Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Anwendungsspezifische proaktive Behandlungen von Prozessen im Bereich Big Data

Lotaire Tchamadeu Tiappi
Angewandte Informatik HAW
Hamburg, Deutschland
Lotaire.tchamadeutiappi@haw-hamburg.de

Lotaire Tchamadeu Tiappi

Grundseminar Master

Wintersemester 2016/2017

Anwendungsspezifische proaktive Behandlung von Prozessen im Bereich Big Data
Im Rahmen des Grundseminras
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Prof Dr. Kai Von Luck und Prof. Dr. Tim.Tiedemann
Abgabe am 31. 07.17

Abstract

Heutzutage sind Daten für kleine, mittelständige und große Unternehmen von Wichtigkeit. Gerade mit der größeren Revolution in der Industrie, wird in der Industrie 4.0 viel mehr Digitalisierung versprochen. Die Digitalisierung an sich selbst garantiert eine Menge Verarbeitung von Daten aus mehreren Quellen. Die Daten werden aus vielfältigen mobilen Endgeräten, GPS, Sensoren...usw. gesammelt, gespeichert und ausgewertet. Viele Open Source Implementierungen wie Hadoop, Spark, Flink...usw. wurden bestimmt, um das Verfahren von Speichern und Zugriff auf massive Daten je nach spezifischer Applikation zu optimieren. Generell im Bereich der Big Data Software Engineering (BDSG) und ins besondere der Datenanalyse taucht immer die Problematik auf, für spezifische Anwendungen proaktive Behandlungen von Prozessen zu definieren. Dieser Artikel befasst sich mit diesem Problem und fasst die aktuelle Lage der Forschung in dem Bereich der Big Data zusammen.

Keywords—Data Analyse; Big Data; Industrie 0.4; Data Persistence ; Data Management

I. EINLEITUNG

Heutzutage sind Daten für kleine, mittelständige und größere Unternehmen von Wichtigkeit. Diese Daten entstehen aus vielfältigen Quellen (Social Network, Smartphone, Tablets, GPS Geräte, Sensoren, log File und vielem mehr) [1]. Die Daten werden massiv gespeichert und bilden, was man heute Big Data nennt. Man unterscheidet generell zu Big Data vielfältige Eigenschaften. Schon McAfee und Brynjofsson hoben im Jahr 2012 hervor, dass die Unterschiede zwischen großen Daten und Analyse eine Frage von Volumen, Geschwindigkeit und Abwechslung sind, da jetzt jede Sekunde mehr Daten das Internet überqueren, als im Internet vor 20 Jahren gespeichert wurde [2]. In den klassischen Varianten werden Daten einfach gespeichert, ohne dass man die Schwierigkeit trifft. Die Variante wird aber im Umgang mit der Big Data schnell seine Grenze zeigen, da es öfter in wenigen Zeiten große Datenmengen speichern soll, die sich öfter in strukturierten und unstrukturierten Formen befinden. Die Frage ist aber, ob eine vernünftige Optimierung von

Datenspeicherung und Datenzugriff durch Auswählen von spezifischen Frameworks oder Open Source je nach spezifischer Anwendung je nach Einsatzbereich keine Lösung wäre, um die heutigen Probleme im Bereich von Big Data zu lösen? In diesem Artikel zeigen wir in der zweiten Sektion das klassische Verfahren von Daten Zugriff/Sammlung, in dem dritten Teil präsentieren wir die Ereignisse für Big Data Engineering, dann stellen wir in dem vierten Teil die Frameworks im Bereich der Big Data vor, im fünften wollen wir ein Beispiel von heutigem Einsatz der Big Data in der Industrie zeigen, die von den Frameworks stark unterstützt werden.

II. KLASSISCHES VERFAHREN VON DATEN ZUGRIFF/ SAMMLUNG

Bei dem klassischen Verfahren waren die Daten in einer definierten Struktur schon festgelegt. Bei Der Bearbeitungsvorgänge sehen die Datenstrukturen gleich aus. Wie es in der folgenden **Abbildung 1** dargestellt ist, sollen die Struktur oder Datenforma im Fall der externen Datenquelle (in Rot markiert) respektiert werden, die für interne RDMS (Relationale Datenbanken Management System) definiert ist. Die wird gemacht, indem die externen Daten Schnittstellen in geforderte Forma angeboten werden. Eine andere Sicht (in Grün markiert) des klassischen Verfahrens besteht darin, die Daten aus der Anwendung aufzurufen, für einen Lösungszweck zu bearbeiten und direkt danach die möglich veränderten und unveränderten Daten wieder zu speichern. Da dieses Verfahren für den heutigen Lösungszweck, aufgrund der heutigen Datenmodelle, die manchmal keine Möglichkeit bieten, dass die Datenformen im Voraus bestimmt werden können, entstehen Überlegungen für flexible Lösungen, die besser geeignet sind.

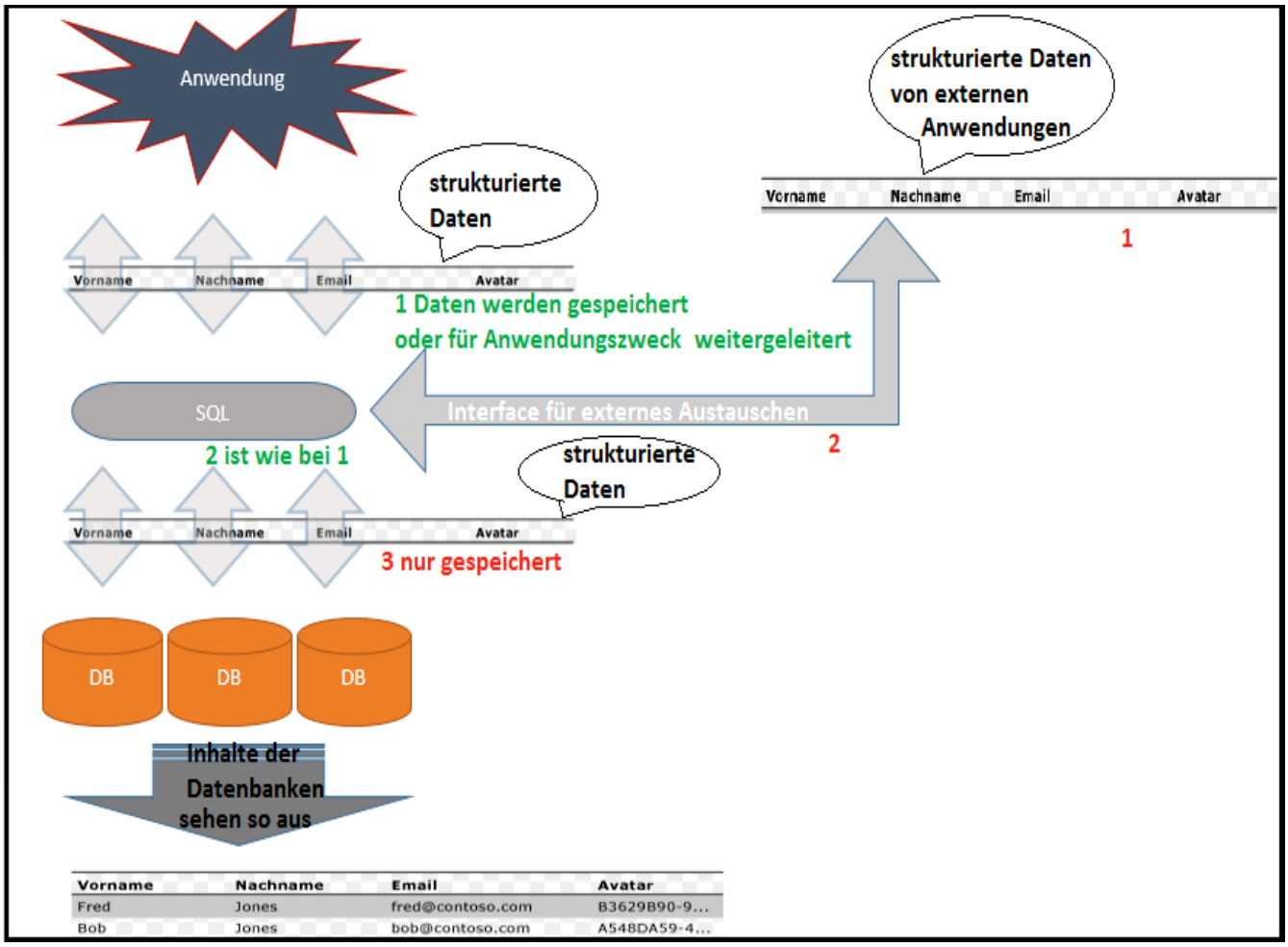

Abbildung 1 klassisches Datenmodellverfahren (eigene Darstellung)

III. Ereignisse für Big Data Engineering

Es wird schon lange Zeit von den Unternehmen verstanden und die Umsetzung der Big Data Operationen in ihrem Geschäft verarbeitet. Heutige Geschäftsprozesse zeigen die Menge von Aktivitäten, die Unternehmen zu schaffen versuchen, um ihren Produkten einen Wert zu verleihen. Big Data im Allgemeinen bezieht sich auf den Satz von drei Merkmalen als Volumen, Vielfalt und Geschwindigkeit von strukturierter, unstrukturierter und semistrukturierter Datenerfassung durch IT-Infrastrukturen und Netzwerken in Speichergeräten und Konversation von den gleichen Daten in nützlichen Informationen für Unternehmen und Gesellschaft. So bezieht sich Volumen auf das große umfangreiche Set von Dateien. Die Größe der Daten ist einfach riesig in Zahlen. Durch die Vielfalt der Daten bleibt immer das Implizieren, dass es Arten und Typen von Daten gibt, durch die Unternehmensziele generiert werden können. Die Geschwindigkeit der Daten impliziert die Sammlungsrate und die Verarbeitungsrate von Daten für verschiedene Anwendungen. Zum Beispiel sollen im Einzelhandel große Datenmengen von Kaufverhalten nahezu in Echtzeit gut analysiert werden. Diese Aufgabe war schon schwer für die alte Technologie. Die Informationen liegen hier in einer nicht

formalisierten Struktur vor und können dadurch vom Computerprogramm über die Schnittstelle keine nicht aggregierten Zugriffe erlauben. Vor allem wie man aus mehreren Anwendungen mit Daten aus mehreren Quellen arbeiten kann und die integrieren kann. Es wird nicht nur schwer Muster in den Daten zu erkennen, sondern auch dieses Verfahren der Mustererkennung schnell zu verarbeiten, ohne dass eine hohe Latenz bei der Anwendung zu merken ist. Deswegen waren solche Fälle schon Ereignisse, die zu Big Datamodell geführt haben. Durch den heutigen Einsatz von strukturierten und unstrukturierten Daten (**Abbildung 2**), heißt es, gibt es die Möglichkeit, dass das Datenmodell von vorne bestimmt oder nicht bestimmt werden kann. Wir haben hier einen freien Spielraum. Das heißt, im Zusammenhang mit der großen Datenverwaltung wird eine Reihe von Fakten zu einem einzigen Fakt. Zum Beispiel lassen sich die Menge von Zahlen, der Mittelwert, Minimum, Maximum und Summe bestimmen [3]. Dadurch dass man heutzutage fast nicht mehr auf die unstrukturierten Daten verzichten kann, konfrontiert man sich mit riesigen Problemen.

Abbildung 2 Unstrukturierte Daten (eigene Darstellung)

Big Data Engineering versucht durch vielfältige Methoden für ständig steigende Daten Lösungen durch Frameworks, wie Hadoop, Spark, Flink oder individuell Open Source wie CEP (Complexe Event Processing) oder auch ML (Machine Learning). In dem nächsten Teil erfahren wir mehr über diese Frameworks und die Unterschiede, die sie voneinander haben.

IV. Frameworks im Bereich Big Data

Seiner Definition nach ist ein Framework ein Programmiergerüst oder ein fertiges Programmteil, generell mit einem sichtbaren Interface für User, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen für den Lösungsteil der Implementierungsphase verwendet wird. Die Methode von Frameworks Verwendung hilft uns schnelle Lösungen für komplexe Probleme zu finden und mit dem Projekt schnell weiter zu kommen.

A. Darstellung der Frameworks im Big Data Bereich

Für die Datenanalyse sind die Frameworks am meisten im Bereich der Big Data unverzichtbar dadurch, dass die schnelle Verarbeitung von größeren Datenmengen gemacht wird, indem man die Mustererkennung im Umgang mit massiv rohen Daten verarbeitet. In der Logistik, Supply Chain Management...usw. werden am meisten folgende Frameworks verwendet; **Apache Hadoop, Apache Spark und Apache Flink.** Also

1) Hadoop

Die Datenmengen wachsen stetig und die Leistung wurde früher durch die CPU gedeckt. Heutzutage sind die Leistungen durch mehr Computer, die parallel arbeiten, unterstützt.

Hadoop ist ein Open Source Java Framework, verteilte Speicher- und Verarbeitungsplattform. Es ist eine Low-cost Hardware, mit einer hohen Zuverlässigkeit durch Fehlertoleranz und hoch skalierend (linear).

Wie auf dem folgenden Bild (**Abbildung 3)** zu sehen ist, verwendet Hadoop-Framework ein verteilt redundantes Speichersystem namens HDFS [4], das Dateien in Blöcken speichert, die in mehreren Rechnern repliziert werden. Ein Hauptserver (Masterknoten) verwaltet die Datenaufteilung und die Replikation in den anderen Chunk-Servern (Arbeiterknoten), die sowohl für die Datenspeicherung als auch für die Verarbeitung verwendet werden.

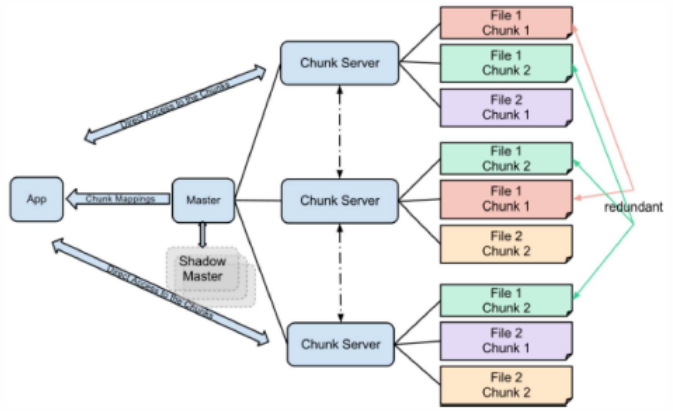

Abbildung 3 Distributed File System architecture [5]

Außerdem verwendet Hadoop hauptsächlich MapReduce als Programmiermodell, um Datensätze zu verarbeiten. MapReduce besteht aus zwei Funktionen: "Map" teilt Probleme in kleinere Teile und "Reduce" kombiniert die Teile wieder zusammen (die Ergebnisse). Die Funktionen Map und Reduce werden von Benutzer geschrieben. MapReduce kümmert sich um alle Details der verteilten Berechnungen. Der Hauptserver (Master-Knoten) wird nicht durch Berechnung überlastet, er ist nur für die Kommunikation mit der Benutzeranwendung und die Verwaltung der anderen Arbeitsknoten verantwortlich. Die Aufgaben werden an Daten gesendet (nicht die an die Arbeitsmaschine gesendeten Daten), die Leistung des Systems verbessern und vor allem die Bandbreite dieses Systems. (Abbildung 4)

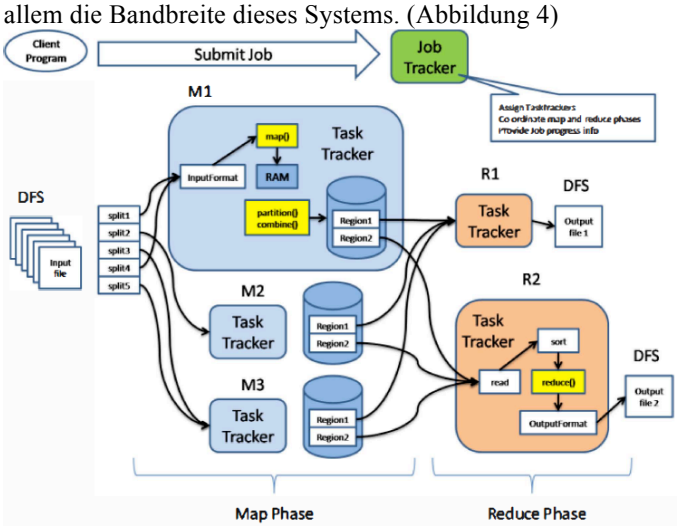

Abbildung 4 MapReduce Architecture[6]

Hadoop eignet sich für die Batch-Analyse, die Verarbeitung auf dem Cluster führt, allerdings brachte sie mehr Starrheit. Also erlaubt es durch seine Map und Reduce Vorgänge keine Echtzeitanalysen, da die Job-Ausgangsdaten zwischen jedem Schritt im verteilten Dateisystem gespeichert werden müssen, bevor der nächste Schritt beginnen kann. Daher ist dieser Ansatz tendenziell langsam aufgrund von Replikation und Festplattenspeicher. Auch Hadoop-Lösungen beinhalten in der Regel Cluster, die schwer zu installieren sind. Es erfordert auch die Integration mehrerer Werkzeuge für verschieden große Daten bei Anwendungsfällen (wie Mahout für maschinelles Lernen und Sturm für Streaming Datenverarbeitung).

2) Apache Spark

Wenn man vorher mit Hadoop etwas Kompliziertes machen wollte, musste man eine Reihe von MapReduce-Jobs zusammenfassen und sie nacheinander ausführen. Jeder dieser Jobs war hoch-Latenz und keiner konnte anfangen, bis der vorherige Job vollständig beendet war. Im Gegensatz dazu betrifft das Spark-Framework Intelligence Data Analytics Aufgaben: Spark ermöglicht es Programmierern komplexe, mehrstufige Datenpipelines mit dem gerichteten azyklischen Graphen (DAG) zu entwickeln. Er erstellt zum Beispiel ein gesteuertes azyklisches Diagramm (DAG), um die Ausführung vor der Planung von Aufgaben, ähnlich wie einen Abfrageausführungsplan, vor der Ausführung von Daten abzurufen. Die Struktur bei Spark läuft wie folgt: Um auf einem Cluster laufen zu können, kann der Spark-Kontext eine Verbindung zu mehreren Typen von Clustermanagern (entweder Sparks eigener Clustermanager, Mesos oder YARN) herstellen, die Ressourcen über Anwendungen hinweg zuordnen. Sobald es angeschlossen ist, erwirbt Spark Verwalter auf Knoten im Cluster, die Prozesse sind, die Berechnungen ausführen und Daten für ihre Anwendung speichern. Als nächstes sendet es ihren Anwendungscode (definiert

durch JAR oder Python-Dateien an SparkContext übergeben) an die Verwalter. Schließlich sendet SparkContext Aufgaben an die ausführenden Verwalter.

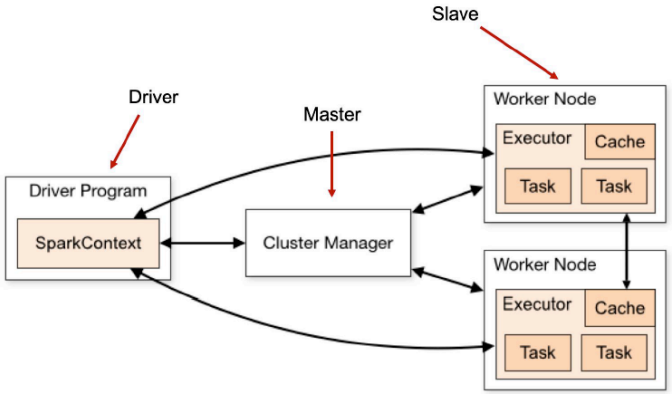

Abbildung 5 Spark-Verfahren [7]

Spark ermöglicht es Programmierern komplexe, mehrstufige Datenpipelines mit dem gerichteten azyklischen Diagramm (DAG) zu entwickeln. Es unterstützt auch in Memory-Daten-Sharing über DAGs, sodass verschiedene Aufträge mit den gleichen Daten arbeiten können. Spark läuft auf dem bestehende Hadoop Distributed File System (HDFS) Infrastruktur, um erweiterte und zusätzliche Funktionalität zu bieten. Es bietet Unterstützung für die Bereitstellung von Spark-Anwendungen in einem bestehenden Hadoop v1-Cluster (mit SMIR-Spark-Inside-MapeReduce) oder Hadoop v2 YARN-Cluster oder sogar Apache Mesos.

1) Apache Flink

Apache Flink ist auch eines der besten Projekte der Apache Foundation. Durch Flink versucht Apache Foundation hier die Lücke in vorherigen Projekten zu verbessern. Der Schwerpunkt von Apache Flink liegt auf der Stream- Programmierung. Die verwendete Abstraktion ist der DataStream, der eine Darstellung eines Streams als einzelnes Objekt darstellt. Operationen sind komponiert (d.h. Pipeline), indem sie Operatoren auf DataStream Gegenstände aufrufen. Flink stellt auch die DataSet-Type für die Batch-Anwendungen zur Verfügung, verkörpert einen einzigen unveränderlichen Multiset -ein Strom von einem Element. Ein Flink-Programm, entweder für Stream- oder Batch-Verarbeitung, ist ein Begriff aus einer Algebra von Operatoren über DataStreams bzw. DataSets.

Würde man die beiden Flink und Spark auf das Level von Dataflow bei paralleler Durchführung der Daten abgleichen, würde man merken, dass Flink den JobGraph (Abb.6) in den ExecutionGraph (Abb. 7) transformiert, indem der JobVertex (ein hierarchischer Ausführer) ein abstrakter Vertex mit einer gewissen Anzahl von ExecutionVertexes (Akteuren) ist und zwar eine je nach paralleler Unteraufgabe.

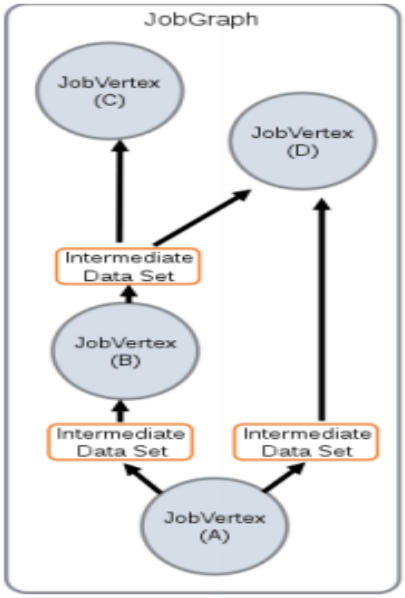

Abbildung 6Flink JobGraph[8]

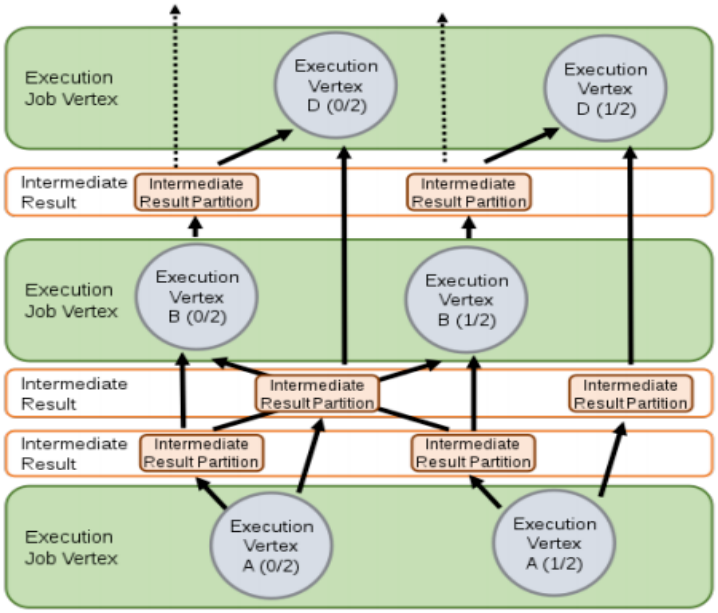

Abbildung 7 Flink Execution Graph [9]

Ein wichtiger Unterschied im Vergleich zum Spark Execution-Graph ist, dass die Abhängigkeit keine Barriere zwischen Akteuren oder hierarchischen Akteuren darstellt: Stattdessen gibt es effektives Symbol Pipelining und Akteure können gleichzeitig durchgeführt werden. Das ist eine natürliche Implementierung für die Stream-Verarbeitung, aber in diesem Fall, da die Laufzeit gleich ist, gilt es für die Batch-Verarbeitung Anwendungen auch. Umgekehrt erfolgt die iterative Verarbeitung nach dem BSP (Bulk Synchronous Parallelism) Ansatz: Eine Auswertung der Schrittfunktion auf allen parallelen Instanzen bildet eine Superstep (wieder ein hierarchischer Akteur), der auch die Granularität der Synchronisation ist; alle parallelen Aufgaben einer Iteration müssen den Superstep vorher abschließen. Der nächste wird initiiert und verhält sich so wie eine Barriere zwischen Iterationen.

Die Stapelverarbeitungen, wie bei Spark für das Daten Streaming haben bisher einen negativen Effekt auf die Latenz, da es immer höher wird. Für die schnelle Datenverarbeitung bringt Flink viel Optimierung, indem die ankommenden Daten direkt verarbeitet werden. Flink unterstützt die Streaming Verarbeitung und Windowing mit Event Time. Dieses Event Time beschreibt Systemunabhängige Zeit (sieht Abbildung 8). Aus dem Grund müssen ankommende Events nicht in der richtigen Reihenfolge verarbeitet werden. Also muss das System nicht auf fehlende Events warten, daher geringe Latenz.

Abbildung 8 Flink Streamingverarbeitung [10]

Apache Hadoop, Flink Spark, Storm... haben alle ein breites Anwendungsfeld und sind für Dutzende von großen Datenszenarien einsetzbar. Die Erweiterung für die SQL-Abfragen (Spark: Spark SQL, Flink: MRQL); Graph Verarbeitung (Spark: GraphX, Flink: Spargel(Basis) und Gelly(Bibliothek)), machine Learning (Spark: MLlib, Flink: Flink ML) und Stream-Verarbeitung (Spark Streaming, Flink Streaming). Beiden sind in der Lage Standalone-Modus im Laufen zu bringen, aber viele verwenden sie auf Hadoop (YARN, HDFS). Sie teilen eine starke Leistung aufgrund der Natur ihrer Speicher. [11]

V. Heutiger Einsatz der Big Data Frameworks

Die Welt sammelt Daten jede Sekunde von jedem Tag und es nimmt nicht ab. Etwa 2,5 Zetta Bytes Daten wurden im Jahr 2012 generiert, und der Trend zeigt, dass die Daten in den folgenden Jahren wachsen werden, da die Anzahl der Unternehmen und Kunden schnell wächst (siehe Abb. 9). In dem heutigen Wettbewerb ist die Wahl von Framework im Bereich der Big Data immer noch von Wichtigkeit, da man aus der Menge von Daten wichtige Informationen ziehen will.

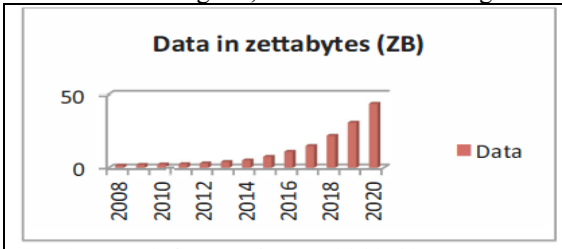

Abbildung 9 Oracle Quelle 2012[12]

Diese Meinung könnte aus folgenden Gründen entstehen.
Zuerst werden Daten kontinuierlich gesammelt. Durch die Technologieorientierten Embedded Geräte wie Autos, Smartphones, RFID-Leser, Webcams und Sensorgeräte entsteht eine riesige Menge von Daten ohne Hilfe von Menschen.
Außerdem sind die Daten von der Natur sehr unterschiedlich. Die großen Datenmengen stammen aus Kamerabild, CCTV, E-commerce-Katalogen usw. Diese Unstrukturierten Datenquellen tragen zu einer viel höheren Vielfalt bei. Die Rolle, die Big Data heutzutage in unserer Gesellschaft spielt, darf nicht unterschätzt werden. Big Data Analyse wird für folgende Zwecke benutzt:

- Echtzeit-Analyse und Vorhersagen über Kundenverhalten und Inventarposition
- Logistik und Supply Chain erhöhen Effizienz bei der Bereitstellung, Beschaffung, Verfolgung von Produkten
- Aggregation von Informationen, um bessere Entscheidungen in der Gesellschaft vorhersagen zu können
- Schnell zunehmende Verbraucherdaten (für Beispiel Mobil) durch Vielfalt von Quellen.

a) Einsatz von Hadoop im Bereich der Logistik

Zuerst wird der Containercode mit Überwachungskameras oder mobilen Geräten erfasst und auf Hadoop verteilte Systemdatei (HDFS) gespeichert. Danach werden auf dem aufgenommenen Bild ein Vorverarbeitungs- und Grau-Farbbildschritt angewendet. Als nächstes zerlegen wir das graue Bild in 20 × 20 Pixelblöcke. Diese Blöcke werden analysiert und klassifiziert auf verschiedenen Maschinen mit MapReduce Programmiermodell, um die Textregionen zu extrahieren. Als nächstes wird ein weiterer Schritt verwendet, um Codezeichen aus den extrahierten Textbereichen zu trennen. Dann wird die optische Zeichenerkennung (OCR) auf einzelne Zeichen mit dem MapReduce-Programmiermodell angewendet. Schließlich wird der Containercode durch Verschmelzung dieser Zeichen erkannt (Abbildung 10).

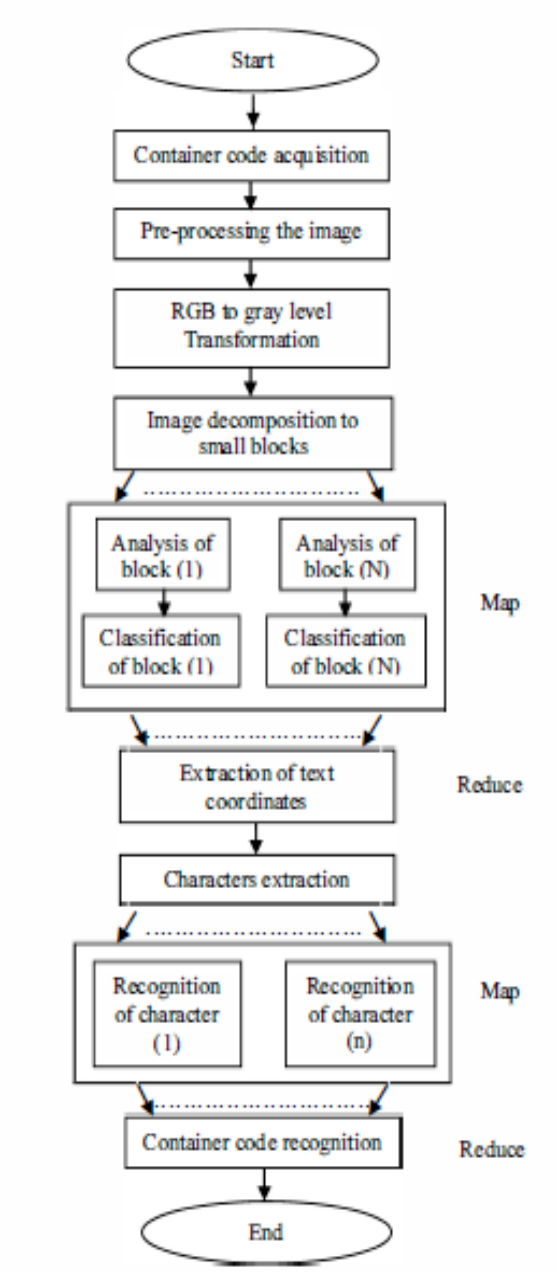

Abbildung 10 Big Data for Containers Code Recognition [13]

a) Einsatz von Spark im Bereich der Supply Chain Management

Das Suppy Chain Management ist einfach das Management von Transport oder Waren- und Dienstleistungensverkehr, es beinhalt auch Lagerung, Haltbarkeit, Analyse von Gütern und Warenverkäufen, Logistik usw. Das Supply Chain Management hilft bei der Planung und Durchführung verschiedener Supply Chain Aktivitäten einer bestimmten Organisation aufzubauen und die aktuelle Marktentwicklung im Zusammenhang mit der Nachfrage und Lieferung von Waren oder Dienstleistungen zu bestimmen und synchronisiert das gleich für die Messung der Leistung der Organisation. Unter Berücksichtigung und unter Betrachtung dieser

Tatsache der virtuelle globale Markt gibt es über eine Milliarde von Bestellungen von etwa Million Kunden jeden Tag Weltweit und damit große Daten entsteht. Für so große Daten gibt es immer eine Frage von Verarbeitung

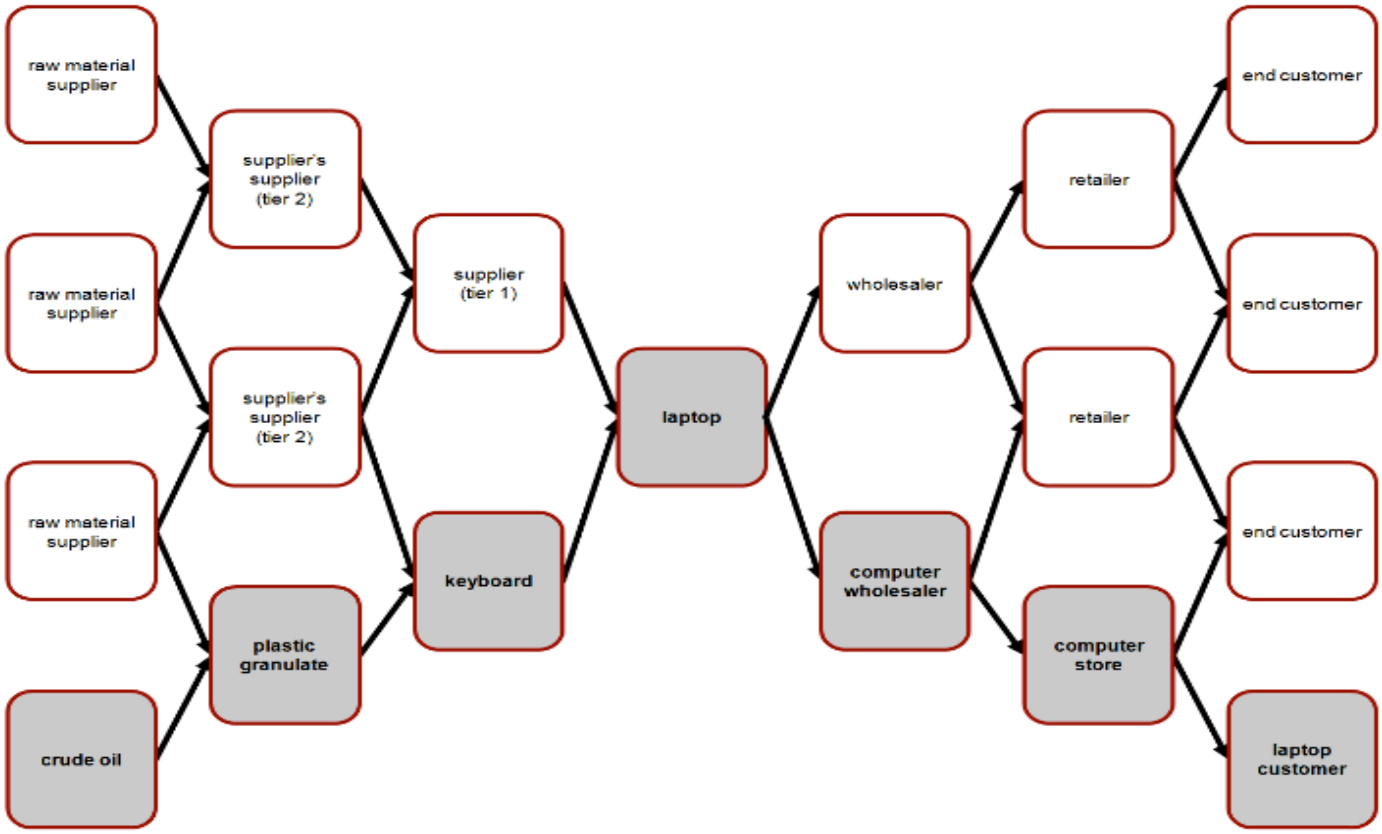

Abbildung 11 Supply Chain Management Managing complex and dynamic supply and demand networks [14]

Hier gibt es große Datenanalysen (sieht Abbildung 11) für Supply Chain Management. Big Data helfen dem Lieferanten beim Sammeln von genauen Information, die ein Maß an Klarheit und Einsicht haben, die nie zuvor erreicht wurden. Analyse helfen bei der Extraktion solcher wertvollen Ressourcen, die zum Austausch einer Information führt und die über die Lieferketten kontextualisiert sind.

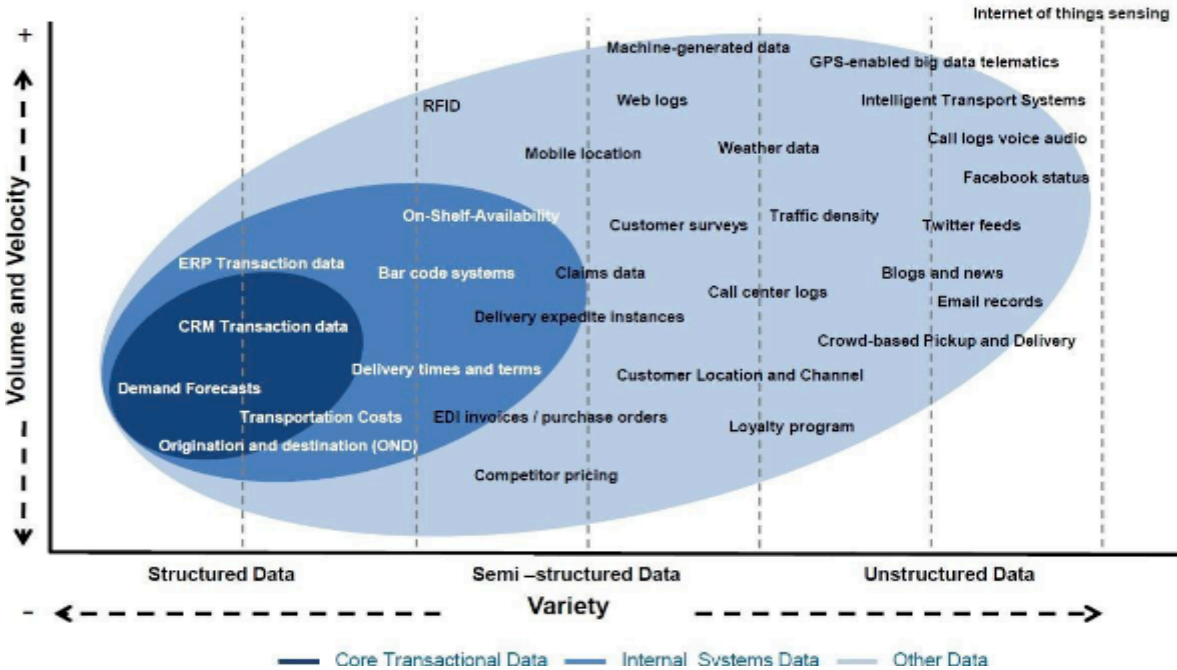

Abbildung 12 SCM Data Volume and Velocity vs Variety [15]

Vieler Hersteller glauben, dass große Daten ein Katalysator für größere Beiträge zur Erhörung der Gewinnspannen sind, da die Mehrheit der Supply Chain außerhalb der Organisation erzeugt wird, dies kann durch die Handlung der Datenquellen nach Vielfalt, Volumen und Geschwindigkeit durch die relative Stufe der strukturierten, semi-strukturierte und unstrukturierte Daten. Die zeigt sich eindeutig der Abbildung 12. Wie bereits erwähnt große Daten im Zusammenhang mit Supply Chain Management besteht aus Informationen wie: Aufträge, Nachfrage und Versorgung Trends, Kunden Dienstleistungen usw., die eine solche Information ansprechen, um mit einem Effiziente Lösung sollte ein schneller Prozess sein, um die Konkurrenz zu schlagen.
Für diesen Fall der Nachfrage für schnellere große Dataanalyse wäre Spark richtig. Es genügt alle Bedürfnisse. Spark hat eine Infrastruktur, die es erlaubt, eine Datenverarbeitungsmaschine zu sein, die in Speicher arbeitet, im RAM; Das heißt, Es gibt kein Lesen und Schreiben mehr, die Daten zurück zu Disk oder Knoten wie bei Hadoop abläuft. Dieses kann der Grund sein, weshalb man sagt, dass der Ablauf bei Spark 100-fache schneller als bei Hadoop[16] ist.

VI. SCHLUSS

In diesem Artikel präsentieren wir das Big Data Software Engineering. Das ständige Steigen von massiven Daten sorgt für Probleme für die Unternehmen, die aus den Daten Informationen bekommen wollen. Wir haben die gerade am meisten angewandten Frameworks dargestellt und haben die kleinen Unterschiede, die in der Streamingverarbeitung zu erkennen sind, aufgezeigt. Hadoop bietet zwar vernünftige Lösungen durch MapReduce Verfahren. Aber der Einsatz zwingt ein Verfahren von Disk IO (input und output on disk). Also verlangt das Verfahren vom Lesen und Schreiben auf der Disk viel Zeit und viele Leistung für alle Cluster. Außerdem haben wir bemerkt, dass Spark durch sein RDDS (**Resilient Distributed Datasets**) Hadoop optimieren kann, indem zum Beispiel RDDs die partitionierten Datenspeicher einliest, die über viele Rechner verteilt sind (typischerweise auf einem Cluster). Spark bietet diese Verbesserung von Hadoop weiter bei der Replikation, Serialization und Optimisation und Lazy Evaluation. Dazu ist noch die starke Verbesserung von Flink bei Spark zu erwähnen, durch effektive Symbole Pipelining und Akteure, die eine wichtige Rolle bei der parallelen Datafolge (Dataflow) spielt. Obwohl Hadoop, Spark und Flink unabhängig laufen können, werden die Big Data Lösungen stärker, wenn diese bei dem Einsatz kombiniert werden.

VII. REFERENZ

[1] Abdelkarim Ben Ayed u.a, " Big Data Analytics for Logistics and Transportation", International Conference on Advanced Logistics and Transportr(IEEE' ICALT 2015), Sp.1

[2] McAfee und Brynjofsson (2012), Big Data: The Management Revolution. In: Harvard business review · October 2012, S. 5

[3] https://de.wikipedia.org/wiki/Unstrukturierte_Daten(19.03.2017)

[4] Abdelkarim Ben Ayed u.a, " Big Data Analytics for Logistics and Transportation", International Conference on Advanced Logistics and Transportr(IEEE' ICALT 2015), Sp.4

[5] Vgl. [4]

[6] https://dzone.com/articles/how-hadoop-mapreduce-works(10.02.2017)

[7] http://spark.apache.org/docs/latest/cluster-overview.html(09:03:2017)

[8] Carbone, P., Fóra, G., Ewen, S., Haridi, S., Tzoumas, K.: Lightweight asynchronous snapshots for distributed dataows. CoRR abs/1506.08603 (2015)

[9] Claudia Misale u.a 2016, "A Comparison of Big Data Frameworks on a Layered Dataflow Model", Proc. Of the 9th Intl Symposium on High-Level Parallel Programming and Applications (HLPP), S.12

[10] https://flink.apache.org/introduction.html

[11] https://www.quora.com/What-is-the-difference-between-Apache-Flink-and-Apache-Spark(05.02.2017)

[12] Kaisler, S Rüstung, F Espinosa, J, A und Geld, W, 2013, Große Daten: Herausforderungen und Fortschritte Vorwärtsbewegung, Systemwissenschaften (HICSS), 2013, 46. Hawaii Internationale Konferenz am 7-10 Jan 2013, 995 – 100

[13] Abdelkarim Ben Ayed u.a, " Big Data Analytics for Logistics and Transportation", International Conference on Advanced Logistics and Transportr(IEEE' ICALT 2015), Sp.5

[14] ,[15] Mr. Harjeet Singh Jaggi u.a, " Integration of Spark framework in Suppy Chain Management", 7th International Conference on Communication, Computing and Virtualization(ICCCV' 2016), S.3

[16] Shyam R u.a, "Apache Spark a Big Data Analytics Platform for Smart Grid", SMART GRID Technologies, August 6-8, 2015 , S.5