Verarbeitung von Streamingdaten – ein Erfahrungsbericht der Testphase mit Oracle NoSQL

Marcus Bender, Rainer Marekwia, Thomas Robert und Dr. Nadine Schöne,
ORACLE Deutschland B.V. & Co. KG

Streamingdaten sind Daten, die als nahezu kontinuierlicher Datenstrom vorliegen und aus einer Vielzahl von Datenquellen stammen können. In diesem Artikel stellen wir zwei Möglichkeiten vor, diese in der Oracle-NoSQL-Datenbank zur schnellen Verarbeitung zu speichern. Die verwendeten Komponenten (Kafka, Flume und Oracle-NoSQL-Datenbank) wurden dazu auf dem Oracle Big Data Cloud Service installiert. Die Autoren gehen auch kurz auf die Eigenschaften der verschiedenen Komponenten ein und verlinken den verwendeten Code.

Streamingdaten stammen meist aus einer Vielzahl von Datenquellen. Sie liegen als nahezu kontinuierlicher Datenstrom vor. In der Regel handelt es sich um viele kleine Datenpakete; Beispiele sind Tweets, Log- oder Sensordaten. Die Auswertung solcher Daten wird für viele Unternehmen immer wichtiger; eine Analyse in nahezu Real-Time erfolgt durch direkte Beobachtung des Datenstroms. Dagegen werden bei einer klassischen Batch-Analyse die Daten erst abgespeichert – entweder in einer relationalen Datenbank nach Anwendung eines Datenschemas (Schema-on-Read) oder in HDFS (gewöhnlich Schema-on-Write) – und danach zeitversetzt analysiert.

Wie geht man aber am besten vor, wenn man aktuelle Daten möglichst schnell analysieren will, zum Beispiel, wenn man einen Überblick über den letzten Fertigungslauf erhalten oder Kundereaktionen auf Twitter direkt nach Launch einer neuen Produktlinie untersuchen möchte?

Für solche Batchanalysen von aktuellen Daten bieten sich NoSQL-Datenbanken an (NoSQL = „Not Only SQL“). Beliebige Daten lassen sich schnell ablegen. Zugriffe und Abfragen sind performant und einfach, und sie sind kostengünstig skalierbar. Verschiedene NoSQL-Datenbanken sind spezialisiert auf unterschiedliche Zwecke und Daten wie Dokumente,

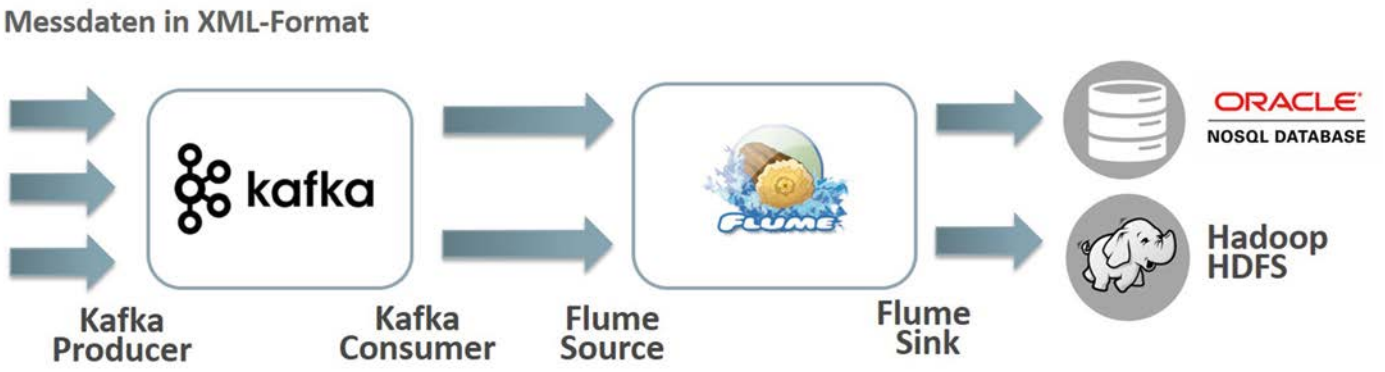

Abbildung 1: Aufgabenstellung ist die Transformation und das Ablegen von Sensordaten in einer Oracle-NoSQL-Datenbank (und gegebenenfalls auch in HDFS) unter Nutzung von Kafka und Flume

XML- oder JSON-Daten, Key-Value-Inhalte oder spaltenbasierte Daten.

Relationale Datenbanken schöpfen hingegen ihr Potenzial am besten aus, wenn bereits ein Datenschema vorliegt. Für den Zugriff auf strukturierte Daten bei gleichzeitigen Schreib- und Lesevorgängen, hoher Concurrency sowie Anforderungen an Transaktionssicherheit und Security sind relationale Datenbanken in der Regel ausgereifter als NoSQL-Datenbanken.

Das Speichern auf HDFS ist einfach umsetzbar. Nahezu alle Big-Data-Komponenten unterstützen HDFS nativ − auch mit optimierter Abspeicherung wie beispielsweise Parquet oder Avro. Abfragen sowie Batch- und Massen-Datenverarbeitung sind in der Regel sehr performant. HDFS ist aber nicht gut für die Verarbeitung vieler kleiner Dateien, sondern eher für größere Dateien geeignet. Auch eine Veränderung bereits gespeicherter Daten („Update") ist nicht vorgesehen: Das Lesen einzelner Sätze aus großen Datenbeständen ist nicht performant und Veränderungen sind in der Regel nur durch Löschen und Einfügen möglich, was zu langen Antwortzeiten führen kann. Damit ist HDFS nicht ausgelegt für eine direkte und schnelle Verarbeitung. Um die Daten im Rohzustand für spätere Batchanalysen vorzuhalten, bietet sich aber das parallele Abspeichern der Daten in HDFS an.

Aufgabenstellung

Ziel ist die Transformation und das Ablegen von Sensordaten in einer Oracle-NoSQL-Datenbank (und gegebenenfalls auch in HDFS, *siehe Abbildung 1*). Hierbei sollen die Daten aus dem Datenstrom über Flume in das NoSQL-Zielsystem übertragen werden. Der Zugriff auf die gespeicherten Daten soll möglichst wahlfrei über beliebige Schlüsselwerte möglich sein.

Datenformate

Streamingdaten können in verschiedenen Formaten vorliegen, etwa als Message-, Text-, XML- oder JSON-Format. Die in diesen Tests verwendeten Daten liegen als XML-Dateien vor. Um die Daten von ihrer Quelle zum Zielort zu bewegen, ist ein Message Broker erforderlich. Die Autoren haben sich hier für Apache Kafka entschieden.

Als Plattform wurde der Oracle Big Data Cloud Service genutzt. Dieser fertig installierte und vorkonfigurierte Cloud Service stellt auf Basis der Cloudera-Distribution bereits viele Komponenten zur Verfügung. Vorhandene Komponenten wie HDFS, Kafka und Flume sind direkt nutzbar, die Oracle-NoSQL-Datenbank lässt sich über eine einfache Installationsprozedur nachinstallieren. *Abbildung 2* zeigt eine Übersicht über die genutzten Komponenten.

Apache Kafka ist ein Broker, also ein verteiltes („distributed") Messaging-System, das für zuverlässige Message-Verwaltung, hohen Datendurchsatz, geringe Latenz und hohe Skalierbarkeit ausgelegt ist. Message-Feeds werden in Topics vorgehalten: Daten werden von Producern in Topics geschrieben und von Consumern aus Topics konsumiert. Weil Kafka ein verteiltes System ist, werden diese Topics partitioniert und auf mehrere Knoten repliziert. Die redundante Speicherung stellt Hochverfügbarkeit und Skalierbarkeit sicher. Hier kommt ein Replikationsfaktor von drei zum Einsatz, es können also bis zu zwei Knoten ausfallen, ohne dass Daten verloren gehen.

Kafka ist ein Multi-Purpose-System: Viele Producer und viele Consumer können viele Topics gleichzeitig nutzen. Damit ist Kafka sehr gut geeignet, wenn Daten von verschiedenen Applikationen verarbeitet werden sollen. Messages sind einfache Byte-Arrays, in denen Objekte in beliebigem Format gespeichert werden. Sie können mit einem Key versehen sein, sodass alle Messages mit demselben Key in dieselbe Partition geschrieben werden. Jede Topic-Partition wird als Log (geordnete Menge von Messages) behandelt. Messages werden für eine bestimmte Zeit vorgehalten, unabhängig davon, ob sie gelesen wurden oder nicht.

Kafka ist unabhängig von anderer Software, mit der Ausnahme von Apache ZooKeeper, das als übergeordnete Instanz für verteilte Ressourcen genutzt wird.

Flume ist ein Framework für das zuverlässige Sammeln, Aggregieren und Bewegen einer großen Anzahl verteilter Dateien. Ein Flume-Agent hat je mindestens eine Source, einen Channel und einen Sink. Die Source empfängt Daten und sendet diese an einen oder mehrere Channels. Ein Channel ist eine Datenqueue, die als Buffer für die Flüsse von Input und Output eines oder mehrerer Sources fungiert. Der Sink liefert die Daten eines einzelnen Channels an ein Ziel wie HDFS oder einen anderen Flume-Agenten. Daten verbleiben solange im Channel, bis ein Sink sie verarbeitet.

Es gibt fertige Schnittstellen, sodass unterschiedliche Sources und Sinks einfach angebunden werden können. Flume repliziert keine Events – wenn der Knoten mit

Abbildung 2: Übersicht der genutzten Komponenten – Apache Flume, Apache Kafka, Oracle NoSQL und Oracle Big Data Cloud Service

dem Flume-Agenten abstürzt, verliert man erst einmal den Zugriff auf die Events im Channel. Durch die Verteilung von mehreren Agenten über einen Cluster erreicht man horizontale Skalierbarkeit.

Flume enthält bereits Schnittstellen für Kafka und kann sowohl als Consumer als auch als Producer dafür fungieren. Flume liefert in dieser Kombination vorgefertigte Schnittstellen, Kafka garantiert Stabilität: Sobald die Daten in Kafka residieren, können sie nicht mehr verloren gehen. Die Nutzung eines Flume-Agenten zusammen mit Kafka wird auch als „Flafka" bezeichnet.

Der erste Schritt beim Aufbau einer Streamingdaten-Pipeline ist die Überlegung, wie die Daten am besten abgespeichert werden sollten, damit eine optimierte, schnelle Analyse erfolgen kann. Um einzelne Sätze aus großen Datenbeständen nachträglich zu verändern oder schnell zu lesen, eignen sich bestimmte NoSQL-Datenbanken wie Oracle NoSQL oder Apache HBase hervorragend. Die Autoren haben sich für die Transformation und das Ablegen von Sensordaten in einer Oracle-NoSQL-Datenbank entschieden. Oracle NoSQL ist ein verteilter Key-Value Store, mit Unterstützung von transaktionaler Verarbeitung (Create, Read, Update & Delete) und schnellem Zugriff auf einzelne Sätze über Keys *(siehe Abbildung 3)*.

Der Artikel beschränkt sich auf eine Speicherung in der NoSQL-Datenbank, einmal als Key-Value-Store und einmal über das Table-API. Daten sind in Oracle NoSQL typischerweise mit einem Replikationsfaktor von drei vorgehalten. Die Replikation auch über Datacenter hinweg liefert Hochverfügbarkeit. Es wird auch von den Replikaten gelesen (Load Balancing) und Treiber berücksichtigen Netzwerk-Latenzen. Durch die dynamische Partitionierung und Verteilung sind Hochverfügbarkeit und Skalierbarkeit garantiert. Konsistenz und Persistenz sind konfigurierbar (ACID <-> BASE).

Das Datenmodell als Key-Value-Store besteht aus einem obligatorischen Major (primären) Key und einem fakultativen Minor (sekundären) Key sowie den zugeordneten Werten (Values). Diese Ansammlung von Key-Value-Paaren wird auf Partitionen verteilt gespeichert. Gleiche Major Keys werden gemeinsam abgelegt (Daten-Kollokation). Einfache Zugriffe sind über GET, PUT und DELETE möglich. Die Identifizierung der Values erfolgt über Teile des Keys, der Value wird ausgelesen.

Oracle NoSQL bietet auch die Möglichkeit, sogenannte „Key-Paths" zu definieren, in denen der Schlüssel aus einem Pfad von Teilschlüsseln zusammengesetzt ist. In unserem Fall sollte dieser Pfad aus den drei Attributen „Ort", „Maschinen-ID" und „Material" zusammengesetzt werden, die aus den XML-Dateien zu extrahieren sind. Ein Beispiel für einen solchen Key-Path ist „/HH/4711/Blech". Es gibt keine Standard-Abfragesprache, aber eine Verwendung von SQL ist durch Nutzung von Oracle-Big-Data-SQL möglich.

Die Oracle-NoSQL-Datenbank erlaubt über das Table-API den Zugriff auf das Table-Interface des Datenstores. Tables sind Tabellen-Abstraktionen mit Zeilen und Spalten und ähneln damit Tabellen in relationalen Datenbanken. Tables haben einen wohldefinierten Primary Key. Das Table-Interface ermöglicht darüber hinaus die Nutzung von sekundären Indizes, die über separate Strukturen den Indexwerten die entsprechenden Primary Keys zuordnen und so schnellen Zugriff auf die Values ermöglichen. Unterstützte Datentypen sind zum Beispiel Integer, String, Date, Array, Map und Record.

Nutzung und Installation der Komponenten

Alle Komponenten sind Bestandteil des Oracle Big Data Cloud Service mit Ausnahme der NoSQL-Datenbank, die nach-

```
<cfg>.channels.channel-kafka.producer.max.request.size=18000000
<cfg>.channels.channel-kafka.producer.max.partition.fetch.
bytes=18000000
<cfg>.sources.src-spooldir.deserializer.maxLineLength=23000000
```

Listing 1

installiert wurde. Der Oracle Big Data Cloud Service ist in diesem Fall ein Cluster mit neun Knoten und insgesamt 306 Cores, 2,3 TB Memory und 864 TB Plattenplatz. Dahinter steckt eine Oracle Big Data Appliance, also ein Engineered System mit Oracle Linux, der Cloudera-Distribution, die Apache-Hadoop-Komponenten beinhaltet, und Oracle-Software wie Advanced Analytics For Hadoop (ORAAH).

Flume enthält fertige Schnittstellen zur Datenspeicherung im Hadoop-Ökosystem, jedoch nicht für Oracle NoSQL. Um die NoSQL-Datenbank als Sink für Flume-Events nutzen zu können, wurde ein Code-Beispiel auf GitHub *(siehe „https://github.com/gvenzl/FlumeKVStoreIntegration")* genutzt. Es erlaubt jedoch keine komplexeren Key-Strukturen, da es alle Sonderzeichen aus dem Schlüssel eliminiert. Deshalb wurde das Git-Projekt dupliziert und für die benötigten Zwecke angepasst.

Als Value sollte die komplette XML-Datei verwendet werden. Da einige der Dateien in diesem Fall bis zu 11 MB groß waren, war die Konfiguration des Kafka-Channels anpassen, um eine fehlerfreie Übertragung der Daten zu gewährleisten. Hier war insbesondere die Anpassung der folgenden Parameter auf die hier angegebenen Werte notwendig *(siehe Listing 1)*.

Ergebnisse

Kafka, Flume, Oracle NoSQL (und gegebenenfalls HDFS) bieten eine performante, skalierbare Möglichkeit, Messages in der Big-Data-Welt zu verarbeiten und später abzufragen. Kafka und Flume sind Software-Lösungen, mit denen man relativ schnell einfache Anforderungen umsetzen kann. Kafka ist eine sehr robuste, vielseitig einsetzbare Big-Data-Komponente, die über das Topic-Konzept einen Datenstrom auch in verschiedene Verarbeitungsstränge laufen lassen kann. So werden beispielsweise XML-Dateien original in HDFS-Parquet-Files archiviert und parallel können dieselben Daten in einem zweiten Strom transformiert und weiterverarbeitet werden. Dieses Konzept wird aus Sicht der Autoren viel Verbreitung finden.

Flume stellt fertige Schnittstellen mit bestimmten Funktionalitäten bereit. Aufwendigere Transformationen sind jedoch allein über Kafka und Flume (Flafka) nicht abbildbar. Dafür müssen wie in diesem Fall die Schnittstellen angepasst beziehungsweise Transformationen in Spark, Python oder Java eingebunden werden. Für die Verarbeitung einzelner Datensätze oder geringer Datenmengen aus großen Datenbeständen eignen sich NoSQL-Datenbanken wie Oracle NoSQL sehr gut. In diesen Tests hat man sich auf Oracle NoSQL konzentriert.

Es ist einfach zu installieren, skalierbar, hochverfügbar und für selektive Anfragen sehr performant. Es bietet originär über Major Keys und Minor Keys guten Zugriff auf bestimmte selektive Datenbestände. Über das Oracle-NoSQL-Table-API in Kombination mit Sekundär-Indizes werden vielfältige Selektionen von einzelnen Datensätzen bis hin zu kleineren Datenmengen unterstützt. Für komplexere Anforderungen muss man jedoch relativ viel Zeit einplanen, da die Umsetzung in der Big-Data-Welt mit Programmieraufwand, Integration in Entwicklungsumgebungen wie Maven und der Kontrolle der Kompatibilität der Softwareversionen unterschiedlicher Komponenten verbunden ist.

Mit dem angepassten Code lassen sich dann jedoch die Dateien (hier: XML-Dateien) hochperformant in der NoSQL-Datenbank ablegen. Die definierte Key-Struktur ermöglicht eine einfache Suche in dem Key-Value Store auch nach Teilschlüsseln. Allerdings müssen bei dem Key-Value-API immer die führenden Schlüsselwerte bei der Suche spezifiziert werden. In diesem Beispiel, wenn der Key aus dem Pfad „/<Ort>/<Maschinen-ID>/<Material>" besteht, kann zwar nach den Werten aller Maschinen eines Ortes gesucht werden, aber nicht nach allen Maschinen für ein bestimmtes Material.

Bei Verwendung des Oracle-NoSQL-Table-API kann über beliebige Teilschlüssel gesucht werden. Hier wurden dann neben dem unveränderten Primary Key die beiden Secondary Keys „Material" und „Maschinen-ID/Material" angelegt. Diese drei Keys ermöglichen nun die Suche über eine beliebige Schlüssel-Kombination. Alle Suchoperationen sind dabei hochperformant und erlauben einen einfachen Zugriff auf beliebige Teilmengen der gespeicherten XML-Dateien, solange sich die gesuchte Teilmenge über die Schlüsselwerte eingrenzen lässt.

Fazit

Die vorgestellten Lösungsansätze sind gut geeignet für alle Anwendungen, die eine schnelle Auswertung von Daten aus einem begrenzten Zeitraum erfordern. Kafka, Flume und Oracle NoSQL (und gegebenenfalls HDFS) bieten in Kombination eine performante, skalierbare Möglichkeit, Messages in der Big-Data-Welt zu verarbeiten und später abzufragen.

Diese Kombination liefert bereits einen guten Funktionsumfang und deckt viele Bedarfe ab. Bei komplexeren Anforderungen wie in unserem Testbeispiel wird man aber um Programmieraufwand nicht herumkommen. Dies ist nach Erfahrung der Autoren mit hohem initialen Zeitaufwand verbunden.

Marcus Bender
marcus.bender@oracle.com

Thomas Robert
thomas.robert@oracle.com

Dr. Nadine Schöne
nadine.schoene@oracle.com

Rainer Marekwia
rainer.marekwia@oracle.com