Big Data Konnektoren: Hadoop und die Oracle DB

Philipp Loer

ORDIX AG, Paderborn

Schlüsselwörter
Hadoop, Hive, OLH, OSCH

Einleitung
Der Vortrag beginnt mit einer Einführung in die Big Data Welt mit Apache Hadoop und Hive. Im Anschluss daran werden die beiden Oracle Datenbank Konnektoren vorgestellt: Der Oracle SQL Connector for HDFS (OSCH) und der Oracle Loader for Hadoop (OLH). OSCH ermöglicht den Zugriff auf in Hadoop gespeicherte Daten aus der Oracle DB heraus. OLH hingegen das Laden von Daten aus Hadoop in eine Oracle Datenbank.

Apache Hadoop
Apache Hadoop ist ein Framework, das die verteilte Verarbeitung von großen Datenmengen auf einem Cluster mit einfachen Programmiermodellen ermöglicht. Genutzt werden hierzu aus handelsüblicher Hardware bestehende Knoten. Es wurde entwickelt, um von wenigen Knoten auf tausende nahezu linear skalieren zu können. Hierbei stellt jeder Knoten sowohl Rechenleistung, als auch Speicherplatz zur Verfügung. Hadoop besteht im Kern aus zwei Komponenten: Dem Resource Manager für die Berechnungen und HDFS (Hadoop Distributes File System) für die Speicherung von großen Datenmengen.
Das Hadoop Framework nimmt dem Programmierer Arbeit ab. Dieser muss nur noch die „Lücken“ (die Map- und Reduce-Funktionen, Spark Code etc.) füllen. Hadoop organisiert die Verteilung der Berechnung im Cluster und alle Aufgaben, die sich aus der Verteilung von Anwendungen ergeben. Die Berechnung selbst muss jedoch vom Programmierer implementiert werden. Hierbei stellt sich unweigerlich die Frage, welche Vorteile Hadoop bietet, wenn doch der Programmierer den eigentlichen Programmcode immer noch selbst entwerfen muss.
Entscheidender Vorteil von Hadoop ist nicht die nahezu lineare Skalierbarkeit, sondern der vereinfachte Umgang mit großen Datenmengen. Das Hadoop Framework steuert die Daten und die Berechnungen im Cluster. Wenn man all dies selbst programmieren wollte, müsste man sich unter anderem um die folgenden Dinge kümmern:

- Koordination der Berechnungen der einzelnen Knoten
- Koordination der Datenverteilung über die Knoten im Cluster
- Erkennung von Fehlern
- Neustart fehlerhafter Berechnungen

Hadoop bietet eine praxiserprobte, robuste Lösung für all diese Probleme. Es reduziert den Aufwand der Entwicklung auf zwei Aufgaben: Erstens muss das Problem auf kleine Teile herunter gebrochen werden, damit es parallel in einem Cluster bearbeitet werden kann. Zweitens muss der Programmcode für diese Berechnung bereitgestellt werden. HDFS als Dateisystem von Hadoop bietet theoretisch

unbegrenzten Speicherplatz. Daher hat sich um Hadoop ein ganzes Ökosystem von Technologien gebildet, um den Umgang mit großen Datenmengen weiter zu vereinfachen.

Die Abbildung 1 zeigt die Einordnung der Knoten in einem Cluster. Sie unterteilen sich zunächst in Master- und Slave-Knoten. Slave-Knoten sind die „Arbeitspferde“ des Clusters. Sie speichern die Daten und nehmen die (Teil-) Berechnungen vor. Die Master-Knoten wiederum unterteilen sich nach ihren jeweiligen Aufgaben: Der Steuerung der Speicherung der Daten sowie der Steuerung der Berechnungen.

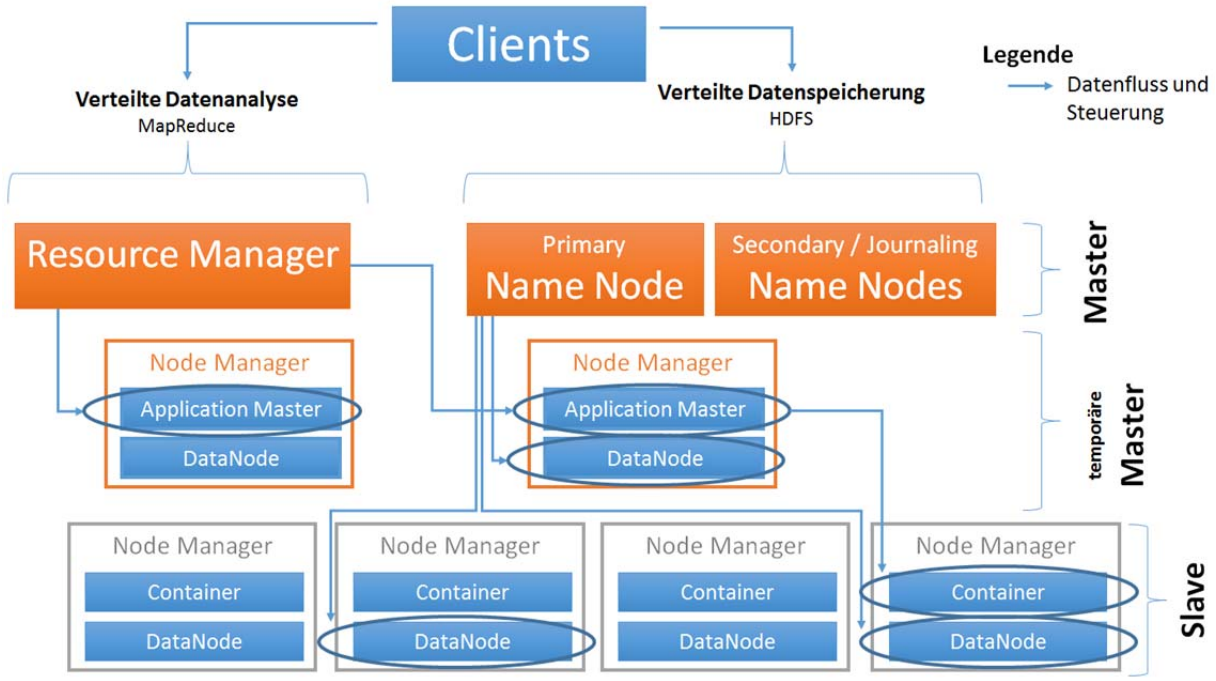

Abb. 1: Hadoop Architektur

Das Hadoop Distributed File System (HDFS)

Dateien werden in HDFS in über mehrere Data Nodes in verteilten Blöcken gespeichert. In einem Dateisystem bilden Blöcke üblicherweise die kleinste Speichereinheit. Der von einer Datei in einem Dateisystem belegte Speicherplatz ist folglich immer ein ganzzahliges Vielfaches der Blockgröße. Während in „General Purpose“-Dateisystemen die Blockgrößen üblicherweise im niedrigen Kilobytebereich liegen, ist in HDFS die Blockgröße von 128 Megabyte (MB) der aktuelle Standard. Durch diese großen Blöcke werden Verwaltungsaufwand und Suchzeit in HDFS für den einzelnen Block reduziert. Durch die Aufgabenverteilung zwischen dem Name Node und den Data Nodes wird zudem eine in verteilten Dateisystemen vorteilhafte Blockabstraktion erreicht. Eine einzelne Datei kann so größer werden als die größte Festplatte des Dateisystems und es wird eine Vereinfachung des Speicher-Subsystems erreicht, denn das Subsystem wird aufgrund seiner geringeren Komplexität weniger fehleranfällig. Darüber hinaus müssen keine Berechtigungsinformationen (Metadaten) auf Blockebene gespeichert werden. Diese können zentral im Data Node verwaltet werden. Die Blockabstraktion vereinfacht zudem die Replikation. Einfache Blockkopien werden verteilt, um

Datenverlust durch Hardwarefehler zu vermeiden. Standardmäßig wird jeder Block auf drei verschiedenen Data Nodes gespeichert.

Apache Hive

Hive wurde ursprünglich 2007 von Facebook entwickelt, um folgendes Problem zu lösen: Die enormen Datenmengen, welche bei Facebook jeden Tag anfallen, sollten in Hadoop gespeichert werden. Leider bot Hadoop zum damaligen Zeitpunkt keine Unterstützung für SQL. Hive wurde daher geschaffen, um Analysten mit guten SQL-Kenntnissen die Möglichkeit zu geben, die Daten zu analysieren ohne sich zuvor die zur Erstellung eines MapReduce Jobs (bzw. seit kurzem auch Spark Jobs) erforderlichen Fähigkeiten und Kenntnisse aneignen zu müssen. Hive ist heute ein von vielen Unternehmen eingesetztes, erfolgreiches Apache Projekt zur Verarbeitung unterschiedlichster Daten.
SQL ist nicht unbedingt die ideale Lösung für jedes Big Data Problem. So ist es beispielsweise für machine learning nicht geeignet. Für viele Analysen, Aggregationen und Abfragen ist es hingegen eine sehr gute Lösung. Der größte Vorteil von SQL ist jedoch seine Verbreitung in der IT und bei den Business Intelligence Werkzeugen. Durch die Verwendbarkeit von SQL kann Hadoop gut mit diesen Produkten interagieren. Der von Hive verwendete SQL-Dialekt heißt Hive Query Language (HQL).
Hive ist nicht für Online Transactions Processing (OLTP) Workloads bestimmt, es bietet keine Unterstützung für Transaktionen, Echtzeitabfragen oder Constraints. Das Haupteinsatzgebiet sind riesige Datenmengen, bei denen nur Daten hinzugefügt werden.

Das von den Nutzern erstellte SQL-Statement wird durch den Hive Interpreter in MapReduce Jobs umgewandelt. Die Erstellung von Joins oder Gruppierungen ist somit deutlich weniger aufwändig als die Erstellung von normalem MapReduce Programmcode. Traditionelle Datenbanken prüfen bei dem Hinzufügen von Daten, ob diese zur Tabellendefinition passen (schema on write). Passen die Daten nicht, so werden diese abgelehnt. Hive hingegen prüft die Daten nicht zum Zeitpunkt des Einfügens, sondern erst, wenn diese gelesen werden. Diese Vorgehensweise wird schema on read genannt. Durch schema on read müssen die Daten nicht vor dem Speichern überprüft werden und können somit schneller gespeichert werden. Das Hinzufügen von Datensätzen ist daher nur ein kopieren oder verschieben. Zudem bietet diese Vorgehensweise den Vorteil, Daten speichern zu können, deren Struktur zum Zeitpunkt der Speicherung noch unbekannt ist.

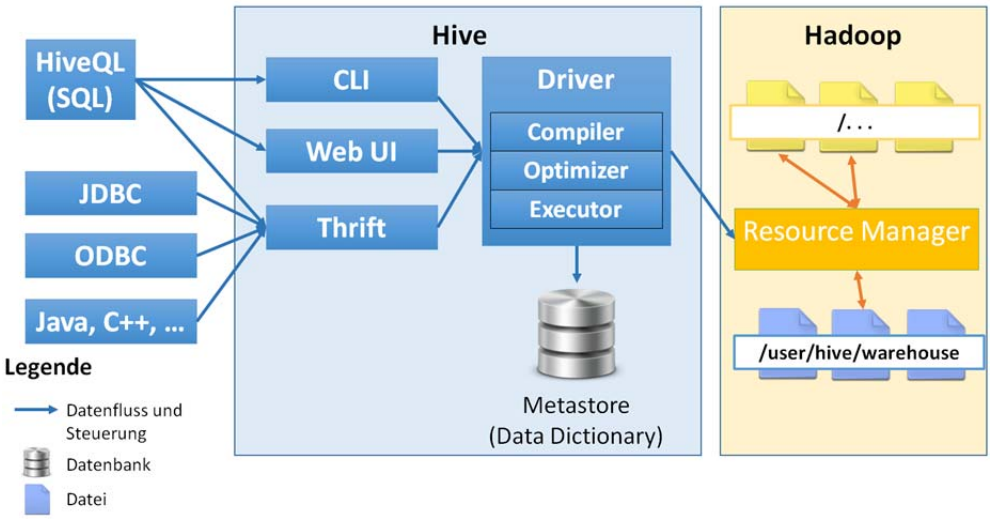

Abb. 2: Hive Architektur

Die Architektur von Hive ist in Abbildung 2 dargestellt. Hive bietet ein Command Line Interface (Beeline), eine webbasierte Benutzeroberfläche sowie ein Thrift-API. Zur Ausführung von HQL können alle drei Schnittstellen genutzt werden. Verbindungen über JDBC, ODBC oder programmiersprachenindividuelle Schnittstellen können nur über Thrift hergestellt werden.

Das HQL-Statement wird zur Ausführung von der Schnittstelle an den Hive Driver übergeben. Daraufhin erstellt der Compiler auf Anforderung des Hive Drivers einen Ausführungsplan. Hierzu wird der Metastore nach den Metadaten (s. u.) befragt. Der vom Compiler erstellte Ausführungsplan wird anschließend vom Hive Driver an die Execution Engine übermittelt. Diese erstellt daraus einen oder mehrere MapReduce Jobs, welche an den Hadoop Cluster zur Ausführung übermittelt werden. Die Ergebnisse gelangen dann über die Execution Engine und den Hive Driver an die Benutzerschnittstelle. Mit Beeline bietet Hive eine Kommandozeile für die Ausführung von HQL-Statements.

Eine Hive Datenbank entspricht dem, was in vielen relationalen Datenbanken unter einem Schema verstanden wird. Innerhalb einer Datenbank können Tabellen angelegt werden. Die Standard-datenbank liegt im Hive Warehouse-Verzeichnis im HDFS (/user/hive/warehouse/). Alle anderen Datenbanken liegen ebenfalls hier, jedoch in separaten Unterverzeichnissen. Tabellen einer Datenbank liegen in jeweils eigenen Unterverzeichnissen.

Der Hive Metastore speichert die Zuordnung in HDFS gespeicherter Dateien zu den Hive-Tabellen. Darüber hinaus enthält er alle Informationen, die für eine Serialisierung und Deserialisierung der Daten benötigt werden. Dieser Systemkatalog vergrößert somit die Flexibilität und ermöglicht einen gewissen Abstraktionsgrad. Der Nutzer muss nur den Tabellennamen wissen, nicht aber in welchen Dateien die Daten liegen, welches Speicherformat verwendet wird oder wie die interne Struktur der Dateien aufgebaut ist. Standardeinstellung für den Hive Metastore ist eine Apache Derby Datenbank auf dem Client. Wird Hive von mehreren Benutzern verwendet, empfiehlt sich die Einrichtung eines gemeinsamen Metastores. In der Regel wird hierzu MySQL oder eine andere relationale Datenbank verwendet.

Oracle SQL Connector for Hadoop (OSCH)

Der Zugriff auf außerhalb der Oracle Datenbank liegende Daten konnte auch schon bei älteren Oracle-Versionen über eine externe Tabelle erfolgen. Dazu musste lediglich angegeben werden, an welchem Ort diese Daten liegen und wie die Daten strukturiert sind. Dieser Ansatz wurde nun auf die Big Data Welt erweitert: Es ist seit der Version 12.1.0.2 möglich, statt einer bisher üblichen csv-Datei, wahlweise in HDFS liegende Text- (z. B. csv) und Oracle Data Pump-Dateien oder eine Hive-Tabelle zu verwenden. Diese externe Tabelle kann in der gewohnten Weise per SQL wie jede andere Datenbanktabelle abgefragt und deren Daten mit anderen Tabellen gejoined werden. Es ist aber weiterhin auch bei diesen sogenannten „enhanced external Tables“ keine Änderung der Daten, z. B. durch Insert- oder Update-Statements, möglich.

Oracle Loader for Hadoop (OLH)

Der Oracle Loader for Hadoop ist ein MapReduce Programm, das im Hadoop Cluster läuft, um Daten aufzubereiten. Es kann diese partitionieren, sortieren und in Oracle Datentypen konvertieren. Durch

diese Auslagerung von Datenbankaufgaben nach Hadoop kann eine verringerte Prozessorlast der Oracle Datenbank während des Ladevorgangs erreicht werden. Der Ladevorgang kann in zwei verschiedenen Modi durchgeführt werden: Online und offline. Beim Offline-Modus werden die vorbereiteten Daten in Data Pump-Dateien in HDFS für einen späteren Ladevorgang zwischengespeichert. Beim Online-Modus werden die Daten direkt in die Oracle Datenbank geschrieben. Dieser Modus besitzt wiederum zwei Optionen: JDBC und direct path.

Kontaktadresse:
ORDIX AG
Philipp Loer
Karl-Schurz Str. 19a
D-33100 Paderborn

Telefon:	+49 (0) 5251-1063-0
Fax:	+49 (0) 180-1673490
E-Mail	doag@ordix.de
Internet:	www.ordix.de