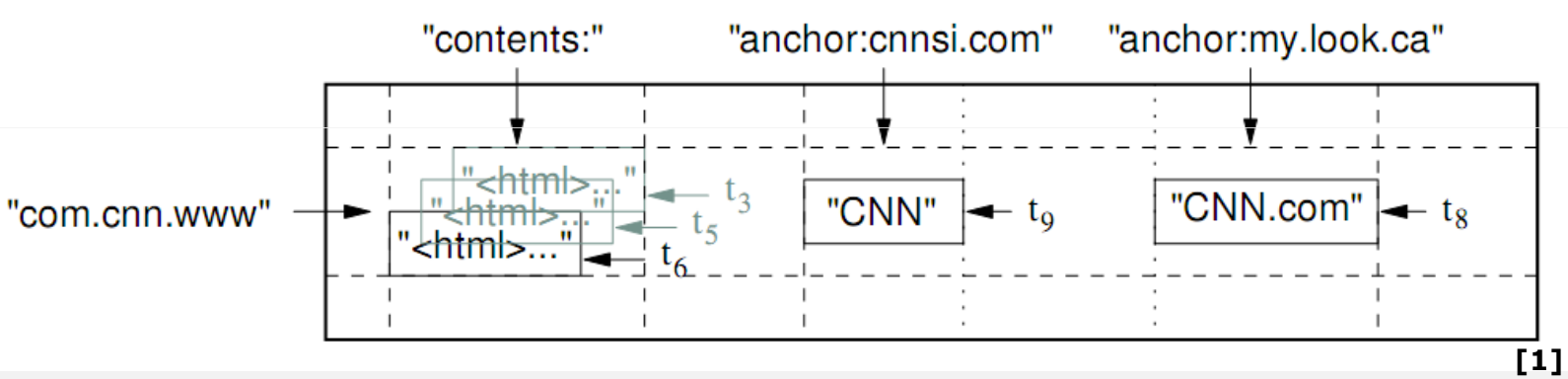
- **Datenmodell** – Spalten
  - Gruppierung in Spalten-Familien

    „SpaltenFamilie:SpaltenName“
  - Spalten-Familien und Spalten-Namen → Strings
  - Zugriffskontrolle und Speicherung als Spalten-Familien



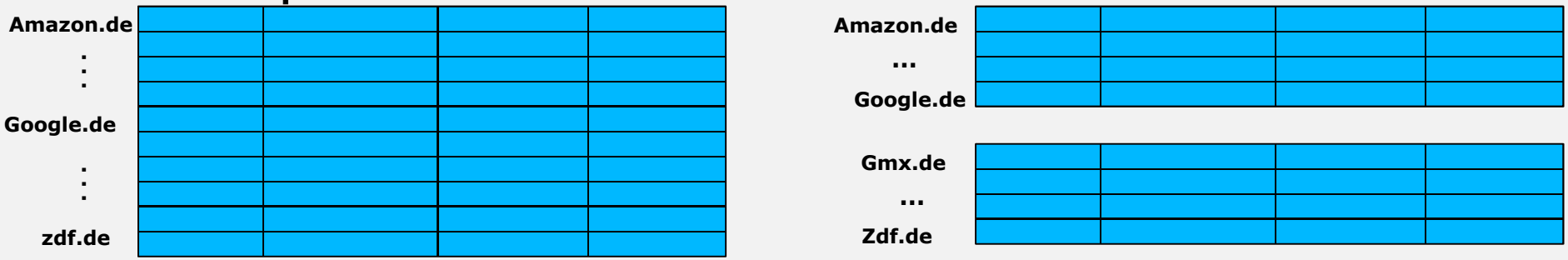
[1]

- **Datenmodell** – Zeitstempel
  - Versionierung von Objekten
  - Zeitstempel → 64 bit – integer
  - Garbage-Collection entsorgt zu alte Versionen

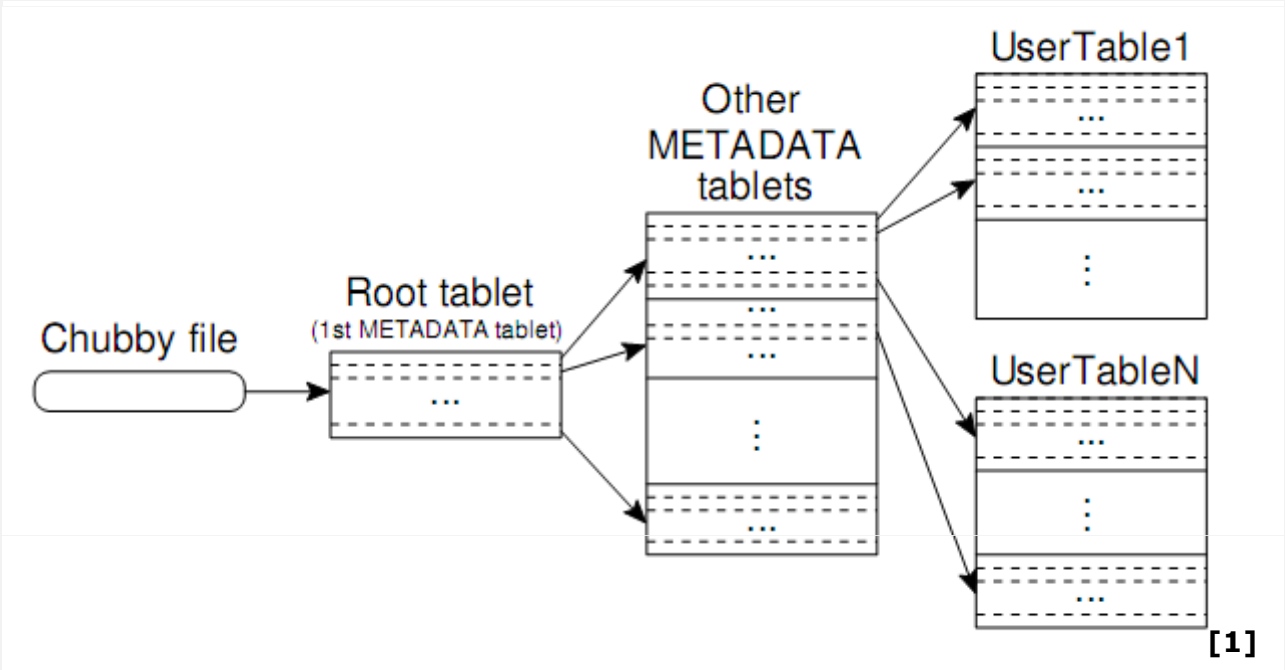


[1]

- **Tablets**
  - Tablets enthalten zusammenhängende Zeilenbereiche
  - ca. 100 – 200 MB Daten pro Tablet
  - ca. 100 Tablets pro Server
    - → Schnelles Recovery
    - → Load-Balancing
  - Aufspalten von Tablets

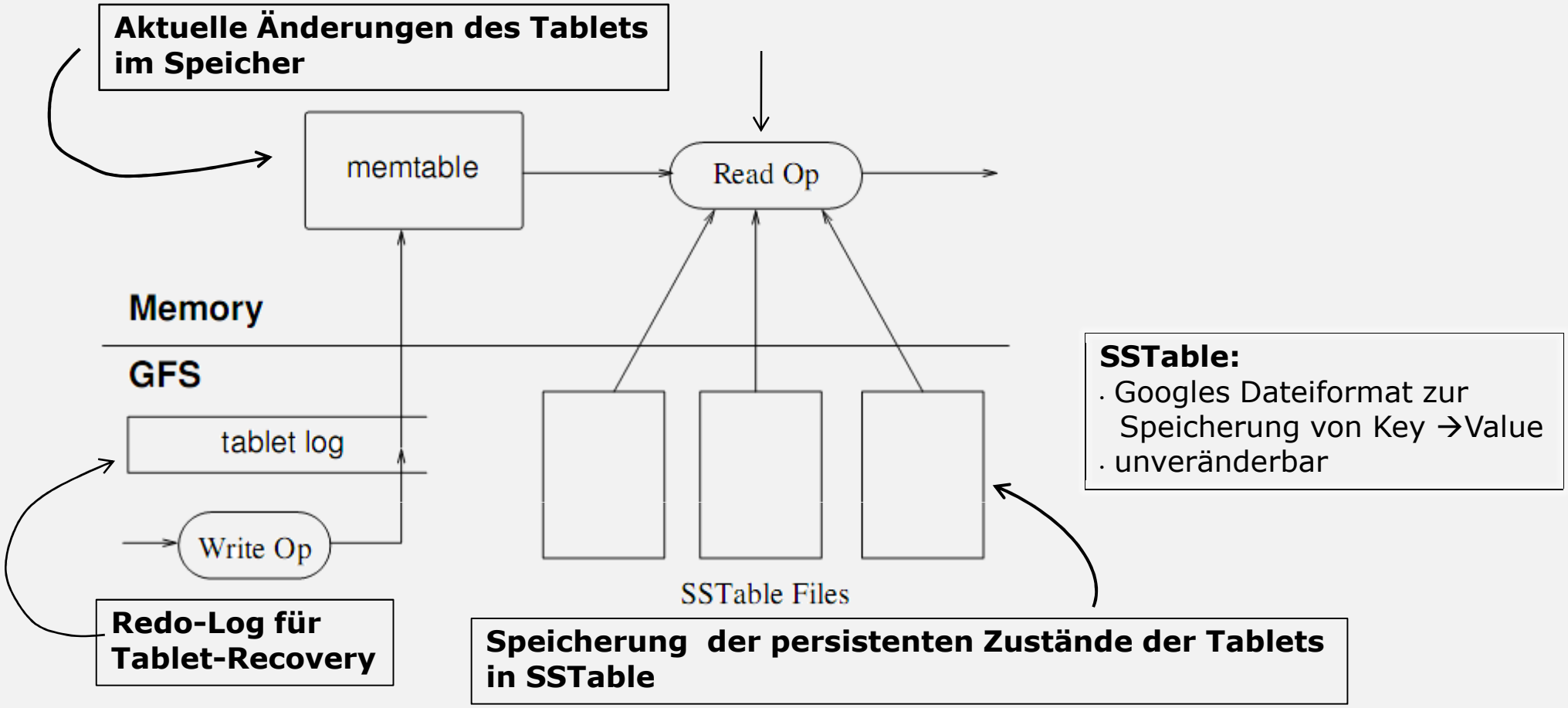


- **Tablets** – Auffinden
  - 3 stufiges hierarchisches System

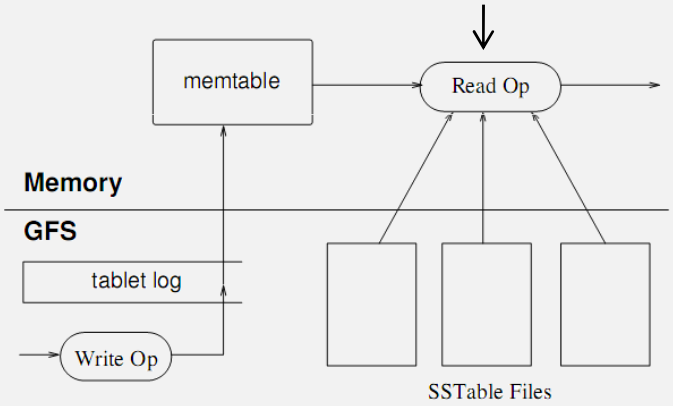


[1]

- **Tablets** – Zugriff/Speicherung



- **Tablets** – Zugriff/Speicherung (fort.)
  - Schreiben von Daten in Tablets
    1. Überprüfen der Validität der Anfrage
    2. Authentifizierung des Absenders
    3. Schreiben in Log-File
    4. Schreiben der Änderungen in Memtable



  - Lesen von Daten aus Tablets
    1. Überprüfen der Validität der Anfrage
    2. Authentifizierung des Absenders
    3. Lesen der Daten aus Memtable und SSTable (merged)

- **Tablets** – Zugriff/Speicherung (fort.)
  - Wenn Memtable Größenschwellwert überschreitet:
    - Memtabel wird eingefroren
    - Neue Memtable wird erstellt
    - Inhalt der Memtable wird in SSTable gespeichert

    →minor Compaction
  - Wenn Anzahl der SSTable-Dateien zu groß wird:
    - Zusammenfassen einiger SSTables zu einer neuen SSTable
    - Löschen der alten SSTable

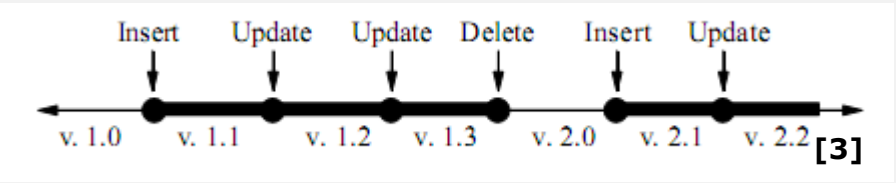
    →major Compaction

- **Chubby:**
  - Lock-System für lose gekoppelte, verteilte Systeme
  - Besteht aus Master-Prozess und mehreren Client-Prozessen
  - Ablage von Meta-Daten
    - Schema-Informationen
    - Meta-Data-Tabelle
  - Systemrelevante Managementaufgaben:
    - Konsistenzerhaltung zw. Chubby-Replika
    - Feststellen von Serverausfällen
    - Einbeziehen von neuen Servern
    - Logging und Recovery
  - Bei Ausfall von Chubby → Ausfall von BigTable

- **PNUTS:** → paralleles und (geographisch) verteiltes Datenbanksystem
- Speziell für Yahoo!‘s Web-Applikationen
- Geringe Latenzzeiten durch geographisch verteilte Datenbank-Replika
- Quasi-Konsistenz zwischen den Replika

- **Datenmodell**
  - vereinfachtes relationales Datenmodell
    - → Records (Zeilen) mit Attributen (Spalten)
    - → blob als Datentyp
  - Einfügen von Spalten jeder Zeit möglich
  - Keine Unterstützung für Datenintegrität
  - Keine Joins, groub-by,…

- **Konsistenzmodell**
  - Reihenfolge der Updates wird garantiert (Serialisierbarkeit)
  - Eine Replika eines Datensatzes wird als Master ernannt (Replika mit max. Work-Load)
  - Alle Veränderungen am Datensatz werden zum Master weitergeleitet



[3]

- **Konsistenzmodell** – fort.

Read-any:
· Lesen einer Version aus der Datenbank
· Datensatz ist nicht zwingend der aktuellste
→ unrepeatable read

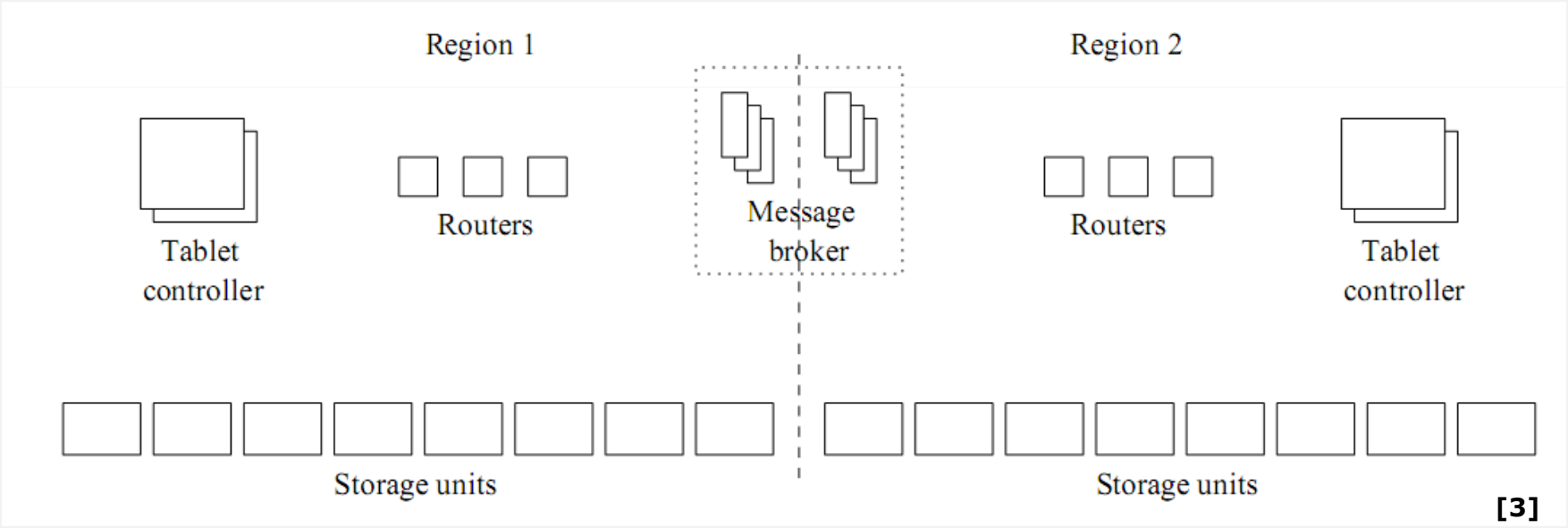
Read-latest:
· Lesen der aktuellsten Version
→ kein unrepeatable read

write:
· Schreiben eines Datensatzes
→ Volle ACID - Eigenschaften

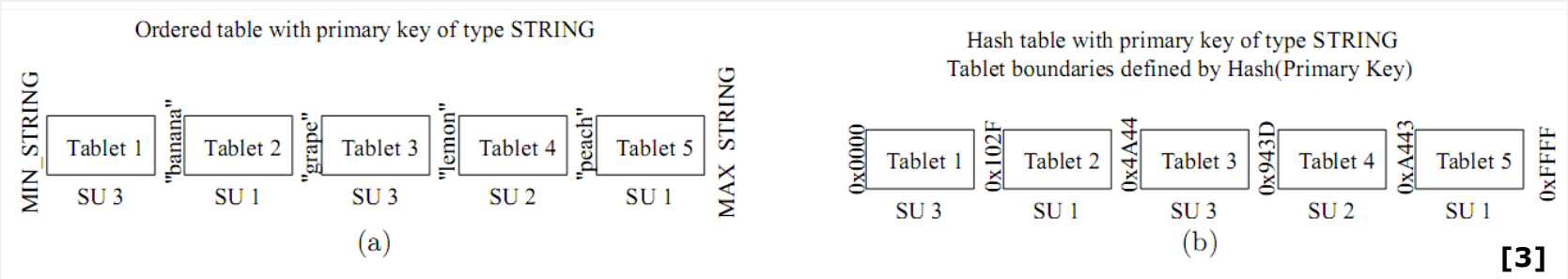
Read-critical<Version>:
· Lesen eines Datensatzes, der mind. der Version entspricht
→ kein unrepeatable read

Test-and-set-write<Version>:
· Schreiben eines Datensatzes, wenn dieser der Version entspricht
· Zwei konkurrierende Aufrufe werden serialisiert

- **Systemarchitektur**



[3]



[3]

- **Yahoo! Message Broker** (YMB)
  - Topic-based publish/submit System
  - Updates gelten als commited, wenn sie dem YMB übergeben wurden
  - Messages werden asynchron an Replikas versendet
  - YMB verantwortlich für Konsistenz & Recovey

- **YMB** – Recovery
  - PNUTS besitzt kein Log von DB-Transaktionen
  - Recovery in 3 Schritten:
    1. Tablet-Controller fordert Kopie einer Remote-Replika an
    2. Senden einer „checkpoint message“ an YMB
    3. Kopieren der Remote-Replika
  - Tablet-Grenzen sind gleich auf allen Replika (2PC)

- Zusammenfassung:
  - Datenmodell: → beide DBS zeigen nach außen ein (quasi-) relationales Datenmodell
    →Horizontale Partitionierung der Tabellen in „Tablets“
  - Konsistenz: → BigTable: Konsistenz der Replikas ist immer gewährleistet
    → PNUTS: nach Update besteht die Möglichkeit, alte Daten zu lesen
  - Recovery: → BigTabel: Redo
    → PNUTS: Recovery nur durch Kopieren (evtl. veralteter) Replikas, kein Logging von Transaktionen
  - Redundante Speicherung:
    → BigTabel: innerhalb von GFS (3 Replikas)
    → PNUTS: redundante Speicherung in global verteilten Datacenters

5. Quellen

[1] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Grube:
Bigtable: A Distributed Storage System for Structured Data.
OSDI'06: Seventh Symposium on Operating System Design and Implementation;
Seattle, WA, November, 2006

[2] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung:
The Google File System.
19th ACM Symposium on Operating Systems Principles;
Lake George, NY, October, 2003.

[3] Cooper, B.F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H., Puz, N., Weaver, D., Yerneni, R.:
PNUTS: Yahoo!'s Hosted Data Serving Platform
Very Large Data Base;
Auckland, NZ (2008)

[4] Mike Burrows:
The Chubby Lock Service for Loosely-Coupled Distributed Systems
OSDI'06: Seventh Symposium on Operating System Design and Implementation;
Seattle, WA, November, 2006

Danke für die Aufmerksamkeit!

Fragen?