Handout - Zeitreihendatenbanken

von Kai Aderneuer, Sebastian Dornack, Fabian Wünderich

Was ist eine Zeitreihendatenbank

Spezielle Datenbank zur Verarbeitung von Zeitreihendaten, bei der der Zeitstempel als Primärschlüssel genutzt wird. Es ermöglicht die Speicherung skalarer Datentypen.

Zeitreihendatenbanken werden genutzt zur Darstellung von Wetterdaten, Prozessorauslastungen, Aktienkurse, Vitaldaten und generell Daten, bei denen die zeitliche Entwicklung relevant ist.

Funktionsweise der Datenbank

Die Speicherung der skalarer Daten wird zusammen mit einem Zeitstempel vorgenommen, dabei werden die Daten spaltenweise abgelegt.

Daten werden unveränderlich (immutable) auf den Datenträger in den Write Ahead Log geschrieben. Dabei wird mittels Write-optimierter Storage Engine ein schnellerer Zugriff auf neue Daten ermöglicht, dies wird durch Vorhalten im Cache und Sharding von Daten auf dem Datenträger ermöglicht.

Die Strukturierung der zu speichernden Daten erfolgt durch benutzerdefinierte Ordnungskriterien:
Quelle der Daten (bspw. Wetterstation)
Art der Daten (z.B. Windgeschwindigkeit, Windrichtung)

Vergleich zu relationalen Datenbank

Im Prinzip kann jede relationale Datenbank auch Zeitreihen speichern. Die Spezialisierung der Zeitreihendatenbank wurde zur Optimierung der Performance bei Speicherung und Abruf entwickelt. Jedoch bietet es keinen vollständigen Ersatz anderer Datenbanksysteme!

Eine gegenseitige Komplementierung ist das Optimum.

- Messwerte in Zeitreihendatenbank
- Metadaten und Strukturinformationen in relationaler Datenbank

- Mischung von ACID- und BASE-Prinzipien
  - Beispiel: das große Bild ist bei Zeitreihendatenbanken wichtiger als der einzelne Datenpunkt (keine Konsistenz)
  - Aber wenn Daten geschrieben werden sollen, dann sofort durch alle Caches direkt auf den Datenträger

Beispiel einer Zeitreihendatenbank - InfluxDB

Influx ist ein Datenbanksystem mit Webinterface als Hauptinterface. Die Daten können via REST-Requests oder CSV-Dateien eingetragen werden, per REST-API wird ebenfalls die Abfrage von Daten mittels eigener Domain Query Language (Flux Script) umgesetzt.
Influx ermöglicht es, die gespeicherten Daten "on-the-fly" mit dem Modul Data Explorer aufzubereiten. Um Erfassung von Daten und das Batching zu erleichtern, wird das Partnerprojekt Telegraf bereitgestellt. Eine Weiterverarbeitung von Daten kann innerhalb der Influx oder in externen Anwendungen passieren. In InfluxDB selbst aggregierte Daten können zur weiteren Verarbeitung und Auswertung in Buckets gespeichert werden, ähnlich wie erfasste Messdaten.
Zudem bietet es die Möglichkeit, die Daten mit dynamischen Dashboards zu ohne weitere Anwendungen zu visualisieren. Außerdem bietet es Unterstützung für wiederkehrende Tasks. Für die InfluxDB gibt es diverse Plugins, die das Sammeln von Daten aus verschiedenen Quellen ermöglichen.

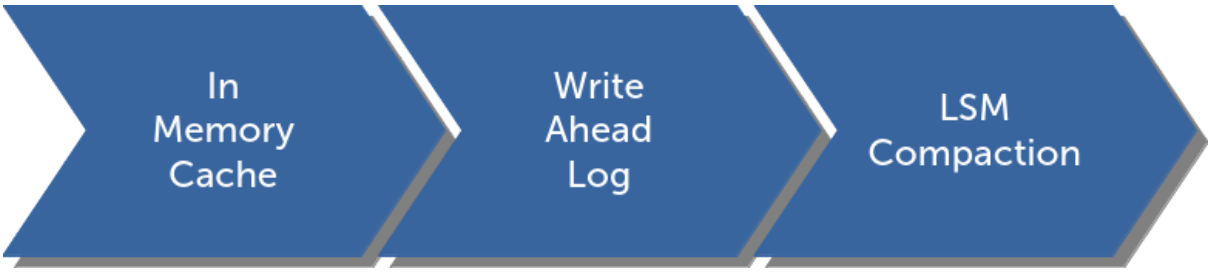
Speicher- und Zugriffsoptimierung durch Sharding nach relativem Alter der Datenpunkte (verschiedene Level des LSM tree je nach Alter). Die Deduplizierung beim Speichern wird realisiert, indem nur geänderte Werte geschrieben werden. Die gesammelten Daten werden in einem ersten Schritt in den sog. Write-Ahead-Log (WAL) geschrieben, sobald der RAM-Cache voll ist. Wenn eine Menge an WAL-Dateien überschritten ist, werden die Daten zusammengefasst und in den LSM tree eingefügt.

Beispiel der Anwendungsfälle:

1. Anomalieerkennung
   - auf Basis vergangener Daten einen "Grundzustand" definieren
   - durch die große Menge an Daten lassen sich Abweichungen von erwarteten Werten feststellen

2. Prognose
   - auf Basis vergangener Entwicklungen auf zukünftige Entwicklungen schließen
   - durch große Datenmengen und Aggregation vieler verschiedener Metriken lassen sich Zusammenhänge ausfindig machen, die genauere Prognosen erlauben

Log Structured Merge Tree



In Memory Cache

Daten werden im RAM in einer Memtable (Baumstruktur) vorgehalten, dies ermöglicht einen schnellen Lookup der neuesten Daten. Die Daten werden in einem In-Memory-Cache vorgehalten, bis eine bestimmte Größe überschritten wird, dann werden die Daten auf den Datenträger geschrieben. Ein Datenverlust ist möglich, wenn vor dem Speichern das System crasht oder neu startet.

Write Ahead Log

Dies ist eine angepasste Datenstruktur (Memtable -> SSTable) für optimierte sequenzielle Schreibgeschwindigkeit. Beim Schreiben wird der fsync()-Syscall verwendet. Dies ist zwar langsam, aber dafür sicher. Die Inhalte von WAL-Dateien werden nur angehängt, nie verändert. Sobald ein Update der Daten durchgeführt wird, wird ein neuer Wert mit demselben Key angehängt. Bei dem Fall, dass Daten gelöscht werden sollen, wird für den Key ein Tombstone (engl.: Grabstein) erstellt, dies ist vergleichbar mit einem Soft delete.

Ein Nachteil des WAL ist, dass das Suchen von spezifischen Werten aufwändiger ist, da das System weiß nicht, in welcher der WAL Dateien der gesuchte Wert steht. Das Durchsuchen und Lesen aller Dateien samt Parsen aller Updates und Soft deletes erfolgt sequenziell.

LSM Compaction

Das Compaction dient als “Aufräumschritt”. Es ist relativ langsam, da das Tree-Balancing durchgeführt werden muss. Die Updates und Soft deletes werden verarbeitet, die Dateien zusammengefasst und logisch gruppiert. Die Einordnung der gesammelten WAL Dateien wird in verschiedenen Leveln des LSM Trees durchgeführt. Je älter die Daten sind, desto mehr können diese in einem Level zusammengefasst werden. Die verschiedenen Level können geshared werden, um die Zugriffszeiten zu optimieren.

Bloom Filter

Ein Bloom Filter ist eine probabilistische Datenstruktur, die verwendet wird, um Zugriffe im LSM tree zu optimieren. Sie gibt dem System eine Aussage darüber, ob sich ein gesuchter Wert garantiert nicht oder möglicherweise in einem Level des LSM tree befindet.