Big-Data-Technologien: Qual der Wahl?

Prof. Dr. Jens Albrecht
Technische Hochschule Nürnberg

Schlüsselworte

Big Data, Hadoop, NoSQL.

Einleitung

Der Big-Data-Hype der vergangenen Jahre hat dazu geführt, dass eine Vielzahl technologischer Lösungen neu auf den Markt kam bzw. reif für den Unternehmenseinsatz gemacht wurde. Letzteres trifft insbesondere für Open Source Produkte wie Hadoop und NoSQL-Datenbanken zu. Auch Oracle hat inzwischen eine Vielzahl an Lösungsbausteinen für Big Data im Portfolio: Angefangen von neuen Features in der Datenbank über NoSQL und Hadoop bis hin zu MySQL.

Mit der Verfügbarkeit technologischer Optionen steigt gleichzeitig die Zahl wirtschaftlich interessanter Anwendungsfälle für Big Data Technologien. Daher müssen sich inzwischen fast alle Firmen mit der Frage auseinandersetzen, ob und wie die neuen Technologien sinnvoll und gewinnbringend eingesetzt werden können. Der Vortrag soll Entscheidern und Entwicklern bei der Klärung dieser Fragen durch Vermittlung des notwendigen Überblicks helfen. Dafür wird das Spektrum verfügbarer Big-Data-Technologien aus dem Hause Oracle aufgezeigt und im Hinblick auf ausgewählte Big-Data-Szenarien kritisch bewertet. Dabei werden neben technologischen Aspekten auch Fragen der Wirtschaftlichkeit angerissen.

Besondere Anforderungen durch Big Data

Da "Größe" etwas Relatives ist, ist auch die Frage, wann Daten eigentlich "Big Data" sind, am besten über eine relative Definition zu beantworten: Unter Big Data werden Daten verstanden, die mit den bisher eingesetzten Technologien entweder nicht effizient oder nicht wirtschaftlich verwaltet und ausgewertet werden können. Sie sprengen die Grenzen klassischer Datenbanken und erfordern daher einen Technologiewechsel auf Seiten der IT. Die Grenzen müssen dabei nicht technologisch, sondern können insbesondere auch wirtschaftlich begründet sein. So ist es technisch kein Problem, hunderte Terabytes an Daten in relationalen Datenbanken zu halten; es ist aber für viele der neuen Anwendungsfälle schlichtweg zu teuer.

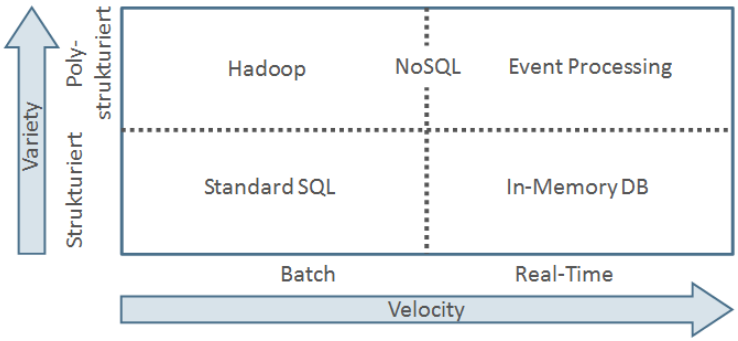

Abb. 1: Big Data Technologien im Überblick (in Anlehnung an)

Der Grund dafür ist, das Big Data oftmals einen geringen Wert in Bezug auf das Datenvolumen haben. Beispiele dafür sind Sensor-, Log- oder Social Media Daten in Terabyte-Größe, die durchaus interessante Informationen beinhalten können. Anders als beispielsweise Buchhaltungsdaten sind einzelne Einträge für sich aber praktisch wertlos. Erst durch starke Filterung und Aggregation der relevanten Informationen können Big Data erschlossen werden. Big Data Technologien stellen daher Möglichkeiten bereit, große Datenmengen effizient und wirtschaftlich auszuwerten.

Abbildung 1 zeigt eine vereinfachte Segmentierung für Big Data Technologien. Datenvielfalt und die Verarbeitungsgeschwindigkeit sind dabei allerdings nur zwei Dimensionen des Kriterienspektrums, das darüber hinaus noch Aspekte wie Skalierbarkeit, Schnittstellen für Import und Datenzugriff, Transaktionssicherheit, Komplexität, Sicherheit, Marktreife sowie letztendlich Kosten für Lizenzen, Betrieb und Wartung umfasst.

Datenvielfalt und Schema-on-Read

Unabhängig davon wie unstrukturiert die Daten sind, die in ein Big-Data-System geladen werden, für die Auswertung und Analyse ist und bleibt SQL erste Wahl. Damit semistrukturierte Daten mit SQL erschlossen werden können, muss ein Schema definiert werden. Anders als beim Data Warehousing, wo die Daten in eine relationale Datenbank geladen werden (*Schema-on-Write*), wird in Big-Data-Systemen ein Schema quasi als View über die Daten gelegt und erst beim Lesen angewendet (*Schema-on-Read*). Dieses Konzept wird in der Oracle Datenbank schon seit langem für External Tables eingesetzt. Schwach strukturierte Rohdaten können dadurch auf unterschiedliche Weisen interpretiert werden und ein ETL-Prozess im klassischen Sinn entfällt. Allerdings gehen dadurch auch viele Möglichkeiten zur Performance-Optimierung verloren.

Hadoop, YARN und Spark

Hadoop ist ein Framework von Software-Komponenten für die skalierbare Haltung und Verarbeitung von Big Data. Grundlage bildet das HDFS, ein verteiltes Dateisystem, das auf einem Shared-Nothing-Cluster aus kostengünstigen Commodity Rechnern läuft. Es zeichnet sich insbesondere durch eine hervorragende Skalierung und Fehlertoleranz aus, hat aber Schwächen bei Random Reads auf kleinen Dateien.

Ebenfalls Bestandteil des Hadoop-Kerns ist das Map-Reduce-Framework, eine verteilte Ablaufumgebung für Jobs zur Verarbeitung der Daten im HDFS. Map-Reduce ist ein Batch-orientiertes Verfahren, das nicht für interaktive Aufgaben geeignet ist. In Hadoop 2.0, das seit Ende 2013 einsatzfähig ist, wurde mit YARN eine generische Ablaufumgebung für verteilte Jobs eingeführt, die sowohl Map-Reduce als auch andere Frameworks einbinden kann. Apache Spark ist beispielsweise ein In-Memory-Framework für die latenzarme Verarbeitung von Jobs im Hadoop-Cluster, das über YARN gesteuert werden kann.

Hadoop 2.0 ist CDH 4.x Basis der Cloudera Distribution, welche für die Big Data Appliance zum Einsatz kommt. Über Konnektoren ermöglicht Oracle aber auch den Zugriff auf andere Hadoop-Installationen. Abbildung 2 zeigt die integrierte Hadooop-Lösung von Oracle.

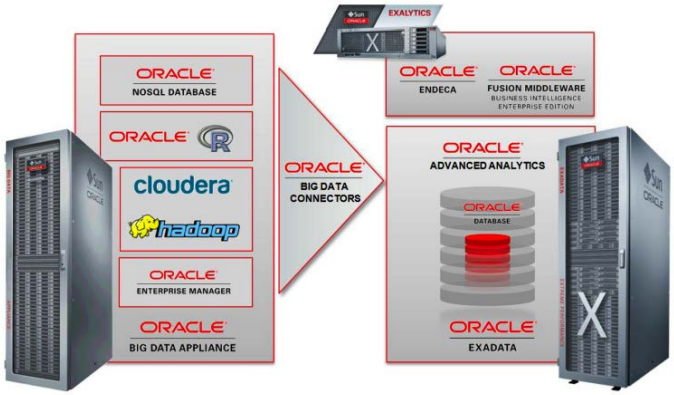

Abb. 2: Integrierte Big Data Lösung von Oracle (übernommen aus [1])

Zugriff auf Hadoop über Oracle Big Data Connectors

Die Oracle Big Data Connectors bestehen aus den folgenden vier Tools.

Oracle Loader for Hadoop: Der Oracle Loader für Hadoop ist ein Kommandozeilen-Werkzeug, das auf Map-Reduce basiert. Die Daten können entweder direkt in die Datenbank gepumpt oder als Dump-File abgelegt und dann über eine External Table eingebunden werden. Als Dump-File-Formate stehen Datapump (binär) und CSV-Text zur Verfügung. Er eignet sich insbesondere, wenn Daten im Rahmen eines ETL-Prozesses zunächst mit Map-Reduce aufbereitet werden und dann geladen werden sollen.

Oracle SQL Connector für HDFS: Der SQL-Connector für HDFS erlaubt die Einbindung von Daten im HDFS als External Table. Üblicherweise wird der SQL-Connector eingesetzt, um im Rahmen eines ELT-Prozesses CSV-Dateien aus dem HDFS in eine DB zu laden, um sie dann relational weiter zu verarbeiten.

ODI Application Adapter vor Hadoop: Der Oracle Data Integrator (ODI) erlaubt die einfache Einbindung von Hadoop über seine grafische Benutzeroberfläche. Für den Datenzugriff generiert ODI Anweisungen in HiveQL, die dann als Map-Reduce-Jobs in Hadoop ausgeführt werden. ODI steuert dabei nicht nur die Generierung, sondern auch die Ausführung der Jobs.

Oracle R Connector For Hadoop: Mit Hilfe dieses Connectors können Hadoop-Daten mit R-Funktionen verarabeitet und zwischen Oracle Datenbank und Filesystem ausgetauscht werden.

Oracle Big Data SQL

Im Sommer 2014 wurde Oracle Big Data SQL vorgestellt. Damit wird es möglich, Daten aus der Oracle DB, Hadoop und Oracle NoSQL gleichzeitig mit SQL zu verknüpfen. Die Besonderheit dabei ist, dass die Smartscan-Technologie der Exadata auf die Hadoop-Landschaft übertragen wurde. Grundidee dabei ist es, die Filterungen und Aggregationen möglichst nahe an der Speicherebene (in diesem Fall Hadoop oder NoSQL-DB) auszuführen und dann die komprimierten Ergebnisse für die Aufbereitung weiter zu reichen.

Leider ist es diese im Marktumfeld einzigartige Funktionalität nur für die Big Data Appliance verfügbar. Als relationale Datenbank ist aktuell eine Exatadata mit Version 12.1 erforderlich.

NoSQL-Datenbanken für Big Data

Es gibt über 100 verschiedene NoSQL-Datenbanken. Das Datenmodell beruht meist auf dem Key-Value-Ansatz, d.h. zu einem (Text-)Schlüssel werden frei definierbare Daten hinterlegt. NoSQL-Datenbanken zeichnen sich durch eine hohe Skalierbarkeit und kurze Reaktionszeiten aus. Daher werden sie oft für Anwendung mit einer hohen Zahl an Random Reads oder Writes eingesetzt. Dem gegenüber stehen beschränkte Abfragemöglichkeiten: Oft steht nur eine Programmierschnittstelle zur Verfügung, komplexe Abfragen sind nicht ohne weiteres möglich. Hinzu kommen deutliche Abstriche bei der Transaktionsfähigkeit. Zusammengesetzte Transaktionen und konsistentes Lesen sind nicht möglich oder erfordern wiederum Abstriche bei der Skalierbarkeit.

Oracle NoSQL-Datenbank

Die Oracle NoSQL-Datenbank basiert auf der Berkeley DB und implementiert das Key-Value-Modell. Sie ist insbesondere für eine hohe Verarbeitungsgeschwindigkeit bei Random Reads und Writes konzipiert. Dafür werden die Daten automatisch mit Hilfe der Keys auf die Hash-Partitions abgebildet, die den Knoten des Clusters automatisch zugeordnet werden (Sharding).

Eine Besonderheit ist die Zusammensetzung des Keys aus einem Major-Part und einem Minor-Part. Der Major-Part dient als Hash-Key für die Verteilung der Daten im Cluster und sorgt somit dafür, dass alle Daten mit dem gleichen Major-Key auf demselben Knoten liegen. Der Minor-Part ist eine Ergänzung, um die beispielsweise bestimmte Attribute eines Objekts abzubilden. Die Daten eines Users könnten beispielsweise über folgende Major-Minor-Keys abgebildet werden:

Major Key	Minor Key	Value
Userid: 4711	Name	"Schmidt, Peter"
Userid: 4711	LastVisit	{ url: "www.someurl.com", timestamp: "2014-10-23 19:23:54 }

Dadurch, dass alle Daten zu einem Major-Key auf einem Server liegen, kann die Oracle NoSQL-DB in beschränktem Maße ACID-Transaktionen erlauben: Sofern gewünscht werden alle Änderungen zu einem Major-Key transaktional ausgeführt. Jede Änderung aktualisiert automatisch den Timestamp eines Eintrags.

Anders als andere NoSQL-Datenbanken erlaubt Oracle NoSQL ein Tuning der Konsistenz. Mit der Konsistenzeinstellung "Absolute" wird sichergestellt, das immer der aktuellste Wert in der Datenbank zurückgegeben wird. Das geht allerdings zu Lasten der Performance, da nur vom Master-Knoten gelesen werden kann. Flexiblere Einstellungen erlauben das Lesen einer beliebigen Kopie.

Die Oracle NoSQL-Datenbank kombiniert damit viele Eigenschaften, die auch andere NoSQL-Systeme haben, mit sinnvollen Alleinstellungsmerkmalen und der Sicherheit eines großen Software-Anbieters im Hintergrund. Insbesondere, wenn die Datenbank-Landschaft durch Oracle dominiert ist, erscheint sie damit attraktiver als MongoDB & Co., weil die Integration mit Oracle Hadoop und der relationalen Oracle DB deutlich einfacher ist.

Die Oracle NoSQL Datenbank kann als Community Edition kostenfrei betrieben werden. In der Enterprise Edition verlangt Oracle Gebühren für Lizenzen und Support, die ungefähr bei 60% der Oracle DB Standard Edition liegen (siehe [4]). Das Produkt ist leistungsfähig und ausgereift.

HBase

HBase ist Bestandteil der Hadoop-Distribution und damit die Standard-NoSQL-Datenbank in einem Hadoop-System. HBase ist für wirklich große Datenmengen konzipiert, d.h. hunderte Millionen oder besser Milliarden von Einträgen. Es ist eindeutig ein Schwergewicht unter den NoSQL-Datenbanken, was sich auch in der Konfiguration und Administration niederschlägt.

HBase basiert konzeptionell auf Google Big Table und ist eine spaltenorientierte, nicht-relationale Datenbank, die ebenfalls für hohe Lese- und Schreibdurchsatz konzipiert wurde. Die Datenbank sichert Lesekonsistenz, d.h. einzelne Datensätze werden atomar geschrieben und beim Lesen wird grundsätzlich der aktuellste Wert zurückgegeben.

Jeder Eintrag in einer Tabelle wird durch einen Rowkey identifiziert. Ein Rowkey kann beliebig viele Spalten haben, die Spaltenfamilien zugeordnet sind. Während letzte bei Anlage einer Tabelle angelegt werden müssen und danach nicht mehr verändert werden dürfen, sind die Spalten oder Attribute pro Datensatz frei wählbar, d.h. die Datensätze einer Tabelle können unterschiedliche Strukturen haben. Spalten können jederzeit zur Laufzeit hinzugefügt werden. Auch in HBase werden die Datensätze automatisch mit Timestamps versehen. Im Gegensatz zu Oracle NoSQL werden Werte aber nicht überschrieben, sondern es wird eine neue Version mit einem neuen Timestamp erzeugt.

HBase und Oracle NoSQL sind sich in vielen Eigenschaften sehr ähnlich und auch die Anwendungsszenarien lassen sich nicht klar abgrenzen. Eine gute Gegenüberstellung ist in [3] zu finden. Oracle NoSQL ist die leichtgewichtigere Variante, die ohne eine Hadoop-System auskommt. Es eignet sich sowohl für Batch-Aufgaben im Rahmen von ETL-Prozessen als auch für hoch interaktive Webanwendungen. Sollen allerdings große Datenmengen in Hadoop verarbeitet und gehalten werden, so bietet sich HBase als Datensenke für aufbereitete Log-Files und Sensordaten an. Ein Zugriff mit SQL ist über Hive realisierbar.

MySQL ohne SQL

Obwohl MySQL nicht Bestandteil der offiziellen Oracle Big Data Strategie ist, kann MySQL eine sinnvolle Plattform für Big Data Anwendungen sein. Große Firmen wie Facebook halten gigantische Datenmengen in MySQL-Datenbanken. Da hier sehr häufig die Datenbank als Key-Value-Store "missbraucht" wurde, gibt es seit MySQL 5.6 die Möglichkeit, die SQL-Verarbeitung komplett zu umgehen und somit Performance zu gewinnen. Dafür wurde die Memcached-API direkt in MySQL implementiert.

MySQL stellt dadurch eine interessante Alternative dar, denn es erlaubt die Kombination von schneller NoSQL-Verarbeitung über die Memcached-API mit allen Vorteilen der ausgereiften InnoDB Speicherengine, einschließlich Transaktionssicherheit. Allerdings ist eine MySQL-Datenbank in einem Oracle-Umfeld immer noch eher als Fremdkörper zu betrachten. Für die meisten Unternehmen kommt es daher im Big Data Umfeld nur für spezielle Aufgaben in Frage.

Oracle DB als Big-Data-Engine

Neben neuen Technologien wie Hadoop und NoSQL bleibt die relationale Datenbank elementarer Bestandteil der Big-Data-Infrastruktur. Wichtigster Aspekt ist neben der Transaktionssicherheit die Nutzung von SQL für komplexe Analysen. Darüber hinaus finden auch NoSQL-Funktionalitäten wie das JSON-Format ihren Weg in die relationale Datenbank.

In-Memory-Option

Mit der Version 12.1 der Datenbank kam Mitte diesen Jahres die In-Memory-Option auf dem Markt [5], die Oracle den Anschluss an IBMs DB2 Blu und SAP HANA verschaffen soll. Ebenso wie die Konkurrenz verwendet Oracle ein spaltenorientiertes Format zur Datenhaltung, um Datenkompression einerseits und die Möglichkeiten der vektorbasierten Verarbeitung moderner CPUs optimal auszunutzen. Anders als bei der Konkurrenz wird nur im Hauptspeicher eine spaltenorientierte Variante der Daten gehalten, das physische Speicherformat bleibt auch bei In-Memory-Tabellen zeilenorientiert. Dadurch gibt es keinerlei Einschränkungen beim Sprachumfang, beim OLTP-Betrieb oder bei der Anlage von Indizes. Viele Indizes werden aber womöglich nicht mehr benötigt.

Die Speicheroptionen sind sehr feingranular konfigurierbar, um die Datenstrukturen möglichst optimal auf die Anforderungen abzustimmen. So können nicht nur ganze Tabellen auf das spaltenorientierte In-Memory-Format umgestellt werden, sondern auch einzelne Partitionen oder auch nur einige häufig benutzte Spalten einer Tabelle.

In Bezug auf diese Flexibilität ist Oracle den Mitbewerbern auf jeden Fall voraus. Preislich langt Oracle hier mit 50% Aufschlag auf den Lizenzpreis der Enterprise Edition ordentlich zu [4]. Ob dieser Preis gerechtfertigt ist, wird sich erst zeigen, wenn belastbare Performancewerte aus der Praxis verfügbar sind.

JSON in der Datenbank

Gleichzeitig mit der In-Memory-Option wurde auch die JSON-Funktionalität freigegeben. NoSQL-Datenbanken werden oft dann eingesetzt, wenn sich die Datenstrukturen schlecht auf ein relationales Datenmodell abbilden lassen oder wenn sehr oft Änderungen im laufenden Betrieb erforderlich wären. JSON ist daher gerade im Internet-Umfeld als Datenformat sehr beliebt, was den Erfolg von sogenannten dokumentenorientierten Datenbanken wie MongoDB begründet.

Die JSON-Funktionalität stellt hauptsächlich Funktionen bereit, um mit JSON-Daten zu arbeiten. Pfadausdrücke erlauben den Zugriff auf einzelne Bestandteile eines JSON-Dokuments mit SQL. Dadurch ist auch eine Indizierung möglich. Für viele Anwendungen eröffnen sich damit neue Möglichkeiten für einen schlanken und flexiblen Schema-Entwurf, ohne auf ein NoSQL-System umsteigen zu müssen.

Fazit

Von den vielen neuen Technologien für Big Data konnten hier nur einige wenige angesprochen werden. Da sich viele Produkte in Bezug auf Leistungsfähigkeit und Funktionalität überschneiden, ist die Konzeption einer Big Data Architektur eine anspruchsvolle Aufgabe, welche durch die Dynamik der Produktentwicklung zusätzlich erschwert wert.

Basis einer Big-Data-Architektur bildet meist Hadoop aufgrund der Möglichkeiten zur kostengünstigen Skalierung und verteilte Datenverarbeitung. Aber SQL wird auf lange Sicht die

beliebteste Sprache bleiben, mit der Daten, egal ob klein oder groß, analysiert werden. Der Entwicklungstrend geht daher verstärkt zu SQL-Datenbanken, die direkt auf Hadoop aufsetzen und für Big Data optimiert sind. Im Gegenzug kontern die klassischen General-Purpose-Datenbanken wie Oracle mit integriertem Hadoop-Zugriff.

Referenzen

[1] Big Data for the Enterprise. Oracle Whitepaper, 2013
http://www.oracle.com/us/products/database/big-data-for-enterprise-519135.pdf

[2] Big-Data-Technologien: Wissen für Entscheider. Bitkom, 2014
http://www.bitkom.org/files/documents/BITKOM_Leitfaden_Big-Data-Technologien-Wissen_fuer_Entscheider_Febr_2014.pdf

[3] Oracle NoSQL vs. HBase and Cassandra
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-cassandra-hbase-1961726.pdf

[4] Oracle Technology Global Price List. Oracle, 2014
http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

[5] Oracle Database In-Memory. Oracle Whitepaper, 2014
http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html

Kontaktadresse:

Prof. Dr. Jens Albrecht
Technische Hochschule Nürnberg
Postfach 210320
D-90121 Nürnberg

Telefon:	+49 (0) 9128-400 3812
Fax:	+49 (0) 911-5880 566
E-Mail	jens.albrecht@th-nuernberg.de
Internet:	www.th-nuernberg.de