Bild: Shutterstock

Sonderdruck aus
BI-SPEKTRUM 5/2018

Echtzeitdatenverarbeitung in Zeiten der digitalen Transformation

Ganzheitliche, eventbasierte Streaming-Strategie

Ein Beitrag von
Matthias Hoffmann

In den Bereichen Data Analytics, Data Science und Machine Learning haben viele Unternehmen bisher eher Batch-orientierte Ansätze verwendet, da hier der technische und organisatorische Komplexitätsgrad deutlich geringer ist als bei einem Streaming-orientierten Ansatz. Der Wert von Informationen hängt aber stark von der Aktualität ab. Die Batch-orientierte Datenverarbeitung und -analyse reicht daher zunehmend nicht mehr aus, um wettbewerbsfähig zu sein. Häufig werden Use-Cases, bei denen Daten in Echtzeit verarbeitet werden müssen, als Insellösungen umgesetzt, das heißt, bei jedem Streaming-Use-Case wird praktisch alles neu konzipiert und umgesetzt. Dies führt zu hohen Kosten in den Bereichen Design, Implementierung, Test, Betrieb und Wartung. Verfolgt ein Unternehmen hingegen eine ganzheitliche, standardisierte Streaming-Strategie, die sich an den Geschäftsvorfällen des Unternehmens (Events) orientiert, können der Aufwand und die Kosten stark reduziert und die Entwicklung auf die fachlichen Herausforderungen konzentriert werden.

Was ist die Information, dass ein Kunde eine hohe Kaufwahrscheinlichkeit für ein Produkt hat, noch wert, wenn der Kunde das Produkt vor einer Stunde bei einem Wettbewerber gekauft hat?

Das Rennen, die neuesten Informationen möglichst in Echtzeit auszuwerten, um hieraus die richtigen Aktionen abzuleiten (Next-Best Action), läuft schon seit vielen Jahren. Aufgrund der digitalen Transformation der Unternehmen gibt es heute aber viel mehr Anwendungsfälle, bei denen Daten in Echtzeit verarbeitet werden müssen, und daher erfährt das Thema Fast Data und Streaming einen regelrechten Hype. Durch die Agilität der Märkte entsteht ein ständig wachsender Bedarf an Informationen, die in Echtzeit zur Verfügung stehen müssen.

Insellösungen für Anwendungsanfälle reichen da nicht mehr. Erforderlich ist eine konsequente ganzheitliche, standardisierte Vorgehensweise für die End-to-End-Verarbeitung von Echtzeitdaten

– von der Entgegennahme über die Verarbeitung bis hin zur Bereitstellung der Ergebnisse. Für die erfolgreiche Umsetzung einer Streaming-Strategie spielt neben dem technischen ganzheitlichen Ansatz die Fokussierung auf die fachlichen Geschäftsvorfälle (Events) des Unternehmens eine entscheidende Rolle.

Technische Architektur

Die Herausforderung beim Aufbau einer ganzheitlichen, standardisierten und robusten Streaming-Plattform besteht darin, zunächst einen „Best of Breed"-Technologie-Stack für die Anforderungen des Unternehmens zu bestimmen.

Die rasante Innovationskraft im Big-Data-Bereich hat in den letzten Jahren einige Software-Lösungen hervorgebracht, die für die Umsetzung von Streaming-Anforderungen optimiert wurden. Die Konzeption eines Big-Data-Systems auf Basis der Lambda-Architektur gilt heute als Standard. Der Lambda-Architekturansatz stammt von Nathan Marz [MaW15] und ermöglicht die echtzeit- und Batch-basierte Datenverarbeitung auf Basis von Big-Data-Technologien. Die Nachteile einer Lambda-Architektur liegen hauptsächlich in der unterschiedlichen Aktualität der Daten im Speed- und Batch-Layer und der mehrfach implementierten Verarbeitungslogik. Deshalb ist der Betrieb einer solchen Umgebung komplex. Diese Probleme löst der sogenannte SMACK-Stack [SMA17], der in den letzten Jahren populär geworden ist. Der SMACK-Stack ist kein eigenständiger Architekturansatz, sondern steht für die Verwendung eines aufeinander abgestimmten Tool-Frameworks (S=Spark, M=Mesos, A=Akka, C=Cassandra, K=Kafka) für die Datenverarbeitung im Big-Data-Umfeld.

Abbildung 1 zeigt eine mögliche technische Architektur, mit der eine ganzheitliche, event-basierte Streaming-Plattform umgesetzt werden kann. Der hier dargestellte Architekturansatz basiert ausschließlich auf frei verfügbaren Open-Source-Komponenten, am Markt sind jedoch auch verschiedene kommerzielle Produkte erhältlich, die für den Aufbau einer Event-basierten Streaming-Plattform verwendet werden können.

Daten können von verschiedenen internen und externen Systemen (1) geliefert werden. Die technische Anbindung hängt von den Quellsystemen, Unternehmensvorgaben und den konkreten Anforderungen ab.

Das Kernstück einer Lambda-Architektur stellt der Hadoop-Cluster (2) dar. Hadoop ist ein Open-Source-Produkt, das frei verfügbar ist und einzeln installiert werden kann. Für Produktivumgebungen setzen viele Unternehmen aber auf kommerzielle Hadoop-Distributionen. Hadoop-Distributionen haben den Vorteil, dass bereits viele Standard-Tools aus dem Hadoop-Ökosystem in der Distribution enthalten sind und dass vom Anbieter sichergestellt wird, dass alle Komponenten zueinander kompatibel sind. Außerdem bieten Hadoop-Distributionen weitere Tools und Funktionen an, die den Betrieb und das Management eines Big-Data-Systems erheblich vereinfachen.

Die Speicherung von Daten aus den Quellsystemen erfolgt im Data Lake (3). Dem Data Lake liegt ein verteiltes Dateisystem zugrunde, das Hadoop Distributed Filesystem (HDFS). Im HDFS werden alle Daten als Datei abgelegt. Das Dateiformat spielt hierbei keine Rolle. Ein Best-Practice-Ansatz unterteilt den Data Lake in zwei Zonen: eine Landing Zone und eine Harmonization Zone. In der Landing Zone werden alle Daten unverändert ohne Transformationen abgelegt. Da es innerhalb eines Big-Data-Systems in der Regel viele Anwendungsfälle gibt, die auf die gleichen Daten-Sets des Data Lake zugreifen, ist es sinnvoll, oft referenzierte Daten-Sets in ein einheitliches, für Big Data optimiertes Dateiformat (z.B. Parquet, Avro) zu transformieren. So lassen sich die Entwicklungskosten und die Systemkomple-

MATTHIAS HOFFMANN ist seit über 20 Jahren im Business-Intelligence-Bereich tätig und beschäftigt sich seit mehr als sieben Jahren intensiv mit dem Thema Big Data. Er ist Partner bei integration-factory und verantwortet dort den Bereich Big Data und das Big Data Research Center, in dem die neuesten Trends im Big-Data- und Hadoop-Umfeld untersucht werden.

E-Mail: hoffmann@integration-factory.de

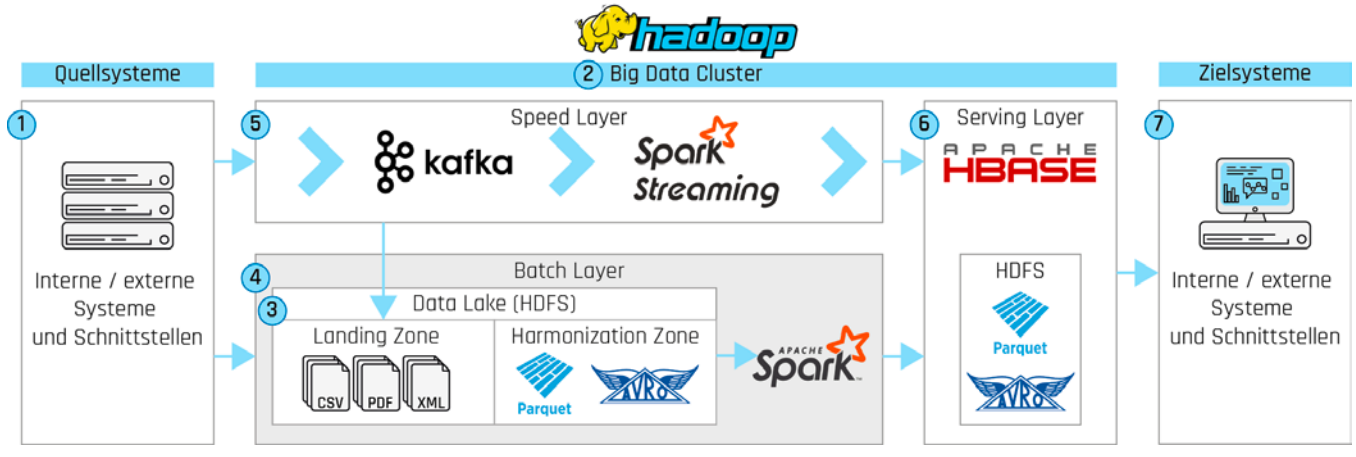

Abb. 1: Big-Data-Lambda-Architektur

xität reduzieren. Die Dateninhalte werden hierbei in der Regel nicht verändert. Diese Art der Format-Harmonisierung geschieht in der Harmonization Zone.

Das grundlegende Konzept einer Lambda-Architektur sieht zwei Layer für die Datenverarbeitung vor, den Batch-Layer und den Speed-Layer. Über den Batch-Layer ❹ werden Daten im Batch-Modus verarbeitet. Der Batch-Layer greift auf Daten aus dem Data Lake zu. Im Allgemeinen werden über den Batch-Layer komplexe Berechnungsprozesse, die auf größere Datenmengen angewendet werden, abgewickelt. Ein Beispiel hierfür ist das Trainieren eines Machine-Learning-Modells mit Massendaten aus den letzten zehn Jahren. Für die Entwicklung und Ausführung der Datenverarbeitungsprozesse wird zumeist das Apache-Spark-Framework genutzt, das mittlerweile den Standard darstellt und in jeder Hadoop-Distribution enthalten ist. Mit Spark lassen sich komplexe, verteilte Berechnungen auf großen Datenmengen effizient durchführen.

Im Gegensatz zum Batch-Layer werden im Speed-Layer ❺ Daten in Echtzeit verarbeitet. Die zentrale Komponente für den Empfang von Daten ist die Streaming-Plattform Apache Kafka. Mit Kafka können Datenströme in sogenannten Topics organisiert und in Echtzeit bereitgestellt werden. Kafka zeichnet sich unter anderem durch seinen hohen Nachrichtendurchsatz, die Skalierbarkeit und seine Fehlertoleranz aus.

Das zunächst für die Batch-Verarbeitung konzipierte Spark-Framework wurde um eine Streaming-Komponente erweitert, um Daten in Echtzeit verarbeiten zu können. Mit Spark Streaming ist es möglich, Datenströme von Kafka entgegenzunehmen, darauf Berechnungen durchzuführen und die Ergebnisse dann für Datenkonsumenten bereitzustellen. Alternativ zu Spark Streaming können für die Verarbeitung von Datenströmen auch andere Frameworks wie beispielsweise Apache Storm [Sto18] eingesetzt werden. In dem hier vorgestellten Ansatz wird Spark Streaming eingesetzt, da Spark fester Bestandteil jeder Hadoop-Distribution ist und durch die Nutzung eines einheitlichen Frameworks für die Batch- und Echtzeit-Verarbeitung Synergien in den Bereichen Tool-Know-how, Entwicklung, Betrieb und Wartung entstehen. Im Bereich des Machine Learning bietet Spark Streaming außerdem die Möglichkeit, Machine-Learning-Modelle, die in der Spark Machine Learning Library MLib enthalten sind, im Streaming-Modus zu nutzen.

Im Serving-Layer ❻ werden die im Batch- und Speed-Layer berechneten Ergebnisse gespeichert und für Datenkonsumenten bereitgestellt. Die Datenspeicherung kann durch Ablage der Ergebnisse als Datei im HDFS erfolgen. Auf diesen Dateien kann dann beispielsweise mit Apache Hive [Hiv18] eine Metadatenschicht definiert werden, die einen einfachen, an SQL angelehnten Datenzugriff ermöglicht.

Diese Vorgehensweise ist sinnvoll, wenn Business-Analysten auf die Daten zugreifen wollen oder die Daten für das Reporting genutzt werden. Für Anwendungsfälle, bei denen ein schnelles Antwortzeitverhalten bei Abfragen entscheidend ist, ist eine Ablage der Daten im HDFS dagegen nicht sinnvoll. In der hier gezeigten Architektur wird Apache HBase verwendet, um einen Zugriff auf große Datenmengen mit schnellen Antwortzeiten zu realisieren. HBase ist ein verteilter, spaltenbasierter Key-Value-Datenspeicher, der auf Basis des HDFS operiert und bereits in allen gängigen Hadoop-Distributionen enthalten ist. HBase ist in der gezeigten Architektur der Datenspeicher des Speed-Layers.

Die im Serving-Layer bereitgestellten Daten können beliebigen Datenkonsumenten ❼ zur Verfügung gestellt werden. Datenkonsumenten können sowohl interne und externe Systeme und Schnittstellen als auch Endanwender sein. Die technischen Möglichkeiten der Bereitstellung sind hierbei nahezu unbegrenzt: von der einfachen Dateilieferung über Webservices bis hin zur direkten Weiterleitung von Informationen in andere Messaging-Systeme in Echtzeit.

Eventbasierter, ganzheitlicher Speed-Layer

Was in Abbildung 1 dargestellt und im vorigen Abschnitt beschrieben wurde, ist die technische Architektur eines Big-Data-Systems, mit der die Datenverarbeitung sowohl in Echtzeit als auch im Batch-Modus möglich ist. Die richtige technische Architektur ist entscheidend für den Aufbau einer Streaming-Plattform. Sie verhindert jedoch nicht, dass Streaming-Use-Cases isoliert betrachtet, konzipiert und umgesetzt werden und dass hierdurch die Systemkomplexität steigt und hohe Kosten in den Bereichen Entwicklung, Test und Betrieb entstehen. Eine ganzheitliche, eventbasierte Streaming-Strategie verfolgt dagegen das Ziel, eine einheitliche und standardisierte prozessuale und technische Vorgehensweise für die Echtzeitdatenverarbeitung zu nutzen, bei der die fachlichen Geschäftsvorfälle des Unternehmens im Fokus stehen.

Technisch lässt sich ein solcher Streaming-Ansatz mit einem standardisierten eventbasierten Speed-Layer umsetzen. Abbildung 2 zeigt einen Speed-Layer, der sich an den Geschäftsvorfällen des Unternehmens orientiert.

Ein standardisierter eventbasierter Speed Layer setzt zunächst voraus, dass eine einheitliche Datenschnittstelle für alle Anwendungsfälle geschaffen wird, um direkte Verbindungen zwischen den Quellsystemen und den Anwendungsfällen zu vermeiden. Dies kann erreicht werden, indem ein einheitliches, standardisiertes Datenformat ❶ auf Basis der Geschäftsvorfälle des Unternehmens modelliert wird. Geschäftsvorfälle können hierbei beispielsweise die Neukunden-Anlage, die Änderung einer Kundenadresse, die Kündigung von Verträgen, aber auch die Synchronisation von Daten aus Geschäftsvorfällen zwischen internen Systemen sein.

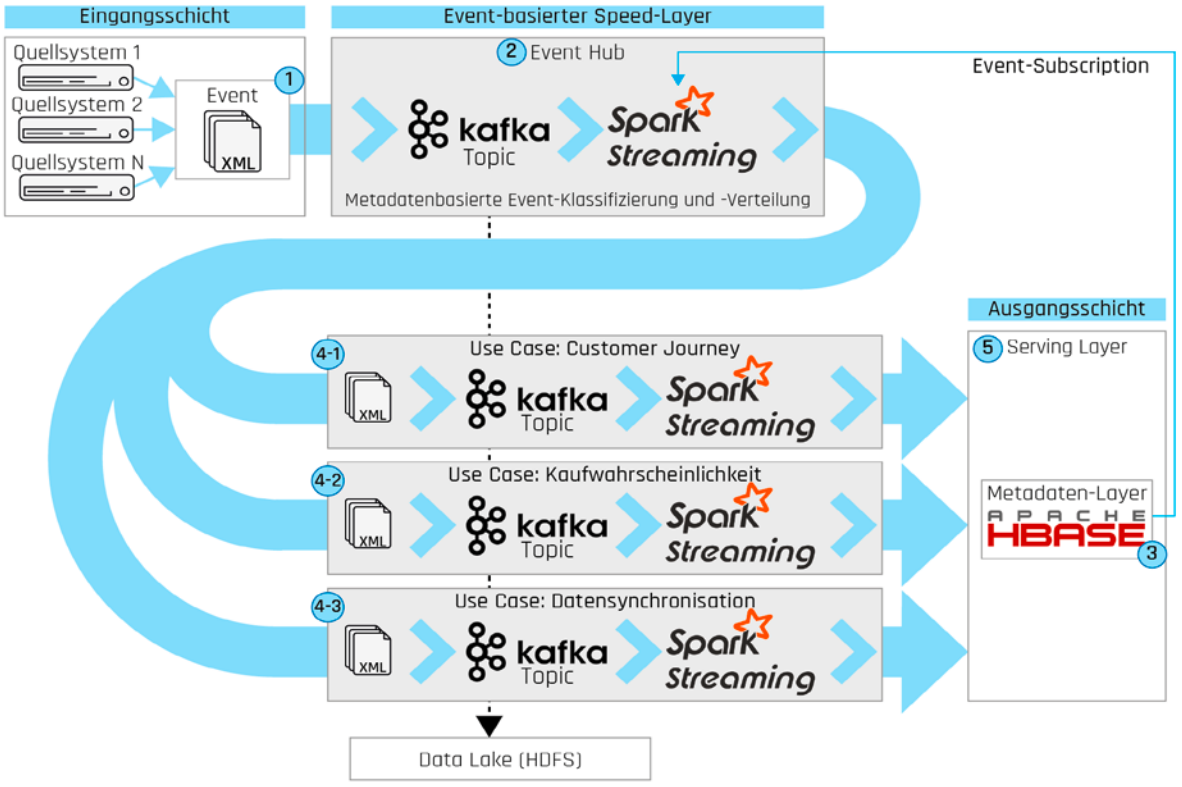

Abb. 2: Standardisierter eventbasierter Speed-Layer

Technisch kann der eventbasierte Datenaustausch über das XML-Format abgebildet werden. Die Beschreibung der verschiedenen Geschäftsvorfälle und ihrer Datenelemente werden in einer XML Schema Definition (XSD) festgelegt.

Die größte Herausforderung bei der Schaffung einer einheitlichen Datenschnittstelle für Geschäftsvorfälle liegt darin, dass alle Quellsysteme ihre Geschäftsvorfälle in das einheitliche Format transformieren müssen. Der Aufwand und die technische Umsetzung hängen hierbei vom jeweiligen System ab. Verwendet ein Quellsystem beispielsweise eine Oracle-Datenbank, so kann bei einer Datenänderung in der Datenbank über ein automatisiertes Change-Data-Capture-Verfahren die Konvertierung in das standardisierte Datenformat erfolgen.

Die zentrale Komponente des eventbasierten Speed-Layers ist der Event-Hub ❷. Im Event-Hub werden zunächst die Geschäftsvorfälle im standardisierten Datenformat über Kafka im Streaming-Modus entgegengenommen. Mittels Spark Streaming werden schließlich die Nachrichten aus Kafka gelesen, die Geschäftsvorfälle anhand des Geschäftsvorfalltyps zentral klassifiziert und an die entsprechenden Kafka-Topics der Use-Cases verteilt.

Die Verteilung der Geschäftsvorfälle erfolgt über eine flexibel konfigurierbare Subskription, mit der Use-Cases Geschäftsvorfälle abonnieren können. Die Subskriptions-Informationen für jeden Use-Case werden hierbei als Parameter in einem Metadaten-Layer ❸ hinterlegt. Der Metadaten-Layer wird technisch durch HBase innerhalb des Serving-Layers umgesetzt und vom Event-Hub zur Laufzeit referenziert.

Die zentrale Aufgabe des Event-Hubs ist also die Weiterleitung der Geschäftsvorfälle in dem einheitlichen, standardisierten Datenformat an die Kafka-Topics der Use-Cases. Aus Revisionsgründen und für das Qualitätsmanagement werden die über den Event-Hub empfangenen Geschäftsvorfälle auch zusätzlich im Data Lake als Datei abgelegt und somit ein Event-Archiv erstellt. Die konkreten fachlichen Anforderungen des speziellen Use-Case werden mit Spark Streaming umgesetzt. Anschließend können die Ergebnisse wieder über den Serving-Layer für die Datenkonsumenten bereitgestellt werden.

Die so etablierte zentrale Verteilungsfunktion des Event-Hubs kann des Weiteren einfach für einen Datenaustausch zwischen den Use-Cases genutzt werden. Benötigt beispielsweise Use-Case 2 Informationen, die in Use-Case 1 berechnet werden, kann Use-Case 1 die entsprechenden Daten als Geschäftsvorfall in dem standardisierten Dateiformat an den Event-Hub senden. Dieser verteilt dann den Geschäftsvorfall an den Use-Case 2 in Echtzeit.

In Abbildung 2 sind beispielhaft drei Use-Cases ❹ dargestellt, die auf Basis eines eventbasierten Speed-Layers umgesetzt werden können:

- Use Case 1: Customer Journey/Custom 360°: Aktualisierung einer 360°-Kundensicht in Echtzeit
- Use Case 2: Berechnung von Produkt-Kaufwahrscheinlichkeiten für Kunden durch den Einsatz von Machine-Learning-Verfahren
- Use Case 3: Datensynchronisation zwischen Bestandssystemen in Echtzeit

Lambda – Kappa – SMACK

Ein eventbasierter, ganzheitlicher Streaming-Ansatz zielt darauf ab, Daten in Echtzeit zu verarbeiten. Daher stellt sich die Frage, warum als technische Architektur für das Big-Data-System

die Verwendung einer Lambda-Architektur empfehlenswert ist. Eine Plattform, die vollständig auf Fast Data / Streaming ausgelegt ist, folgt einer Kappa-Architektur. Der Kappa-Architektur-Ansatz wurde erstmalig von Jay Krebs beschrieben [Kre14]. Vereinfacht ausgedrückt, ist eine Kappa-Architektur eine Lambda-Architektur ohne den Batch-Layer.

Für den Aufbau einer ganzheitlichen, eventbasierten Streaming-Plattform empfiehlt es sich jedoch, eine Lambda-Architektur zu verwenden, da in der Praxis häufig unterstützend die Batch-Verarbeitung und ein Data Lake benötigt werden. Aus Revisionsgründen ist beispielsweise das Erstellen eines Event-Archivs und Event-Logs im Data Lake zwingend notwendig. Weitere Anwendungsfälle, die einen Batch-Layer notwendig machen, sind das Trainieren von Machine-Learning-Modellen, die anschließend in Echtzeit genutzt werden sollen, oder das Prozessieren von Event-Historien.

Eine gute und innovative Alternative zur Lambda-Architektur stellt der SMACK-Ansatz dar, besonders wenn sehr große Datenmengen in Echtzeit verarbeitet und bereitgestellt werden müssen. Mit einem darauf basierenden System ist es beispielsweise möglich, mehr als eine halbe Million Messages pro Sekunde zu verarbeiten.

Fazit: Ganzheitlich statt Insellösung!

Unternehmen, die heute in Zeiten der digitalen Transformation in der Lage sind, ihre Daten unkompliziert und ganzheitlich in Echtzeit auszuwerten, um auf dieser Basis schnell und zielgerichtet die richtigen Aktivitäten abzuleiten, haben einen entscheidenden Wettbewerbsvorteil. Aufgrund der Vielzahl von verfügbaren Streaming-Lösungen, der Skalierbarkeit, Performanz und Fehlertoleranz eignet sich Big-Data-Technologie sehr gut als technisches Fundament für den Aufbau einer Streaming-Plattform.

Die richtige technische Architektur reicht jedoch nicht aus, um Insellösungen, also die isolierte Betrachtung und Umsetzung von Streaming-Use-Cases, zu verhindern. Abhilfe schafft hier ein ganzheitlicher, eventbasierter Streaming-Ansatz, bei dem die fachlichen Geschäftsvorfälle des Unternehmens in den Fokus gestellt werden und eine hochgradig standardisierte, technische Vorgehensweise für die Umsetzung von Streaming-Use-Cases verwendet wird. Mit diesem ganzheitlichen, unternehmensweiten Blick auf die Echtzeitdatenverarbeitung lassen sich der Aufwand, die Kosten und die Realisierungszeit (Time-To-Market) für Streaming-Use-Cases erheblich reduzieren.

Literatur

[Hiv18] Apache Hive: http://hive.apache.org/

[Kre14] Kreps, J.: Questioning the Lambda Architecture. O'Reilly Media 2014, https://www.oreilly.com/ideas/questioning-the-lambda-architecture, abgerufen am 26.10.2018

[MaW15] Marz, N. / Warren, J.: Big Data: Principles and best practices of scalable real-time data systems. Manning Publications Co. 2015

[SMA17] McFadin, P.: The SMACK Stack – A new architecture for today's data-rich modern applications. O'Reilly 2017, https://www.oreilly.com/ideas/the-smack-stack, abgerufen am 10.11.2018

[Sto18] Apache Storm: http://storm.apache.org/