Hochschule für Technik, Wirtschaft und Kultur Leipzig

Fakultät Informatik, Mathematik und Naturwissenschaften

Abstract

Oberseminar "Datenbanksysteme - Aktuelle Trends"

Hauptspeicherdatenbanken

Eingereicht von: Alexander Müller

Matrikel: 14IN-M
Matrikelnummer: 64704

Dozent: Prof. Kudraß

Leipzig, 24.07.2015

Inhaltsverzeichnis

1 Motivation

Die Anforderungen an ein modernes Datenbank-Management-System sind groß und wachsen immer weiter. In heutigen Unternehmen existieren riesige Datenbestände in verschiedensten Quellen, welche kombiniert und effizient verarbeitet werden müssen. Die Daten sollten dabei möglichst in Echtzeit analysiert werden können, um Entscheidungsträgern im Unternehmen Informationen und Erkenntnisse zu liefern, was im Moment gerade passiert. Das ermöglicht es in unmittelbar auf Veränderungen zu reagieren.

Eine solche interaktive Entscheidungsfindung setzt voraus, dass auch komplexe Analysen und Berechnungen nicht wie bisher Minuten oder sogar Stunden dauern, sondern nur noch wenige Sekunden. Um diese Anforderungen zu erfüllen, muss die Zugriffsgeschwindigkeit auf die Daten als großer Leistungsfaktor verbessert werden. Diese war bisher bei herkömmlichen Datenbankmanagementsystemen durch die Verwendung von Festplattenlaufwerken begrenzt.

1.1 In-Memory Technologie

Wenn man die Speicherhierarchie betrachtet, bietet der Hauptspeicher wesentlich höhere Zugriffsgeschwindigkeiten als Festplatten und ist der schnellste Speichertyp, der eine große Datenmenge aufnehmen kann. Die Idee sämtliche Daten einer Datenbank in den Hauptspeicher zu verlegen ist nicht neu, doch durch die stetig sinkenden Kosten für Hauptspeicher ist es mittlerweile zu einer realistischen Option für Unternehmen geworden.

Somit können riesige Datenmengen im Hauptspeicher verarbeitet werden und Analyse- und Transaktionsergebnisse können ebenfalls direkt bereitgestellt werden. Auch der Datenzugriff wird dadurch enorm beschleunigt und die Datenbewegung gleichzeitig minimiert.

Durch die Verwendung vom Arbeitsspeicher als primären Speicherort entstehen jedoch neue Herausforderungen, die ein DBMS lösen muss. Der Datenzugriff der Datenbankoperationen war bisher auf das sequentielle Lesen und Schreiben einer Festplatte optimiert und muss jetzt an die Vorteile des schnellen, wahlfreien Zugriffs angepasst werden. Auch die optimale Ausnutzung und Verwaltung des Speicherplatzes ist ein wichtiges Kriterium, denn dieser kann zum neuen Flaschenhals des Systems werden. Ein besonderes Augenmerk muss auch auf die Logging und Recovery-Mechanismen gelegt werden, da es sich bei Hauptspeicher um ein flüchtiges Speichermedium handelt und beispielsweise bei einem Stromausfall seinen Inhalt verliert.

1.2 Verarbeitung von Ereignissen

Die Einsatzmöglichkeiten von In-Memory-Datenbanken sind überall dort möglich, wo große Datenmengen zeitkritisch verarbeitet werden müssen. In Unternehmen betrifft das zunehmend die Verarbeitung von Ereignisdaten, um möglichst sofort auf Ereignisse reagieren zu können. Dabei ist der Datensatz eines einzelnen Ereignisses relativ klein, jedoch die Anzahl der Ereignisse und somit auch der Datensätze sehr hoch.

Ein weiterer interessanter Einsatzort ist die Analyse von Spiel-Ereignissen in online Spielen. Spieler neigen dazu in kritischen Spielsituationen Geld für virtuelle Güter auszugeben, um diese zu überstehen. Durch die in Echtzeit verknüpften Nutzerdaten und aktuellen Spielereignisse können gezielte Werbe- und Rabattaktionen auf diese Güter angesetzt werden, welche mit realem Geld bezahlt werden können.

Weiterhin ist die Auswertung von Sensordaten eines Formel-1-Rennwagen während eines Rennens praktizierbar. Die bis zu 600 verbauten Sensoren erzeugen hunderte Ereignisse pro Sekunde, die während des Rennens erfasst, verarbeitet und analysiert werden um Fahrzeugparameter sofort zu optimieren. Beispielsweise zur Verbesserung des Kraftstoffverbrauchs und der Höchstgeschwindigkeit aber auch zum Erkennen von Bauteilfehlern am Fahrzeug.

1.3 Echtzeit-Anforderungen von Geschäftsanwendungen

Zwei typische Geschäftsanwendungen bei denen die Performance durch die Verwendung von In-Memory-Technologie erheblich verbessert werden konnte, sind die automatisierte Erstellung von Mahnungen und der Available-to-Promise-Check.

Bei einem Mahnlauf in einem SAP-System werden Geschäftspartner mit überfälligen Forderungen automatisiert gemahnt. Ein herkömmlicher Mahnvorgang dauerte bisher mindestens 20 Minuten. Durch die Verwendung von In-Memory-Datenbanken werden hingegen nur noch wenige Sekunden benötigt. SAP selbst schreibt dazu:

„Viele Kundenanwendungen, die gegenwärtig mehr als zwei oder drei Stunden Zeit benötigt haben, können nun in weniger als zwei bis drei Sekunden ausgeführt werden.“ [1]

Auch die Durchführung eines Available-to-Promise (ATP) Check verursacht einen hohen Berechnungsaufwand auf dem gesamten Datenbestand und bringt plattenbasierte Datenbanksysteme an ihre Grenzen. Bei dem Check wird bestimmt, ob eine Menge eines Materials oder Produkts zu einem Bedarfstermin zur Verfügung steht oder ob es gegebenenfalls zu einem späteren Zeitpunkt oder in einer geringen Menge bereitgestellt werden kann. Mit In-Memory Technologie kann somit eine Anfrage auf den aktuellsten Daten und ohne eine vorherige Aggregation der Werte durchgeführt werden.

1.4 OLTP- und OLAP-Anwendungen

Online Transaction Processing (OLTP) und Online Analytical Processing (OLAP) Anwendungen richten jeweils einen sehr unterschiedlichen Workload an ein Datenbanksystem. Der OLTP-Workload ist schreibintensiv und besteht überwiegend aus kurzen Lese- und Schreibtransaktionen. Im Gegensatz dazu steht der OLAP-Workload mit hauptsächlich lesenden Anfragen, mit einer großen Anzahl von beteiligten Tupeln und aufwendigen Aggregationen. Aus diesem Grund war es bisher üblich für unterschiedliche Anwendungen getrennte Systeme eizusetzten, welche auf den jeweiligen Workload spezifisch angepasst wurden. Das hatte jedoch den Nachteil, dass die Daten redundant gespeichert wurden und in einem der Systeme deshalb nicht ganz aktuell waren

Durch die den Einsatz von In-Memory-Datenbanksystemen verringert sich die Antwortzeit auch bei komplexen Analyseabfragen erheblich, sodass sich die verschiedenen Workloads nicht mehr gegenseitig behindern. Damit ist das Betreiben von zwei getrennten Systemen überflüssig und alle Anfragen können auf den aktuellen Daten durchgeführt werden. Ein aufwendiger ETL-Prozess ist damit ebenfalls nicht mehr notwendig.

2 DBMS-Architektur

Die Architektur eines In-Memory-Datenbankmanagementsystem soll am Beispiel von SanssouciDB erläutert werden. Die Konzepte und Prototypen auf denen SanssouciDB beruht, sind hauptsächlich am Hasso-Plattner-Institut in Potsdam entwickelt worden. Bei SanssouciDB handelt es sich um ein SQL-Datenbanksystem, welches analytische und transaktionale Datenverarbeitung auf gleicher Datenbasis durchführt und sich aus typischen Datenbankkomponenten, wie Query Builder, Metadaten, Transaktions-Manager, usw. zusammensetzt.

2.1 Datenspeicherung im Hauptspeicher

SanssouciDB hält sämtliche Daten permanent im Hauptspeicher bereit und kann trotzdem nicht auf den Einsatz von nichtflüchtigen Speichermedien verzichten. Festplatten oder auch Flash-Laufwerke werden verwendet, um die Datenbank beim Start in den Hauptspeicher zu laden und alle Änderungen dauerhaft festzuhalten.

Die Daten werden außerdem spaltenweise statt zeilenweise im Datenbanksystem gespeichert. Das hat den Vorteil, dass gleiche Einträge in den Spalten nur einmal abgespeichert werden und damit Speicherplatz gespart werden kann. Auch Aggregationen, welche nur bestimmte Spalten betreffen, können somit effizient umgesetzt werden.

SanssouciDB unterscheidet bei der Datenspeicherung auch aktive und passive Daten. Bei aktiven Daten handelt es sich um Daten von Geschäftsprozessen, die noch nicht abgeschlossen sind. Diese werden zur schnellen Verarbeitung im Hauptspeicher abgelegt. Passive Daten dagegen sind Daten von Geschäftsprozessen, die vollständig abgeschlossen sind und nicht mehr geändert werden. Diese werden nicht im Hauptspeicher abgelegt, da sie

weniger häufig benötigt werden. Dieses Vorgehen dient zur Verringerung der Datenmenge im Hauptspeicher.

2.2 Drei-Schichten Architektur

SanssouciDB kann in drei logische Schichten unterteilt werden. Den „Distribution Layer“, der die Kommunikation mit Anwendungen, die Transaktionslogik, die Erstellung von Query Execution Plans und die Metadatenspeicherung übernimmt. Der Hauptspeicher als Mittelschicht, speichert die aktiven Daten, den Differential Buffers und die Indexe der Datenbank. Die unterste Schicht stellt der nichtflüchtige Speicher dar, welcher die Speicherung der passiven Daten, der Protokolle und Snapshots übernimmt.

3 Basisoperationen

Die Datenbankoperationen auf dem Hauptspeicher sind ohne Optimierung auf Festplattenzugriff programmiert, was die Algorithmen wesentlich einfacher gestaltet.

3.1 Wörterbuch-Kodierung

Ein wichtiger Aspekt bei der Umsetzung der Operationen ist die Wörterbuch-Kodierung. Sie wird verwendet, um große Dateneinträge wie beispielsweise Texte durch kurze Integer-Werte darzustellen. Wie der Name vermuten lässt, wird bei der Kodierung ein Wörterbuch angelegt in dem jedem Wert in der Spalte eine eindeutige Zahl zugeordnet wird. Identische Werte werden dabei nicht erneut in das Wörterbuch aufgenommen, sondern durch dieselbe Zahl repräsentiert. Bei vielen identischen Werten kann somit eine hohe Kompressionsrate erzielt werden. Außerdem sind die Wörterbücher in der Regel sortiert, was beispielsweise einen Lookup erheblich beschleunigt. Die Wörterbuch-Kodierung ist ebenfalls die Grundlage für weitere Kompressionstechniken, wie die Prefix- und die Lauflängenkodierung.

Neben dem Wörterbuch jeder Spalte existieren sogenannte Attribut Vektoren, welche die Wörterbucheinträge den jeweiligen Tupeln zuordnet und die sequentiell im Speicher abgelegt sind. Wenn ein Wert in der Datenbank gelöscht wird, müssen also alle nachfolgenden Tupel im Attribut Vektor verschoben werden, um den sequentiellen Speicherbereich zu erhalten.

Durch ein Insert oder Update in der Datenbank muss in der Regel ein neuer Wörterbucheintrag angelegt werden. Das führt zu erheblichen Mehraufwand durch die eventuell notwendige Umsortierung des Wörterbuches und die daraus folgende Reorganisation des Attribut-Vektors.

Die Lösung, um mögliche Leistungseinbußen einzudämmen ist die Verwendung eines „Differential Buffer“. Neue Einträge werden dabei nicht direkt in der Hauptpartition, sondern in einem als B+ Baum organisiertem Wörterbuch abgelegt, in dem die Einträge in der Reihenfolge ihres Einfügens gespeichert werden. Das Einfügen ist damit wesentlich kostengünstiger und die neuen Einträge werden nur periodisch über einen Merge-Prozess in die Hauptpartition eingefügt. Um dann den aktuellen Zustand der Datenbank zu lesen, ist es notwendig die Hauptpartition und den Differential Buffer gleichzeitig abzufragen und beide Teilergebnisse zu kombinieren.

3.2 Insert-Only Datenbank

Bei der Verwendung einer Insert-Only Strategie für eine Datenbank werden neue Datensätze grundsätzlich nur hinzugefügt und Aktualisierungs- und Lösch-Operationen nicht auf bestehenden physischen Tupeln durchgeführt. Solche modifizierenden Operationen werden in ein technisches Einfügen umgesetzt, wobei ein Zeitstempel mitgeschrieben wird. Veraltete Daten bleiben dadurch in der Datenbank und werden lediglich für ungültig erklärt. Das hat den Vorteil das Unternehmen die Veränderungen der Daten über die Zeit nachvollziehen und rekonstruieren können.

Zur Realisierung der Insert-Only Datenbank gibt es mehrere Implementierungsmöglichkeiten. Eine Möglichkeit ist Speicherung des Einfüge-Datums als „gültig ab“-Wert für jedes Tupel. Bei diese Variante ist das Einfügen sehr einfach, jedoch der Lesevorgang unter Umständen sehr aufwändig. Wenn das neuste Tupel gesucht wird, müssen alle anderen Tupel des Eintrags gelesen werden, um überprüfen zu können welches das Aktuellste ist.

Eine zweite Möglichkeit ist die Speicherung eines Zeitintervalls, das heißt für jedes Tupel einen „gültig ab“- und einen „gültig bis“-Zeitpunkt zu erstellen. Der „gültig ab“-Wert ist wie bei der ersten Variante der Zeitpunkt des Einfügens. Das weitere „gültig bis“-Attribut wird gesetzt, wenn das Tupel durch ein neueres ersetzt wird. Der Lesevorgang des aktuellsten Tupels wird dadurch vereinfacht, da nur nach dem entsprechendem Schlüssel und leerem „gültig bis“-Attribut gesucht werden muss. Das Einfügen ist dagegen etwas komplexer, weil das „gültig bis“-Attribut erst gesetzt werden muss.

4 Logging und Recovery

Die Dauerhaftigkeit einer Transaktion ist eine grundsätzliche Eigenschaft, die auch bei In-Memory Datenbanken gewährleistet werden muss. Dazu ist es notwendig die Daten zusätzlich auch auf einem persistenten Speichermedium abzulegen.

4.1 Logging

Zur Wiederherstellung von Daten im Fehlerfall z. B. bei Hardware-Ausfällen oder einem Strom Ausfall müssen sämtliche Änderungen in Log-Dateien auf persistenten Speicher abgelegt werden. Bei jeder durchgeführten Transaktion wird deshalb synchron ein Protokolleintrag geschrieben, um sicherzustellen, dass alle Transaktionen dauerhaft sind. Außerdem werden die geäderten Seiten ebenfalls in den persistenten Speicher geschrieben. Dies geschieht jedoch asynchron in regelmäßigen Abständen als sogenannte „Snapshots“.

Die Log-Infrastruktur von SanssociDB setzt sich aus den folgenden drei Teilen zusammen: den Wörterbuch-Logs, den Wert-Logs und den Snapshots des Hauptspeichers. Wörterbuch- und Wert-Logs erfassen die durch eine Transaktion geänderten Daten. Snapshots hingegen sind eine komplette Kopie des Hauptspeichers in binärer Form. Sie sollen einen möglichen Wiederherstellungsprozesses beschleunigen, da lediglich die Log-Einträge nach dem letzten Snapshot eingelesen werden müssen.

4.2 Recovery

Desaster Recovery ist ein wichtiger Aspekt bei einem produktiven Einsatz eines Datenbanksystems. Denn es ist die Fähigkeit eines Systems sich nach einem Systemausfall selbst wiederherzustellen.

So werden nach einem Neustart der Datenbank, die Datenbankseiten über einen Snapshot aus dem persistenten Speicher wieder in den Hauptspeicher geladen. Alle Änderungen die noch nicht im Snapshot erfasst wurden, werden anschließend über die Datenbankprotokolle wiederhergestellt, welche in umgekehrter Reihenfolge eingelesen werden. So wird gewährleistet, dass die Datenbank mit genau demselben Status wie vor dem Systemausfall verfügbar ist.

Um den Recovery-Prozess zu beschleunigen werden bei SanssociDB zusätzlich verschiedene Metadaten gespeichert, wie zum Beispiel die Anzahl der Zeilen im Hauptspeicher, die Anzahl der benötigten Bits zur Wörterbuch-Kodierung oder die Position des letzten Snapshots.

5 Literaturverzeichnis

[1] SAP News, „SAP HANA 1.0 sorgt für Durchbruch im Datenmanagement durch In-Memory Computing Engine," SAP, 1 Dezember 2010. [Online]. Available: http://de.news-sap.com/2010/12/01/sap-hana-1-0-sorgt-fur-durchbruch-im-datenmanagement-durch-in-memory-computing-engine/.

[2] H. Plattner, Lehrbuch In-Memory Data Management : Grundlagen der In-Memory-Technologie / Hasso Plattner, Wiesbaden: Springer Gabler, 2013.

[3] B. Berg und P. Silvia, Einführung in SAP HANA, Bonn: Galileo Press, 2013.

[4] J. Krüger, „In-Memory Data Management for Enterprise Applications," 2012. [Online]. Available: http://www.iaria.org/conferences2012/filesDBKDA12/Keynote_Jens_Krueger_v3.pdf.

[5] M. Faust, „Technical Deep-Dive in a Column-Oriented In-Memory Database," 2013. [Online]. Available: http://www.sapevent.ch/landingpagesfr/Manager/uploads/306/07_SAPHANA_PlattnerInstitut_Faust_021013.pdf.