Big Data - Massenserialisierung mit Apache Hadoop

Ralph Tröger, GS1 Germany GmbH, Köln
Marcel Amende, ORACLE Deutschland B.V. & Co. KG, Düsseldorf

Schlüsselworte
Big Data, Apache Hadoop, GS1, EPC, GPS, Logistik

Einleitung

„Big Data“ ist aktuell der heißeste Trend in der IT-Branche. Angetrieben vom durchschlagenden Erfolg der Big Data-Anwendungen bei Internet-Schwergewichten wie Google (Aufbau des Suchindexes), Amazon (Ermittlung personalisierter Produktvorschläge) oder Facebook („Social Graph“-Generierung aus Mediendaten) wächst das Interesse auch in klassischen Branchen. Hier ergeben sich völlig neue Analyseansätze: Erstmals ist es möglich, immens große (bspw. im Petabyte-Bereich liegende) und unstrukturierte Datenmengen kostengünstig zu verwalten, zu verarbeiten und nach immer neuen Gesichtspunkten zu analysieren. Voraussetzung für die Verarbeitung von Big Data ist die Identifizierung von Schlüsselinformationen in den Datenströmen, wie z. B. Identifikationsnummern.

Problemstellung: steigende Datenvolumina in der Logistik

In Handel, Logistik und Industrie ist die Massenserialisierung von Objekten (u. a. Produkte, Sendungs- oder Transporteinheiten) einer der Megatrends der Stunde. Durch das konsistente Set an Standards von GS1 – angefangen von der individuellen Kennzeichnung von Geschäftsobjekten mittels Electronic Product Code (EPC) bis hin zu modernen Schnittstellenstandards wie EPCIS (EPC Information Services) – werden Warenströme transparent und durchgängig verfolgbar.

Die technologischen Voraussetzungen sind somit gegeben, Geschäftsprozesse feingranular, d. h. bis auf Instanz-Ebene, mess- und überwachbar zu machen, daraus Optimierungspotenziale abzuleiten und letztlich sogar komplett neue Geschäftsmodelle zu entwickeln. Die Kehrseite der Medaille ist jedoch eine hohe Zunahme des Datenvolumens, da statt der rein mengenmäßigen Dokumentation von Warenbewegungen („420 Stück von Artikel 4711“) Einzelteilidente („Artikel 4711 mit Seriennummern 12, 98, 678, 679, …“) übertragen werden.

Allerdings nimmt das Datenvolumen nicht allein durch Massenserialisierung zu: Informationen zu Geschäftsprozessen werden entlang der Lieferkette von den verschiedensten Technologien zur automatischen Identifikation (Barcode- und RFID-Lesegeräte, Telematikeinheiten, Sensoren, etc.) quasi „im Vorbeigehen“ erfasst und in den verschiedensten Formaten übermittelt (vgl. Abb. 1).

Meist werden diese Daten allerdings nur im Zusammenhang mit einer lokalen Applikation bzw. in sehr eingeschränktem Umfang (bspw. Daten von Kassenscannern zur Produktidentifikation) genutzt. Der Wert dieser Daten steigt jedoch immens, wenn man in der Lage ist, sie miteinander in Beziehung zu setzen, um Muster (z. B.: „War die verspätete Lieferung durch einen Stau oder durch fehlende Lagerbestände verursacht?“) zu erkennen.

Herkömmliche Datenverarbeitungstechnologien sind nicht perfekt geeignet, hochvolumige und unstrukturierte Daten zu speichern bzw. zu prozessieren. Die Ablage von massenhaft uninterpretierten Rohdaten in relationalen Datenbanksystemen ist schlichtweg teuer, ein klassisches Filesystem ist fehleranfällig und schwer handhabbar. Diese Lücke schließt das Apache Hadoop Framework mit dem Hadoop Distributed Filesystem (HDFS).

Apache Hadoop Framework – Hadoop Distributed Filesystem
Nehmen wir einen Logeintrag einer GPS-basierten Telematikeinheit als Beispiel, welche schon heute häufig von Logistikdienstleistern in LKWs, Wechselbrücken oder Containern mitgeführt werden. Ein einfacher Wegpunkt lässt sich durch wenige hundert Bytes beschreiben:

```
<wpt lat="42.578917" lon="-75.116146">
  <ele>44.826904</ele>
  <time>2012-11-11T11:11:00Z</time>
  <name>urn:epc:id:giai:4012345.667788</name>
  <sym>Dot</sym>
  <type>Dot</type>
</wpt>
```

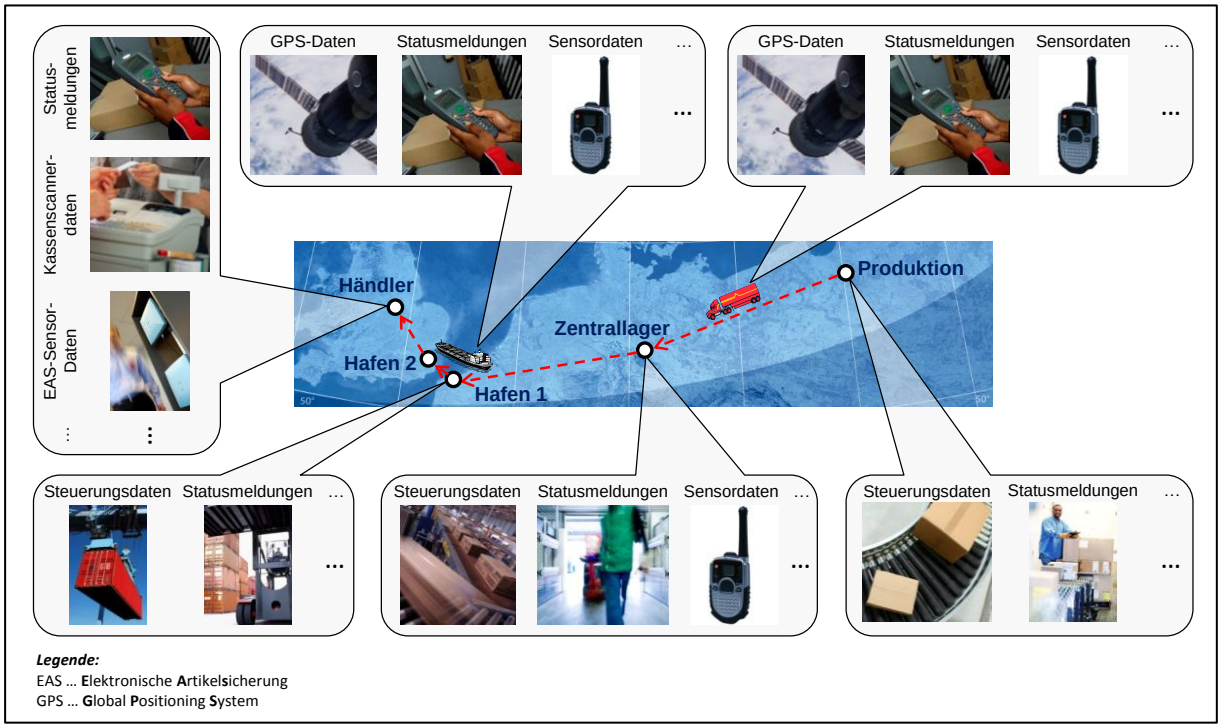


Abb. 1: Beispiele für Big Data in einer Lieferkette

Bei sekündlicher Positionsaufnahme würden sich die Logdateien über ein Jahr hinweg schon bei einer kleinen Fahrzeugflotte von 50 LKWs und einer durchschnittlichen täglichen Fahrzeit von 9 Stunden auf über 100 GB summieren. Die Datei enthält auf den ersten Blick eine Menge von Redundanzen, jedoch lassen sich aus diesem Datenstrom – insbesondere bei Korrelation z. B. mit Standort- und Lieferinformationen – eine Vielzahl von Erkenntnissen gewinnen:

- Wie war die Auslastung der Ressource?
- Wie war die Verfügbarkeit von Ressourcen an bestimmten Standorten?
- Welche Strecken waren besonders staugefährdet?
- Wurden Geschwindigkeiten und Ruhezeiten eingehalten?
- Wie hoch war die Liefertreue?

Der Wert solcher Antworten für eine Planungs- bzw. Routenoptimierung sollte den Aufwand für eine zentrale Datenhaltung rechtfertigen, wenn diese praktikabel gestaltet werden kann.

Das verteilte Hadoop Dateisystem eignet sich in diesem Sinne ideal: Große Dateien werden in Blöcken (N x 64MB) verwaltet und von den einzelnen Knoten über verschiedene Server hinweg über ein TCP/IP-basiertes Protokoll repliziert. Die Anforderungen an die Wahrscheinlichkeiten von Datenverlust bestimmt die Konfiguration der Anzahl von Kopien (2 - N) der Datenblöcke im Netzwerk. Datenänderungen (Updates) sind in diesem auf die Verarbeitungsgeschwindigkeit ausgerichteten System konzeptionell nicht vorgesehen. Dateien werden über die native Java API vor allem statisch gespeichert und wiederholt lesend zugegriffen.

Immense Datenmengen verlangen nach einem völlig neuen Programmiermodell, mit dem die Datenverarbeitung massiv parallel in großen Gruppen von Rechnereinheiten erfolgen kann. Dies bietet der MapReduce-Mechanismus des Apache Hadoop Frameworks.

Apache Hadoop Framework – MapReduce
Das MapReduce-Programmiermodell gliedert sich in drei Verarbeitungsschritte: „Map“, „Shuffle/Sort“ und „Reduce“ (siehe Abbildung 2).

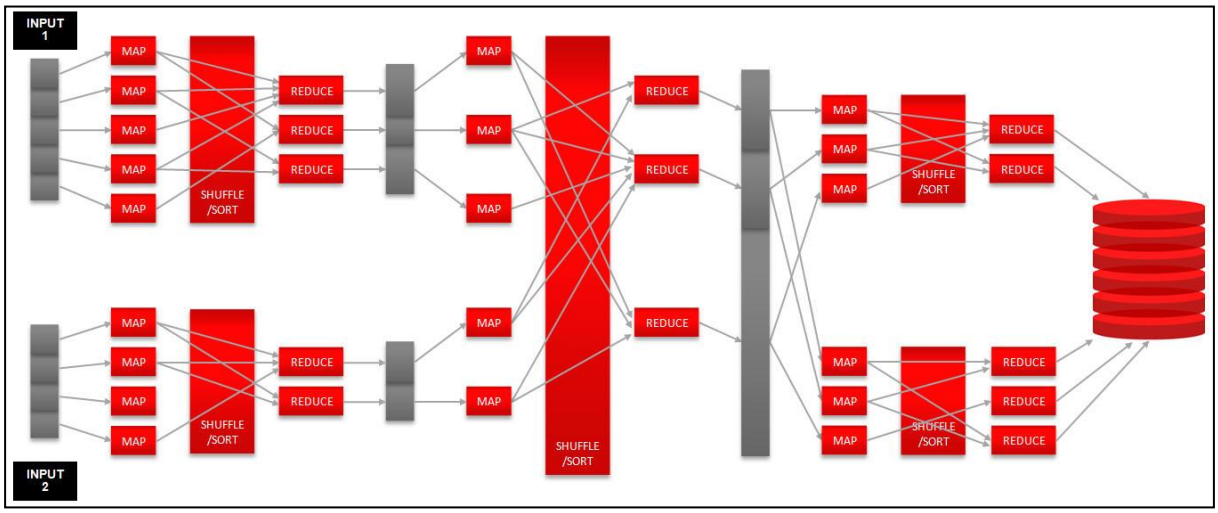


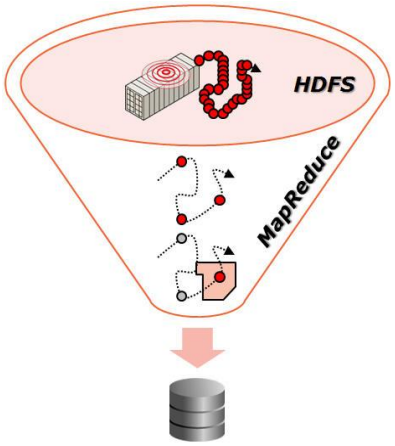
Abb. 2: Funktionsprinzip von Apache Hadoop

Im ersten Verarbeitungsschritt „Map“ werden massiv parallele Verarbeitungsthreads auf allen Knoten gestartet, die Dateien aus den HDFS-Blöcken möglichst nahe ihrer Speicherung, d. h. idealerweise auf dem lokalen Knoten, in ein Key/Value-Paar verarbeiten. In jeder Datei wird also nach einem im Sinne des jeweiligen Verarbeitungsprozesses sinnvollen Schlüsselwert gesucht. Der „Shuffle/Sort“-Algorithmus sammelt anschließend alle Paare mit demselben Schlüssel und startet jeweils einen eigenen Thread im Verarbeitungsschritt „Reduce“. Hier stehen alle zusammengehörigen Daten als Vektor zur Verfügung, um diese zu interpretieren und auf ein bestimmtes Ergebnis zu reduzieren.

Je nach Daten- und Verarbeitungskomplexität kann diese Verarbeitung kaskadierend erfolgen. Als Ziel für die identifizierten geschäfts- bzw. prozessrelevanten Informationen bieten sich neben dem HDFS selbst für die weitere Verarbeitung und Interpretation nun Data Warehouse, relationale und NoSQL-Datenbanken an.

Kommen wir auf das obige Beispiel der Transportverfolgung von Wechselbrücken mittels GPS-Logdateien zurück. Hier lassen sich drei Informationsquellen erschließen: Zum Einen beschreibt die Gesamtheit der GPS-Logeinträge das Bewegungsmuster der mittels GIAI (Global Individual Asset

Identifier) identifizierbaren Wechselbrücken. Zum Anderen sind geplante Standorte und Fahrziele der Wechselbrücken durch räumliche Polygone beschreibbar. Schlussendlich existiert zu den geplanten Bewegungen im Normalfall immer ein entsprechender Auftrag (Purchase Order) in einem Anwendungssystem.



Über einen mehrstufigen MapReduce-Verarbeitungsprozess können nun geschäftsrelevante Ereignisse aus den Datenbeständen gefiltert werden: U. a. lässt sich das Verweilen an einem fixen Standort ab einer bestimmten Dauer als Absetzen einer Wechselbrücke interpretieren. Die kontinuierlichen GPS-Daten können daher in einem ersten Schritt auf Verweilpunkte und -dauer reduziert werden. Desweiteren sind nur diejenigen Verweilpunkte relevant, die in das Raumpolygon eines Betriebshofs fallen. Existiert nun für einen speziellen Verweilzeitraum ein Transportauftrag mit passendem Ziel, ist eine Geschäftstransaktion (bspw. „Auslieferung“) automatisch zuordnungsfähig. Das diskrete Geschäftsereignis lässt sich nun strukturiert beschreiben und in ERP-, Datenbank- oder Business Intelligence-Systemen weiterverarbeiten. Für die Beschreibung dieser Ereignisdaten bietet sich insbesondere der EPCIS-Standard von GS1 an (siehe nachfolgende XML-Struktur eines EPCIS-Transaktionsevents):

```
<TransactionEvent>
  <eventTime>2012-11-22T14:34:39.172+01:00</eventTime>
  <bizTransactionList><!-- Verweis auf Auftrag -->
    <bizTransaction type="urn:epcglobal:cbv:btt:po">
    http://doag.org/po/12345678</bizTransaction>
  </bizTransactionList>
  <parentID>urn:epc:id:giai:4012345.667788</parentID><!-- ID der
Wechselbrücke -->
  <epcList><!-- Inhalt der Wechselbrücke -->
    <epc>urn:epc:id:sscc:4012345.0999999998</epc>
    <epc>urn:epc:id:sscc:4012345.0999999999</epc>
  </epcList>
  <action>OBSERVE</action>
  <!-- Geschäftsvorfall ‚Auslieferung‘ -->
  <bizStep>urn:epcglobal:cbv:bizstep:accepting</bizStep>
  <disposition>urn:epcglobal:cbv:disp:in_progress</disposition>
  <readPoint><id>urn:epc:id:sgln:4012341.23456.12568</id></readPoint>
  <bizLocation><id>urn:epc:id:sgln:4000001.23456.25</id></bizLocation><!--
ID Betriebshof -->
</TransactionEvent>
```

Oracle Big Data Appliance

Die Oracle Big Data Appliance bringt mit der Cloudera Distribution die einzig verfügbare und vorinstallierte Apache Hadoop Distribution am Markt mit. Diese wird durch eine “R” Distribution zur Datenanalyse ergänzt, der Oracle NoSQL-Datenbank und einer Reihe von Adaptern u. a. zur Anbindung von Hadoop an Oracle Datenbanken, dem Oracle Data Integrator und “R”.

Für die Umsetzung von Big Data-Konzepten benötigt man neben der Software auch die passende Hardware: Für die Speicherung der unstrukturierten Daten in einem Hadoop Filesystem braucht man eine Menge Plattenkapazität. Die Datenverarbeitung über Map-Reduce-Mechanismen benötigt immense parallelisierbare Rechenleistung. Darüber hinaus erfordert die Datenreplikation und die Weiterverarbeitung relevanter Informationen in Datenbank- und Warehouse-Systemen ein schnelles Netzwerk.

Genau dies spiegelt sich im Aufbau der Oracle Big Data Appliance wider, die als sofort einsatzbereites Komplettsystem aus Hard- und Software geliefert wird: 18 SunFire Server mit jeweils 12 Festplatten à 12 TB und 12 Cores sind über ein latenzfreies 40GB/s Infiniband Netzwerk miteinander verbunden. Ein einzelnes Big Data-Rack bietet somit mehr als ein halbes Petabyte Speicherkapazität und 216 Cores für die Parallelisierung der Verarbeitungslogik.

Durch Kombination mehrerer Systeme sind diese Parameter nahezu beliebig skalierbar. Je nach Anwendungsfall bietet sich auch die Erweiterung mit einem Exadata Engineered System für die strukturierte Speicherung der Ergebnisdaten, einem Exalytics Engineered System für die Ergebnisdatenauswertung und einem Exalogic Engineered System für die Datenintegration über das Infiniband Netzwerk an.

Kontaktadressen:

Ralph Tröger
GS1 Germany GmbH
Maarweg 133
D-50825 Köln
Telefon: +49 (0) 221-94714 243
E-Mail: troeger@gs1-germany.de
Internet: www.gs1-germany.de

Marcel Amende
ORACLE Deutschland B.V. & Co. KG
Hamborner Str. 51
D-40472 Düsseldorf
Telefon: +49 (0) 211-74839 539
E-Mail: Marcel.Amende@oracle.com
Internet: www.oracle.de