BigTable

11.12.2012
Else

Einführung

- Distributed Storage System
- im Einsatz bei Google (2006)
- speichert strukturierte Daten
- “petabyte-scale”, > 1000 Nodes
- nicht relational, “NoSQL”
- setzt auf GFS auf

Die BigTable

- dünnbesetzte, sortierte, multidimensionale Map
- bt[row: str; column: str; time: int] → value

- Row: bis zu 64kb, lexikographisch sortiert
- Value: uninterpretiertes Byte-Array

Columns

- bestehen aus Column Family und Qualifier

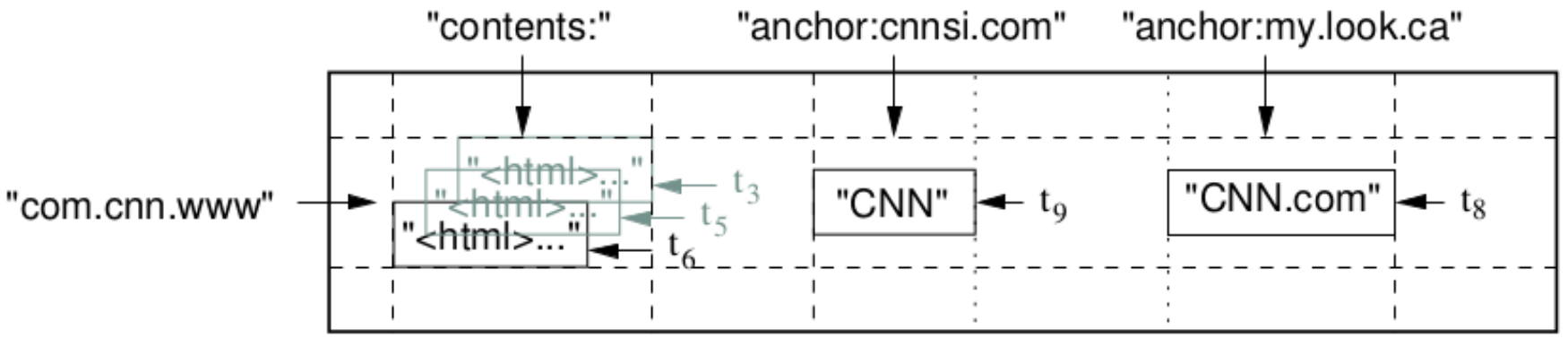
  column := family:qualifier

- Family muss vorher angelegt werden
- Daten in Column Family vom selben Typ
  - werden zusammen komprimiert
- wenige Column Families, unbegrenzt viele Qualifier
- Grundlage für ACL

Timestamps

- jede Zelle kann beliebig viele Versionen haben
- Versionen: 64-bit Integer
- frei wählbar oder vom System vergeben
- in umgekehrter Reihenfolge gespeichert
- Garbage Collection:
  - letzte n Versionen (Anzahl)
  - neueste Versionen (Alter)

Beispiel: Webtable



- “com.cnn.www” Row Key
- “contents” Column Family
- “anchor” Column Family
- “cnnsi.com” Column Qualifier
- t_n Versionen

API: Schreiben

```
Table *T = OpenOrDie(“/bigtable/web/webtable”);


RowMutation r1(T, “com.cnn.www”);
R1.Set(“anchor:www.c-span.org”, “CNN”);
R1.Delete(“anchor:www.abc.com”);
Operation op;
Apply(&op, &r1);
```

API: Lesen

```
Scanner scanner(T);
ScanStream *stream;
Stream = scanner.FetchColumnFamily(“anchor”);
stream->SetReturnAllVersions();
Scanner.Lookup(“com.cnn.www”);
for (; !stream->Done(); stream->Next()) {
printf(“%s %s %11d %s\n”,
     scanner.RowName(),
     stream->ColumnName(),
     stream->MicroTimestamp(),
     stream->Value());
}
```

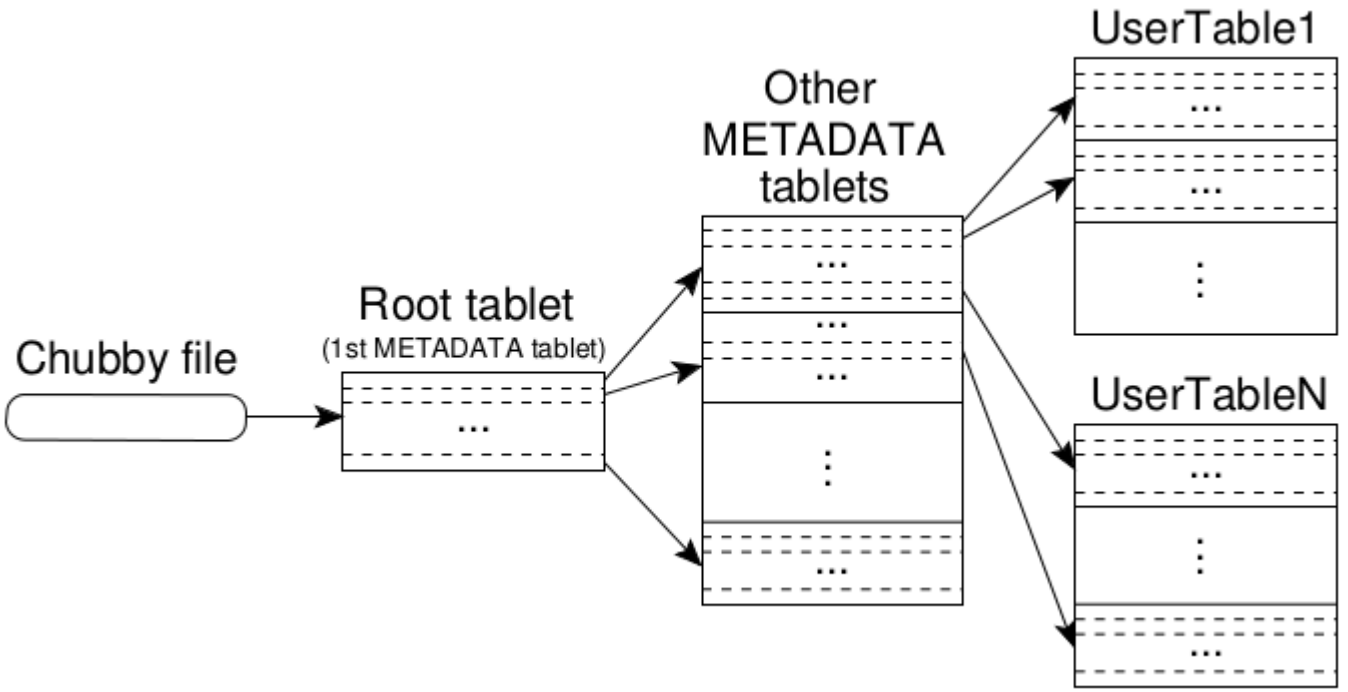
SSTables

- “Sorted String Table”
- interne Datenstruktur für Tablets
- persistente, geordnete, immutable Map:
  sstable[key: byte] → value: byte
- enthält Datenblöcke (typischerweise 64k)
- Blockindex am Ende der Tabelle (im Speicher)

Tablets

- Tablet: Row-Range, verteilt auf Nodes (vertikale Fragmentierung)
- zu Beginn: nur ein Tablet
- Tablet Server können dynamisch hinzugefügt bzw. entfernt werden (vom Master)
- jeder Tablet Server hat bis zu tausend Tablets
- Clients lesen und schreiben auf Tablet Servern

Tablet Location

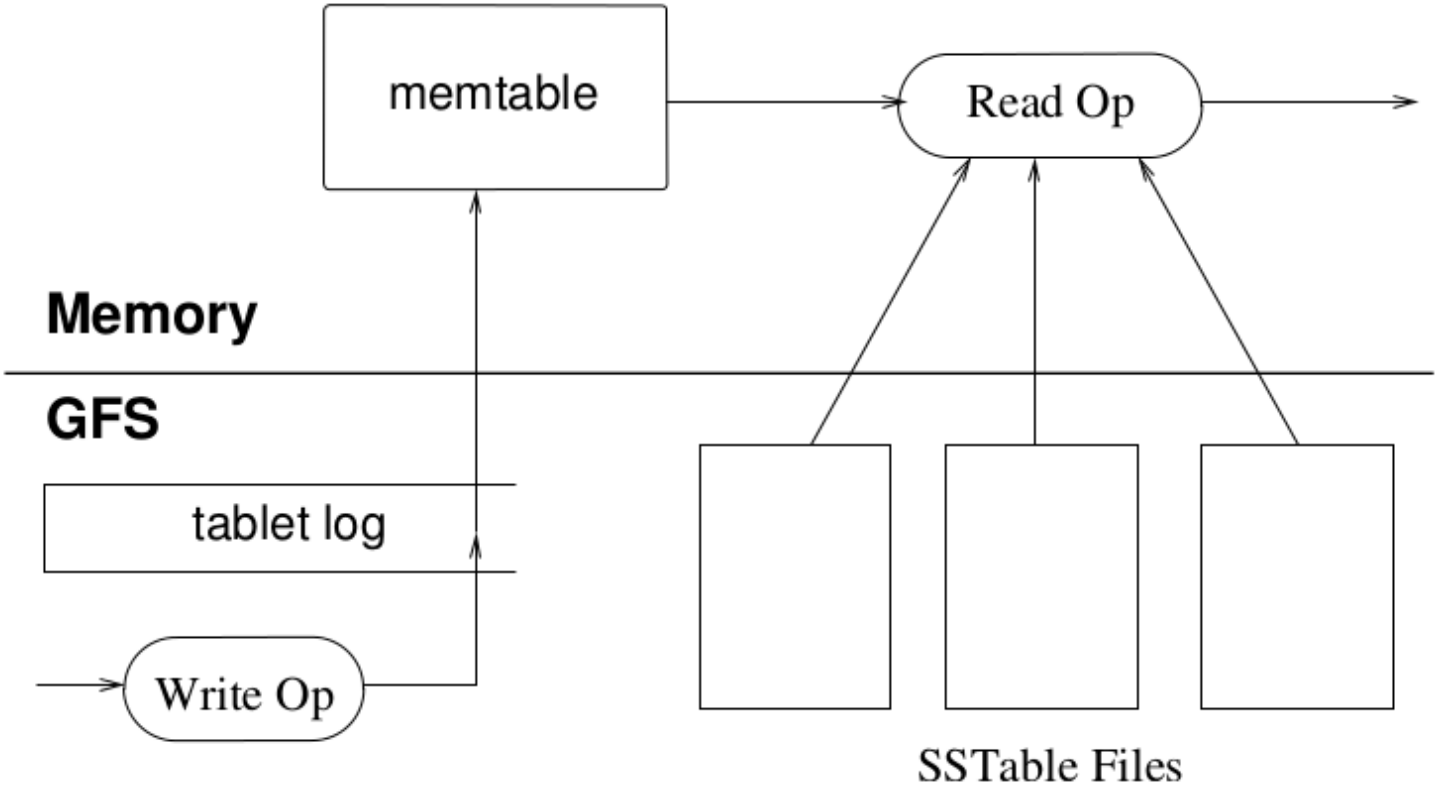


- Root Tablet: enthält alle Tablet Locations
- wird nie fragmentiert (nie mehr als drei Level)
- Metadata wird vom Client gecached, Prefetching

Tablet Assignment

- jedes Tablet genau einem Server zugeordnet
- Tablet Server registrieren sich in Lock Service
- wenn ein Tablet nicht zugeordnet ist, sucht der Master einen passenden Server heraus
- wenn Master abstürzt, sucht neuer Master alle Tablet Server heraus, und scannt METADATA
- Splits werden von Tablet Server erledigt

Tablet Serving



“Leckerbissen” (1)

- Locality Groups
  - Column Families bilden jeweils eine Locality Group
  - können als “in-memory” markiert werden
- Compression
  - Locality Groups können komprimiert werden
  - Komprimierung erfolgt blockweise
  - wird von Entwickler festgelegt
  - Kompressionsrate: ~10:1

“Leckerbissen” (2)

- Caching
  - Scan Cache: serverseitiger key-value Cache
  - Block Cache: SSTable Block Cache
- Bloom Filter
  - ermöglich Vorhersage, ob Key/Column vorhanden
  - Wenn nein: mit Sicherheit nicht; wenn ja: vielleicht
  - kann von Anwendung aktiviert werden
- ...

Open Source

- Hadoop: Framework für verteilte, skalierbare Anwendungen
- HDFS: Hadoop Filesystem
- HBase: freie BigTable Implementierung
- Disclaimer: alles Java ;-)

- HyperTable

Quelle

- Illustrationen und Infos aus:

  Chang, F., Dean, J., & Ghemawat, S. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Database Systems. Retrieved from http://dl.acm.org/citation.cfm?id=1365816

Nicht weiter erwähnt…

- Locking
- Transaktionen (Commit-Logs)
- Performance Evaluation

KTHXBYE