Wide column-stores für Architekten

Andreas Buckenhofer
Daimler TSS GmbH
Ulm

Schlüsselworte
Big Data, Hadoop, HBase, Cassandra, Use Cases, Row Key, Hash Table

NoSQL Datenbanken
In den letzten Jahren wurden NoSQL-Datenbanken immer beliebter, um den Anforderungen nach hoher Verfügbarkeit, Ausfallsicherheit, Performanz, usw zu genügen. Zurzeit etablieren sich vier verschiedene Typen

- Key Value Stores
  Das Datenmodell ist sehr einfach mit Paaren aus einem eindeutigen. Schlüssel und einem zugeordneten Wert bzw. einer Werteliste. Der Zugriff auf die Daten erfolgt ausschließlich über den Schlüssel. Idealerweise werden alle Daten im Cache gehalten, um schnelle Lesezugriffe zu ermöglichen (Caching).
- Document Stores
  Die Applikation legt XML-ähnliche Strukturen (JSON, BSON) in der Datenbank ab. Die Datenbank interpretiert das Modell nicht („schemalos“).
- Wide-column Stores
  Daten sind spaltenweise organisiert als Schlüssel und Werte („columns“). Diese Daten können durch übergeordnete Schlüssel weiter gruppiert werden („column family“).
- Graph DBs
  Das Datenmodell beinhaltet Elemente für die Modellierung von Knoten und Kanten sowie deren Eigenschaften. Die gesuchten Informationen sind weniger die Daten selbst, sondern die Verknüpfungen zwischen den Daten.

Im Big Data-Umfeld wurden von allem die Wide-column Stores bekannt (HBase, Cassandra, Google BigTable), da diese Datenbanken riesige Datenmengen verarbeiten können und dabei Antwortzeiten im Millisekundenbereich liefern.

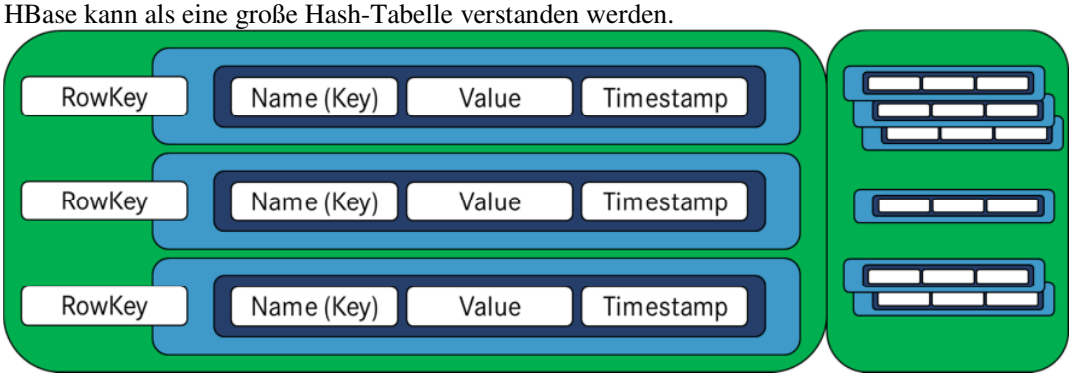
HBase ist bekannt als die NoSQL-Datenbank von Hadoop. Die üblichen Distributionen von Hortonworks, Cloudera, u.a. enthalten dementsprechend HBase. Cassandra dagegen wird Standalone betrieben. Cassandra wurde ursprünglich bei Facebook entwicklet, danach an Apache übergeben. Kommerziellen Support bietet z.B. DataStax.

	HBase	Cassandra
Infrastuktur	Wird i.d.R. mit Hadoop verwendet (Standalone möglich). Nutzung häufig in BigData / DWH / Analytics Architekturen.	Standalone. Nutzung häufig zur Auswertung Real/Right Time Transaktionen.

Serverknoten	Master und Slaves (Regions).	Alle Knoten sind gleichberechtigt. Kein Masterknoten.
Cluster, Replikation	Replikation zu einem gespiegelten Cluster möglich. Replikation über Rechenzentrumsgrenzen nicht empfohlen.	Knoten sind als Ring angeordnet. Daten werden repliziert, auch über Rechenzentrumsgrenzen hinweg möglich.
Sortierung RowKey	Automatisch, nicht änderbar. Kurze Scans nach RowKeys möglich und sinnvoll.	Konfigurierbar, ob Sortierung nach RowKey erwünscht ist.
Coprocessors	Erweiterung von HBase vergleichbar mit Stored Procedures, Trigger	Nicht unterstützt
CAP-Theorem	CP	CP oder AP möglich
Sekundäre Indexe	Nein (nur über Phoenix)	Ja

HBase Datenmodellierung
Auf den ersten Blick ähnelt das Datenmodell von Wide-column Stores den relationalen Tabellen, da ebenfalls eine Strukturierung nach Tabellen, Spalten und Zeilen erfolgt. Jedoch täuscht dieser erste Eindruck erheblich. Bei der Modellierung ist es notwendig, sich vom relationalen Modell zu lösen und die Wide-column Stores-Spezifika zu verstehen und zu nutzen.

HBase kann als eine große Hash-Tabelle verstanden werden.



Tabellen werden organisiert in

- Row Keys
  Eindeutiger Schlüssel. Indexierung und Sortierung der Daten.
- Column Families
  Column Families gruppieren die Daten. Daten auf die häufig und gemeinsam zugegriffen werden, sollten innerhalb einer Column Family organisiert werden. Column Families bestehen aus beliebig vielen Spalten, die dynamisch zur Laufzeit erzeugt werden können.

- Spalten
  Key Value-Paare aus einem Namen und einem Wert. Spalten werden nicht vorab definiert, sondern zur Laufzeit erstellt.
- Versionen
  Daten pro Spalte werden automatisch versioniert. Wird z.B. ein bereits vorhandener Row Key + Column Family Identifier + Spalten Key eingefügt, so werden die Daten versioniert (Verhalten kann auch ausgeschaltet werden).
  Anmerkung: Bei Cassandra erfolgt keine Versionierung, sondern ein Update.

Da es keine Indexe gibt, ist der Row Key entscheidend bei der Modellierung. Über den Row Key können gezielte Suchanfragen erfolgen. Ein Row Key

- Ist häufig zusammengesetzt (z.B. aus einer ID und einer Timestamp)
- Wird häufig gehasht, um möglichst gleich lange Schlüssel zu haben (Kollisionsgefahr!)

Der Zugriff auf eine Zelle erfolgt durch

rowkey => {column family => {column qualifier => {version}}}

Für die Modellierung ist die Kenntnis der Zugriffspfade notwendig, um das Datenmodell darauf ausrichten zu können. Denormalisierung, Datenduplizierung und Row Key Design sind die wichtigsten Techniken.

Als Operationen stehen zur Verfügung:

- Put
  Einfügen von Daten. Ist der Row Key bereits vorhanden, so werden die Daten zum bestehenden Row Key aktualisiert
- Get
  Selektion von Daten
- Delete
  Löschen von Daten
- Scan
  Lesen mehrerer Zeilen (profitiert von Sortierung) sowie die Angabe eines Filters möglich
- Increment
  Erhöhung eines Zeilenzählers zur Gewährleistung einer atomaren Transaktion

HBase genügt keinen ACID-Transaktionen. Der Zugriff auf eine Zeile ist jedoch atomar. HBase unterstützt keine Joins. Diese müssen manuell programmiert werden durch

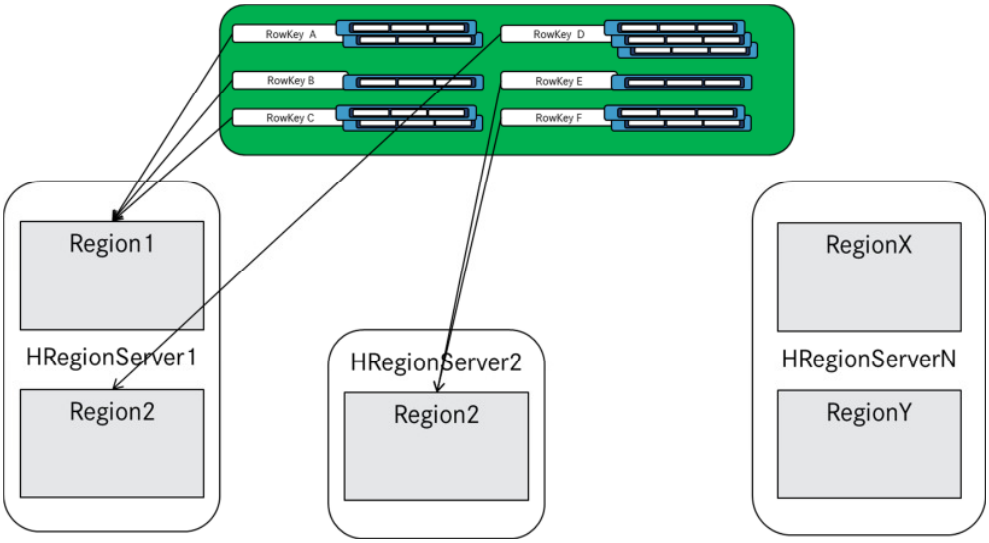
- Denormalisierung
- Zusammenfügen der Daten innerhalb des Programmcodes

HBase Architektur

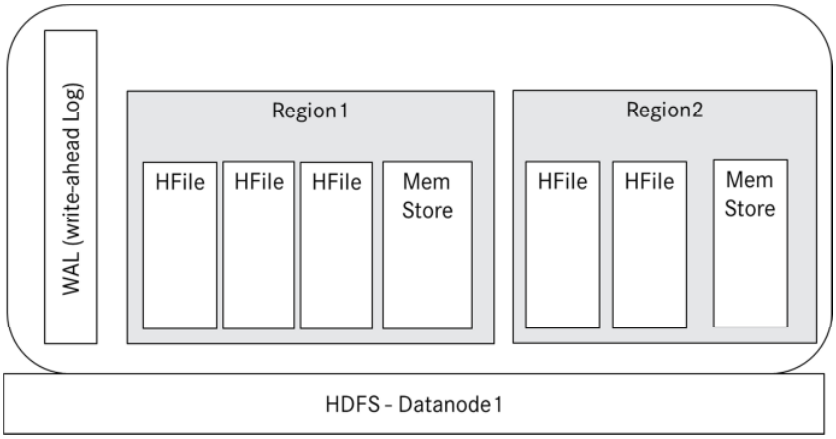
HBase setzt auf Hadoop auf und nutzt HDFS zur Speicherung von HBase-Files. Komponenten von HBase:

- HBase Master
  Der Master dient zur übergeordneten Verwaltung der DB und koordiniert die Region Server als Slaves.

- HBase Region Server
  Tabellen werden ab einer bestimmten (konfigurierbaren) Größe auf mehrere Regionen verteilt (partitioniert). Ein Regionserver kann mehrere Regionen enthalten.
- Zookeeper
  Verteilter, hoch verfügbarer Koordinator-Service. Stellt Client-Verbindungen die Meta-Tabelle zur Verfügung. Die Tabelle enthält eine Zuordnung von RowKeys zu Gegionen bzw RegionServer.
- HDFS
  Verteiltes, fehlertolerantes Dateisystem zur Speicherung der Daten



Ein Regionserver besteht aus ein oder mehreren Regionen. Eine Region hat einen MemStore zur Speicherung neuer Daten. Neue Daten werden zuerst in ein WAL (write-ahead Log) auf Platte geschrieben. Die eigentlichen Datendateien sind die sog. HFiles, in die die im MemStore gepufferten Daten gespeichert werden. Durch sog. Compactions werden HFiles regelmäßig zusammengeführt in eine neue Datei.



HBase Use Cases
HBase dagegen kann Daten in Millisekunden ausliefern und besitzt eine sehr niedrige Latenz bei wahlfreiem Zugriff. Daher kommt HBase vor allem dann zum Einsatz, wenn ein performanter, interaktiver, hoch skalierbarer Zugriff auf Daten notwendig ist. HBase wird hier häufig als Speicher von MapReduce/Batch-Prozessen oder als Speicher von Streaming verwendet. Die Lambda-Architektur von Nathan Marz sieht HBase dementsprechend in der View-Schicht vor.

NoSQL bedeutet kein SQL. Der Zugriff auf HBase folgt in der Regel mittels Java auf das zur Verfügung stehende API. Im BI/DWH Bereich sind solche Zugriffe eher unüblich. Daher besteht alternativ der Zugriff mittels SQL über

- Hive (langsam)
- Apache Phoenix
- CQL für Cassandra

Gerade der Zugriff über Phoenix bietet sich an, um HBase in bestehende Tools einbetten zu können.

Datenmodellierung ist auch im Hadoop bzw NoSQL-Umfeld wichtig, um

- Daten zu verstehen
- Performanz zu garantieren
- Entwicklung zu beschleunigen
- Qualität der Software zu verbessern
- Wartungskosten zu reduzieren
- Gemeinsames Verständnis zu fördern
- Schema-on-read: Modellversionen auch nach Jahren noch verstehen

“Data modeling is the process of learning about the data, and regardless of technology, this process must be performed for a successful application.”
Steve Hoberman: Data Modeling for Mongo DB, Technics Publications 2014

Kontaktadresse:
Andreas Buckenhofer
Daimler TSS GmbH
Business Unit Analytics
Wilhelm-Runge-Straße 11
89081 Ulm, Germany
Telefon: +49–(0)731/505–6345
E-Mail Andreas.Buckenhofer@daimler.com
Internet: http://www.daimler-tss.com