Chapter 4
Big Data Acquisition

Klaus Lyko, Marcus Nitzschke, and Axel-Cyrille Ngonga Ngomo

4.1 Introduction

Over the last years, the term big data was used by different major players to label data with different attributes. Moreover, different data processing architectures for big data have been proposed to address the different characteristics of big data. Overall, data acquisition has been understood as the process of gathering, filtering, and cleaning data before the data is put in a data warehouse or any other storage solution.

The position of big data acquisition within the overall big data value chain can be seen in Fig. 4.1. The acquisition of big data is most commonly governed by four of the Vs: volume, velocity, variety, and value. Most data acquisition scenarios assume high-volume, high-velocity, high-variety, but low-value data, making it important to have adaptable and time-efficient gathering, filtering, and cleaning algorithms that ensure that only the high-value fragments of the data are actually processed by the data-warehouse analysis. However, for some organizations, most data is of potentially high value as it can be important to recruit new customers. For such organizations, data analysis, classification, and packaging on very high data volumes play the most central role after the data acquisition.

The goals of this chapter are threefold: First, it aims to identify the present general requirements for data acquisition by presenting open state-of-the-art frameworks and protocols for big data acquisition for companies. Our second goal is then to unveil the current approaches used for data acquisition in the different sectors. Finally, it discusses how the requirements to data acquisition are met by current approaches as well as possible future developments in the same area.

K. Lyko (✉) • M. Nitzschke • A.-C. Ngonga Ngomo
University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany
e-mail: lyko@informatik.uni-leipzig.de; nitzschke@informatik.uni-leipzig.de; ngonga@informatik.uni-leipzig.de

J.M. Cavanillas et al. (eds.), *New Horizons for a Data-Driven Economy*,
DOI 10.1007/978-3-319-21569-3_4

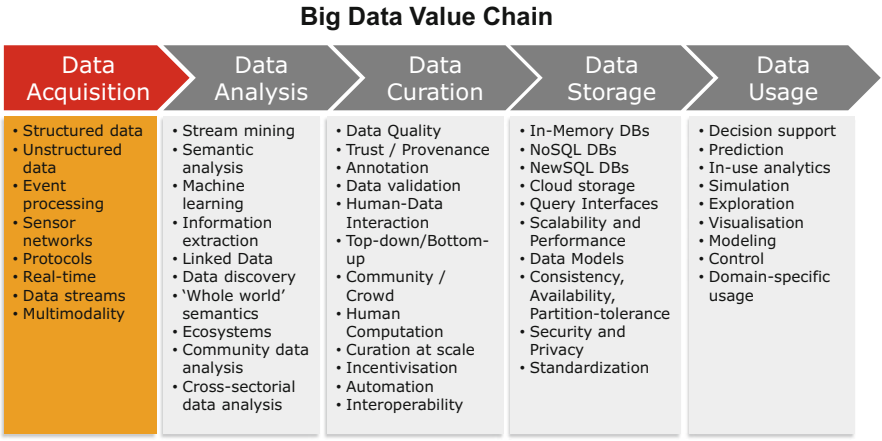

Fig. 4.1 Data acquisition in the big data value chain

4.2 Key Insights for Big Data Acquisition

To get a better understanding of data acquisition, the chapter will first take a look at the different big data architectures of Oracle, Vivisimo, and IBM. This will integrate the process of acquisition within the big data processing pipeline.

The big data processing pipeline has been abstracted in numerous ways in previous works. Oracle (2012) relies on a three-step approach for data processing. In the first step, the content of different data sources is retrieved and stored within a scalable storage solution such as a NoSQL database or the Hadoop Distributed File System (HDFS). The stored data is subsequently processed by first being reorganized and stored in an SQL-capable big data analytics software and finally analysed by using big data analytics algorithms.

Velocity (Vivisimo 2012) relies on a different view on big data. Here, the approach is more search-oriented. The main component of the architecture is a connector layer, in which different data sources can be addressed. The content of these data sources is gathered in parallel, converted, and finally added to an index, which builds the basis for data analytics, business intelligence, and all other data-driven applications. Other big players such as IBM rely on architectures similar to Oracle's (IBM 2013).

Throughout the different architectures to big data processing, the core of data acquisition boils down to gathering data from distributed information sources with the aim of storing them in scalable, big data-capable data storage. To achieve this goal, three main components are required:

1. Protocols that allow the gathering of information for distributed data sources of any type (unstructured, semi-structured, structured)
2. Frameworks with which the data is collected from the distributed sources by using different protocols

3. Technologies that allow the persistent storage of the data retrieved by the frameworks

4.3 Social and Economic Impact of Big Data Acquisition

Over the last years, the sheer amount of data that is produced in a steady manner has increased. Ninety percent of the data in the world today was produced over the last 2 years. The source and nature of this data is diverse. It ranges from data gathered by sensors to data depicting (online) transactions. An ever-increasing part is produced in social media and via mobile devices. The type of data (structured vs. unstructured) and semantics are also diverse. Yet, all this data must be aggregated to help answer business questions and form a broad picture of the market.

For business this trend holds several opportunities and challenges to both creating new business models and improving current operations, thereby generating market advantages. Tools and methods to deal with big data driven by the four Vs can be used for improved user-specific advertisement or market research in general. For example, smart metering systems are tested in the energy sector. Furthermore, in combination with new billing systems these systems could also be beneficial in other sectors such as telecommunication and transport.

Big data has already influenced many businesses and has the potential to impact all business sectors. While there are several technical challenges, the impact on management and decision-making and even company culture will be no less great (McAfee and Brynjolfsson 2012).

There are still several boundaries though. Namely privacy and security concerns need to be addressed by these systems and technologies. Many systems already generate and collect large amounts of data, but only a small fragment is used actively in business processes. In addition, many of these systems lack real-time requirements.

4.4 Big Data Acquisition: State of the Art

The bulk of big data acquisition is carried out within the message queuing paradigm, sometimes also called the streaming paradigm, publish/subscribe paradigm (Carzaniga et al. 2000), or event processing paradigm (Cugola and Margara 2012; Luckham 2002). Here, the basic assumption is that manifold volatile data sources generate information that needs to be captured, stored, and analysed by a big data processing platform. The new information generated by the data source is forwarded to the data storage by means of a data acquisition framework that implements a predefined protocol. This section describes the two core technologies for acquiring big data.

4.4.1 Protocols

Several of the organizations that rely internally on big data processing have devised enterprise-specific protocols of which most have not been publicly released and can thus not be described in this chapter. This section presents the commonly used open protocols for data acquisition.

4.4.1.1 AMQP

The reason for the development of Advanced Message Queuing Protocol (AMQP) was the need for an open protocol that would satisfy the requirements of large companies with respect to data acquisition. To achieve this goal, 23 companies compiled a sequence of requirements for a data acquisition protocol. The resulting AMQP (Advanced Message Queuing Protocol) became an OASIS standard in October 2012. The rationale behind AMQP (Bank of America et al. 2011) was to provide a protocol with the following characteristics:

- **Ubiquity:** This property of AMQP refers to its ability to be used across different industries within both current and future data acquisition architectures. AMQP's ubiquity was achieved by making it easily extensible and simple to implement. The large number of frameworks that implement it, including SwiftMQ, Microsoft Windows Azure Service Bus, Apache Qpid, and Apache ActiveMQ, reflects how easy the protocol is to implement.
- **Safety:** The safety property was implemented across two different dimensions. First, the protocol allows the integration of message encryption to ensure that even intercepted messages cannot be decoded easily. Thus, it can be used to transfer business-critical information. The protocol is robust against the injection of spam, making the AMQP brokers difficult to attack. Second, the AMQP ensures the durability of messages, meaning that it allows messages to be transferred even when the sender and receiver are not online at the same time.
- **Fidelity:** This third characteristic is concerned with the integrity of the message. AMQP includes means to ensure that the sender can express the semantics of the message and thus allow the receiver to understand what it is receiving. The protocol implements reliable failure semantics that allow systems to detect errors from the creation of the message at the sender's end before the storage of the information by the receiver.
- **Applicability:** The intention behind this property is to ensure that AMQP clients and brokers can communicate by using several of the protocols of the Open Systems Interconnection (OSI) model layers such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and also Stream Control Transmission Protocol (SCTP). By these means, AMQP is applicable in many scenarios and industries where not all the protocols of the OSI model layers are required and used. Moreover, the protocol was designed to support different messaging patterns including direct messaging, request/reply, publish/subscribe, etc.

- **Interoperability:** The protocol was designed to be independent of particular implementations and vendors. Thus, clients and brokers with fully independent implementations, architectures, and ownership can interact by means of AMQP. As stated above, several frameworks from different organizations now implement the protocol.
- **Manageability:** One of the main concerns during the specification of the AMQP was to ensure that frameworks that implement it could scale easily. This was achieved by ensuring that AMQP is a fault-tolerant and lossless wire protocol through which information of all types (e.g. XML, audio, video) can be transferred.

To implement these requirements, AMQP relies on a type system and four different layers: a transport layer, a messaging layer, a transaction layer, and a security layer. The type system is based on primitive types from databases (integers, strings, symbols, etc.), described types as known from programming, and descriptor values that can be extended by the users of the protocol. In addition, AMQP allows the use of encoding to store symbols and values as well as the definition of compound types that consist of combinations of several primary types.

The transport layer defines how AMQP messages are to be processed. An AMQP network consists of nodes that are connected via links. Messages can originate from (senders), be forwarded by (relays), or be consumed by nodes (receivers). Messages are only allowed to travel across a link when this link abides by the criteria defined by the source of the message. The transport layer supports several types of route exchanges including message fanout and topic exchange.

The messaging layer of AMQP describes the structure of valid messages. A bare message is a message as submitted by the sender to an AMQP network.

The transaction layer allows for the "coordinated outcome of otherwise independent transfers" (Bank of America et al. 2011, p. 95). The basic idea behind the architecture of the transactional messaging approach followed by the layer lies in the sender of the message acting as controller while the receiver acts as a resource as messages are transferred as specified by the controller. By these means, decentralized and scalable message processing can be achieved.

The final AMQP layer is the security layer, which enables the definition of means to encrypt the content of AMQP messages. The protocols for achieving this goal are supposed to be defined externally from AMQP itself. Protocols that can be used to this end include transport layer security (TSL) and simple authentication and security layer (SASL).

Due to its adoption across several industries and its high flexibility, it is likely that AMQP will become the standard approach for message processing in industries that cannot afford to implement their own dedicated protocols. With the upcoming data-as-a-service industry, it also promises to be the go-to solution for implementing services around data streams. One of the most commonly used AMQP brokers is RabbitMQ, whose popularity is mostly due to the fact that it implements several messaging protocols including JMS.

4.4.1.2 Java Message Service

Java Message Service (JMS) API was included in the Java 2 Enterprise Edition on 18 March 2002, after the Java Community Process in its final version 1.1 ratified it as a standard.

According to the 1.1 specification JMS "provides a common way for Java programs to create, send, receive and read an enterprise messaging system's messages". Administrative tools allow one to bind destinations and connection factories into a Java Naming and Directory Interface (JNDI) namespace. A JMS client can then use resource injection to access the administered objects in the namespace and then establish a logical connection to the same objects through the JMS provider.

The JNDI serves in this case as the moderator between different clients who want to exchange messages. Note that the term "client" is used here (as the spec does) to denote the sender as well as receiver of a message, because JMS was originally designed to exchange message peer-to-peer. Currently, JMS offers two messaging models: point-to-point and publisher-subscriber, where the latter is a one-to-many connection.

AMQP is compatible with JMS, which is the de facto standard for message passing in the Java world. While AMQP is defined at the format level (i.e. byte stream of octets), JMS is standardized at API level and is therefore not easy to implement in other programing languages (as the "J" in "JMS" suggests). Also JMS does not provide functionality for load balancing/fault tolerance, error/advisory notification, administration of services, security, wire protocol, or message type repository (database access).

A considerable advantage of AMQP is, however, the programming language independence of the implementation that avoids vendor-lock in and platform compatibility.

4.4.2 Software Tools

With respect to software tools for data acquisition, many of them are well known and many use cases are available all over the web so it is feasible to have a first approach to them. Despite this, the correct use of each tool requires a deep knowledge on the internal working and the implementation of the software. Different paradigms of data acquisition have appeared depending on the scope these tools have been focused on. The architectural diagram in Fig. 4.2 shows an overall picture of the complete big data workflow highlighting the data acquisition part.

In the remainder of this section, these tools and others relating to data acquisition are described in detail.

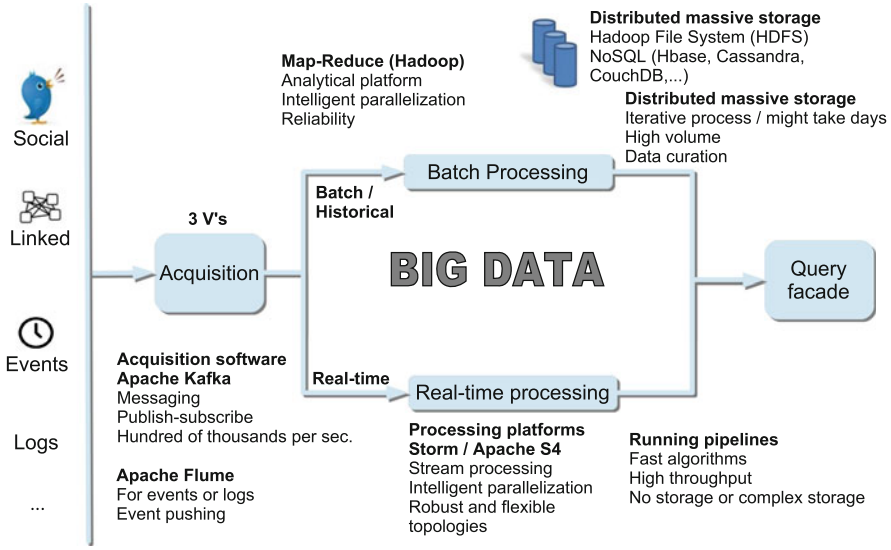

Fig. 4.2 Big data workflow

4.4.2.1 Storm

Storm is an open-source framework for the robust distributed real-time computation on streams of data. It started off as an open-source project and now has a large and active community. Storm supports a wide range of programming languages and storage facilities (relational databases, NoSQL stores, etc.). One of the main advantages of Storm is that it can be utilized in many data gathering scenarios including stream processing and distributed RPC for solving computationally intensive functions on-the-fly, and continuous computation applications (Gabriel 2012). Many companies and applications are using Storm to power a wide variety of production systems processing data, including Groupon, The Weather Channel, fullcontact.com, and Twitter.

The logical network of Storm consists of three types of nodes: a master node called Nimbus, a set of intermediate Zookeeper nodes, and a set of Supervisor nodes.

- **The Nimbus:** is equivalent to Hadoop's JobTracker: it uploads the computation for execution, distributes code across the cluster, and monitors computation.
- **The Zookeepers:** handle the complete cluster coordination. This cluster organization layer is based upon the Apache ZooKeeper project.
- **The Supervisor Daemon:** spawns worker nodes; it is comparable to Hadoop's TaskTracker. This is the place where most of the work of application developers goes into. The worker nodes communicate with the Nimbus via the Zookeepers to determine what to run on the machine, starting and stopping workers.

A computation is called topology in Storm. Once deployed, topologies run indefinitely. There are four concepts and abstraction layers within Storm:

- **Streams:** unbounded sequence of tuples, which are named lists of values. Values can be arbitrary objects implementing a serialization interface.
- **Spouts:** are sources of streams in a computation, e.g. readers for data sources such as the Twitter Streaming APIs.
- **Bolts:** process any number of input streams and produce any number of output streams. This is where most of the application logic goes.
- **Topologies:** are the top-level abstractions of Storm. Basically, a topology is a network of spouts and bolts connected by edges. Every edge is a bolt subscribing to the stream of a spout or another bolt.

Both spouts and bolts are stateless nodes and inherently parallel, executing as many tasks across the cluster. From a physical point of view a worker is a Java Virtual Machine (JVM) process with a number of tasks running within. Both spouts and bolts are distributed over a number of tasks and workers. Storm supports a number of stream grouping approaches ranging from random grouping to tasks, to field grouping, where tuples are grouped by specific fields to the same tasks (Madsen 2012).

Storm uses a pull model; each bolt pulls events from its source. Tuples traverse the entire network within a specified time window or are considered as failed. Therefore, in terms of recovery the spouts are responsible to keep tuples ready for replay.

4.4.2.2 S4

S4 (simply scalable streaming system) is a distributed, general-purpose platform for developing applications that process streams of data. Started in 2008 by Yahoo! Inc., since 2011 it is an Apache Incubator project. S4 is designed to work on commodity hardware, avoiding I/O bottlenecks by relying on an all-in-memory approach (Neumeyer 2011).

In general keyed data events are routed to processing elements (PE). PEs receive events and either emit resulting events and/or publish results. The S4 engine was inspired by the MapReduce model and resembles the Actors model (encapsulation semantics and location transparency). Among others it provides a simple programming interface for processing data streams in a decentralized, symmetric, and pluggable architecture.

A stream in S4 is a sequence of elements (events) of both tuple-valued keys and attributes. A basic computational unit PE is identified by the following four components: (1) its functionality provided by the PE class and associated configuration, (2) the event types it consumes, (3) the keyed attribute in this event, and (4) the value of the keyed attribute of the consuming events. A PE is instantiated by the platform for each value of the key attribute. Keyless PEs are a special class of PEs with no keyed attribute and value. These PEs consume all events of the corresponding type and are typically at the input layer of an S4 cluster. There is a large number of standard PEs available for a number of typical tasks such as

aggregate and join. The logical hosts of PEs are the processing nodes (PNs). PNs listen to events, execute operations for incoming events, and dispatch events with the assistance of the communication layer.

S4 routes each event to PNs based on a hash function over all known values of the keyed attribute in the event. There is another special type of PE object: the PE prototype. It is identified by the first three components. These objects are configured upon initialization and for any value it can clone itself to create a fully qualified PE. This cloning event is triggered by the PN for each unique value of the keyed attribute. An S4 application is a graph composed of PE prototypes and streams that produce, consume, and transmit messages, whereas PE instances are clones of the corresponding prototypes containing the state and are associated with unique keys (Neumeyer et al. 2011).

As a consequence of this design S4 guarantees that all events with a specific value of the keyed attribute arrive at the corresponding PN and within it are routed to the specific PE instance (Bradic 2011). The current state of a PE is inaccessible to other PEs. S4 is based upon a push model: events are routed to the next PE as fast as possible. Therefore, if a receiver buffer fills up events may be dropped. Via lossy checkpointing S4 provides state recovery. In the case of a node crash a new one takes over its task from the most recent snapshot. The communication layer is based upon the Apache ZooKeeper project. It manages the cluster and provides failover handling to stand-by nodes. PEs are built in Java using a fairly simple API and are assembled into the application using the Spring framework.

4.4.2.3 Kafka

Kafka is a distributed publish-subscribe messaging system designed to support mainly persistent messaging with high-throughput. Kafka aims to unify offline and online processing by providing a mechanism for a parallel load into Hadoop as well as the ability to partition real-time consumption over a cluster of machines. The use for activity stream processing makes Kafka comparable to Apache Flume, though the architecture and primitives are very different and make Kafka more comparable to a traditional messaging system.

Kafka was originally developed at LinkedIn for tracking the huge volume of activity events generated by the website. These activity events are critical for monitoring user engagement as well as improving relevancy in their data-driven products. The previous diagram gives a simplified view of the deployment topology at LinkedIn.

Note that a single Kafka cluster handles all activity data from different sources. This provides a single pipeline of data for both online and offline consumers. This tier acts as a buffer between live activity and asynchronous processing. Kafka can also be used to replicate all data to a different data centre for offline consumption.

Kafka can be used to feed Hadoop for offline analytics, as well as a way to track internal operational metrics that feed graphs in real time. In this context, a very appropriate use for Kafka and its publish-subscribe mechanism would be

processing related stream data, from tracking user actions on large-scale websites to relevance and ranking tasks.

In Kafka, each stream is called a “topic”. Topics are partitioned for scaling purposes. Producers of messages provide a key which is used to determine the partition the message is sent to. Thus, all messages partitioned by the same key are guaranteed to be in the same topic partition. Kafka brokers handle some partitions and receive and store messages sent by producers.

Kafka consumers read from a topic by getting messages from all partitions of the topic. If a consumer wants to read all messages with a specific key (e.g. a user ID in case of website clicks) he only has to read messages from the partition the key is on, not the complete topic. Furthermore, it is possible to reference any point in a brokers log file using an offset. This offset determines where a consumer is in a specific topic/partition pair. The offset is incremented once a consumer reads the topic/partition pair.

Kafka provides an at-least-once messaging guarantee and highly available partitions. To store and cache messages Kafka relies on file systems, whereas all data is written immediately to a persistent log without necessarily flushing to disk. In combination the protocol is built upon a message set abstraction, which groups messages together. Therewith, it minimizes the network overhead and sequential disk operations. Both consumer and producer share the same message format.

4.4.2.4 Flume

Flume is a service for efficiently collecting and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tuneable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows online analytic applications. The system was designed with these four key goals in mind: reliability, scalability, manageability, and extensibility

The purpose of Flume is to provide a distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of log data from many different sources to a centralized data store. The architecture of Flume NG is based on a few concepts that together help achieve this objective:

- **Event:** a byte payload with optional string headers that represent the unit of data that Flume can transport from its point of origin to its final destination.
- **Flow:** movement of events from the point of origin to their final destination is considered a data flow, or simply flow.
- **Client:** an interface implementation that operates at the point of origin of events and delivers them to a Flume agent.
- **Agent:** an independent process that hosts flume components such as sources, channels, and sinks, and thus has the ability to receive, store, and forward events to their next-hop destination.

- **Source:** an interface implementation that can consume events delivered to it via a specific mechanism.
- **Channel:** a transient store for events, where events are delivered to the channel via sources operating within the agent. An event put in a channel stays in that channel until a sink removes it for further transport.
- **Sink:** an interface implementation that can remove events from a channel and transmit them to the next agent in the flow, or to the event's final destination.

These concepts help in simplifying the architecture, implementation, configuration, and deployment of Flume.

A flow in Flume NG starts from the client. The client transmits the event to its next-hop destination. This destination is an agent. More precisely, the destination is a source operating within the agent. The source receiving this event will then deliver it to one or more channels. The channels that receive the event are drained by one or more sinks operating within the same agent. If the sink is a regular sink, it will forward the event to its next-hop destination, which will be another agent. If instead it is a terminal sink, it will forward the event to its final destination. Channels allow for the decoupling of sources from sinks using the familiar producer-consumer model of data exchange. This allows sources and sinks to have different performance and runtime characteristics and yet be able to effectively use the physical resources available to the system.

The primary use case for Flume is as a logging system that gathers a set of log files on every machine in a cluster and aggregates them to a centralized persistent store such as the Hadoop Distributed File System (HDFS). Also, Flume can be used as an HTTP event manager that deals with different types of requests and drives each of them to any specific data store during a data acquisition process, such as an NoSQL databases like HBase.

Therefore, Apache Flume is not a pure data acquisition system but acts in a complementary fashion by managing the different data types acquired and transforming them to specific data stores or repositories.

4.4.2.5 Hadoop

Apache Hadoop is an open-source project developing a framework for reliable, scalable, and distributed computing on big data using clusters of commodity hardware. It was derived from Google's MapReduce and the Google File System (GFS) and written in JAVA. It is used and supported by a large community and is both used in production and research environments by many organizations, most notably: Facebook, a9.com, AOL, Baidu, IBM, Imageshack, and Yahoo. The Hadoop project consists of four modules:

- **Hadoop Common:** for common utilities used throughout Hadoop.
- **Hadoop Distributed File System (HDFS):** a highly available and efficient file system.

- **Hadoop YARN (Yet Another Resource Negotiator):** a framework for job scheduling and cluster management.
- **Hadoop MapReduce:** a system to parallel processing large amounts of data.

A Hadoop cluster is designed according to the master-slave principle. The master is the name node. It keeps track of the metadata about the file distribution. Large files are typically split into chunks of 128 MB. These parts are copied three times and the replicas are distributed through the cluster of data nodes (slave nodes). In the case of a node failure its information is not lost; the name node is able to allocate the data again. To monitor the cluster every slave node regularly sends a heartbeat to the name node. If a slave is not recognized over a specific period it is considered dead. As the master node is a single point of failure it is typically run on highly reliable hardware. And, as precaution a secondary name node can keep track of changes in the metadata; with its help it is possible to rebuild the functionality of the name node and thereby ensure the functionality of the cluster.

YARN is Hadoop's cluster scheduler. It allocates a number of containers (which are essential processes) in a cluster of machines and executes arbitrary commands on them. YARN consists of three main pieces: a ResourceManager, a NodeManager, and an ApplicationMaster. In a cluster each machine runs a NodeManager, responsible for running processes on the local machine. Resource-Managers tell NodeManagers what to run, and Applications tell the ResourceManager when to run something on the cluster.

Data is processed according to the MapReduce paradigm. MapReduce is a framework for parallel-distributed computation. As data storage processing works in a master-slave fashion, computation tasks are called jobs and are distributed by the job tracker. Instead of moving the data to the calculation, Hadoop moves the calculation to the data. The job tracker functions as a master distributing and administering jobs in the cluster. Task trackers carry out the actual work on jobs. Typically each cluster node is running a task tracker instance and a data node. The MapReduce framework eases programming of highly distributed parallel programs. A programmer can focus on writing the more simpler map() and reduce() functions dealing with the task at hand while the MapReduce infrastructure takes care of running and managing the tasks in the cluster.

In the orbit of the Hadoop project a number of related projects have emerged. The Apache Pig project for instance is built upon Hadoop and simplifies writing and maintaining Hadoop implementations. Hadoop is very efficient for batch processing. The Apache HBase project aims to provide real-time access to big data.

4.5 Future Requirements and Emerging Trends for Big Data Acquisition

Big data acquisition tooling has to deal with high-velocity, variety, and real-time data acquisition. Thus, tooling for data acquisition has to ensure a very high throughput. This means that data can come from multiple resources (social networks, sensors, web mining, logs, etc.) with different structures, or be unstructured (text, video, pictures, and media files) and at a very high pace (tens or hundreds of thousands events per second). Therefore, the main challenge in acquiring big data is to provide frameworks and tools that ensure the required throughput for the problem at hand without losing any data in the process.

In this context, emerging challenges for the acquisition of big data include the following:

- Data acquisition is often started by tools that provide some kind of input data to the system, such as social networks and web mining algorithms, sensor data acquisition software, logs periodically injected, etc. Typically the data acquisition process starts with single or multiple end points where the data comes from. These end points could take different technical appearances, such as log importers, Storm-based algorithms, or even the data acquisition may offer APIs to the external world to inject the data, by using RESTful services or any other programmatic APIs. Hence, any technical solution that aims to acquire data from different sources should be able to deal with this wide range of different implementations.
- To provide the mechanisms to connect the data acquisition with the data pre- and post-processing (analysis) and storage, both in the historical and real-time layers. In order to do so, the batch and real-time processing tools (i.e. Storm and Hadoop) should be able to be contacted by the data acquisition tools. This is implemented in different ways. For instance Apache Kafka uses a publish-subscribe mechanism where both Hadoop and Storm can be subscribed, and therefore the messages received will be available to them. Apache Flume on the other hand follows a different approach, storing the data in a NoSQL key-value store to ensure velocity, and pushing the data to one or several receivers (i.e. Hadoop and Storm). There is a red thin line between data acquisition, storage, and analysis in this process, as data acquisition typically ends by storing the raw data in an appropriate master dataset, and connecting with the analytical pipeline (especially for real-time, but also batch processing).
- To come up with a structured or semi-structured model valid for data analysis, to effectively pre-process acquired data, especially unstructured data. The borders between data acquisition and analysis are blurred in the pre-processing stage. Some may argue that pre-processing is part of processing, and therefore of data analysis, while others believe that data acquisition does not end with the actual gathering, but also with cleaning the data and providing a minimal set of coherence and metadata on top of it. Data cleaning usually takes several steps,

such as boilerplate removal (i.e. removing HTML headers in web mining acquisition), language detection and named entities recognition (for textual resources), and providing extra metadata such as timestamp, provenance information (yet another overlap with data curation), etc.
- The acquisition of media (pictures, video) is a significant challenge, but it is an even bigger challenge to perform the analysis and storage of video and images.
- Data variety requires processing the semantics in the data in order to correctly and effectively merge data from different sources while processing. Works on semantic event processing such as semantic approximations (Hasan and Curry 2014a), thematic event processing (Hasan and Curry 2014b), and thingsonomy tagging (Hasan and Curry 2015) are emerging approaches in this area, within this context.
- In order to perform post- and pre-processing of acquired data, the current state-of the art provides a set of open-source and commercial tools and frameworks. The main goal when defining a correct data acquisition strategy is therefore to understand the needs of the system in terms of data volume, variety, and velocity, and take the right decision on which tool is best to ensure the acquisition and desired throughput.

4.6 Sector Case Studies for Big Data Acquisition

This section analyses the use of big data acquisition technology within a number of sectors.

4.6.1 Health Sector

Within the health sector big data technology aims to establish a holistic approach whereby clinical, financial, and administrative data as well as patient behavioural data, population data, medical device data, and any other related health data are combined and used for retrospective, real-time, and predictive analysis.

In order to establish a basis for the successful implementation of big data health applications, the challenge of data digitalization and acquisition (i.e. putting health data in a form suitable as input for analytic solutions) needs to be addressed.

As of today, large amounts of health data are stored in data silos and data exchange is only possible via Scan, Fax, or email. Due to inflexible interfaces and missing standards, the aggregation of health data relies on individualized solutions with high costs.

In hospitals patient data is stored in CIS (clinical information system) or EHR (electronic health record) systems. However, different clinical departments might use different systems, such as RIS (radiology information system), LIS (laboratory information system), or PACS (picture archiving and communication system) to

store their data. There is no standard data model or EHR system. Existing mechanisms for data integration are either adaptations of standard data warehouse solutions from horizontal IT providers like Oracle Healthcare Data Model, Teradata's Healthcare Logical Data Model, IBM Healthcare Provider Data Model, or new solutions like the i2b2 platform. While the first three are mainly used to generate benchmarks regarding the performance of the overall hospital organization, the i2b2 platform establishes a data warehouse that allows the integration of data from different clinical departments in order to support the task of identifying patient cohorts. In doing so, structured data such as diagnoses and lab values are mapped to standardized coding systems. However, unstructured data is not further labelled with semantic information. Besides its main functionality of patient cohorts identification, the i2b2 hive offers several additional modules. Besides specific modules for data import, export, and visualization tasks, modules to create and use additional semantics are available. For example, the natural language processing (NLP) tool offers a means to extract concepts out of specific terms and connect them with structured knowledge.

Today, data can be exchanged by using exchange formats such as HL7. However, due to non-technical reasons such as privacy, health data is commonly not shared across organizations (phenomena of organizational silos). Information about diagnoses, procedures, lab values, demographics, medication, provider, etc., is in general provided in a structured format, but not automatically collected in a standardized manner. For example, lab departments use their own coding system for lab values without an explicit mapping to the LOINC (Logical Observation Identifiers Names and Codes) standard. Also, different clinical departments often use different but customized report templates without specifying the common semantics. Both scenarios lead to difficulties in data acquisition and consequent integration.

Regarding unstructured data like texts and images, standards for describing high-level meta-information are only partially collected. In the imaging domain, the DICOM (Digital Imaging and Communications in Medicine) standard for specifying image metadata is available. However, for describing meta-information of clinical reports or clinical studies a common (agreed) standard is missing. To the best of our knowledge, for the representation of the content information of unstructured data like images, texts, or genomics data, no standard is available. Initial efforts to change this situation are initiatives such as the structured reporting initiative by RSNA or semantic annotations using standardized vocabularies. For example, the Medical Subject Headings (MeSH) is a controlled vocabulary thesaurus of the US National Library of Medicine to capture topics of texts in the medical and biological domain. There also exist several translations to other languages.

Since each EHR vendor provides their own data model, there is no standard data model for the usage of coding systems to represent the content of clinical reports. In terms of the underlying means for data representation, existing EHR systems rely on a case-centric rather than on a patient-centric representation of health data. This hinders longitudinal health data acquisition and integration.

Easy to use structured reporting tools are required which do not create extra work for clinicians, i.e. these systems need to be seamlessly integrated into the clinical workflow. In addition, available context information should be used to assist the clinicians. Given that structured reporting tools are implemented as easy-to-use tools, they can gain acceptance by clinicians such that most of the clinical documentation is carried out in a semi-structured form and the quality and quantity of semantic annotations increases.

From an organizational point of view, the storage, processing, access, and protection of big data has to be regulated on several different levels: institutional, regional, national, and international level. There is a need to define who authorizes which processes, who changes processes, and who implements process changes. Therefore, a proper and consistent legal framework or guidelines [e.g. ISO/IEC 27000] for all four levels are required.

IHE (integrating the healthcare enterprise) enables plug-and-play and secure access to health information whenever and wherever it is needed. It provides different specifications, tools, and services. IHE also promotes the use of well-established and internationally accepted standards (e.g. Digital Imaging and Communications in Medicine, Health Level 7). Pharmaceutical and R&D data that encompass clinical trials, clinical studies, population and disease data, etc. is typically owned by the pharmaceutical companies, research labs/academia, or the government. As of today, a lot of manual effort is taken to collect all the datasets for conducting clinical studies and related analysis. The manual effort for collecting the data is quite high.

4.6.2 Manufacturing, Retail, and Transport

Big data acquisition in the context of the retail, transportation, and manufacturing sectors becomes increasingly important. As data processing costs decrease and storage capacities increase, data can now be continuously gathered. Manufacturing companies as well as retailers may monitor channels like Facebook, Twitter, or news for any mentions and analyse these data (e.g. customer sentiment analysis). Retailers on the web are also collecting large amounts of data by storing log files and combining that information with other data sources such as sales data in order to analyse and predict customer behaviour. In the field of manufacturing, all participating devices are nowadays interconnected (e.g. sensors, RFID), such that vital information is constantly gathered in order to predict defective parts at an early stage.

All three sectors have in common that the data comes from very heterogeneous sources (e.g. log files, data from social media that needs to be extracted via proprietary APIs, data from sensors, etc.). Data comes in at a very high pace, requiring that the right technologies be chosen for extraction (e.g. MapReduce). Challenges may also include data integration. For example, product names used by customers on social media platforms need to be matched against IDs used for

product pages on the web and then matched against internal IDs used in Enterprise Resource Planning (ERP) systems. Tools used for data acquisition in retail can be grouped by the two types of data typically collected in retail:

- Sales data from accounting and controlling departments
- Data from the marketing departments

The dynamite data channel monitor, recently bought by Market Track LLC, provides a solution to gather information about product prices on more than 1 billion "buy" pages at more than 4000 global retailers in real time, and thus allows to study the impact of promotional investments, monitor prices, and track consumer sentiment on brands and products.

The increasing use of social media not only empowers consumers to easily compare services and products both with respect to price and quality, but also enables retailers to collect, manage, and analyse large volumes and velocity of data, providing a great opportunity for the retail industry. To gain competitive advantages, real-time information is essential for accurate prediction and optimization models. From a data acquisition perspective means for stream data computation are necessary, which can deal with the challenges of the Vs of the data.

In order to bring a benefit for the transportation sector (especially multimodal urban transportation), tools that support big data acquisition have to achieve mainly two tasks (DHL 2013; Davenport 2013). First, they have to handle large amounts of personalized data (e.g. location information) and deal with the associated privacy issues. Second, they have to integrate data from different service providers, including geographically distributed sensors (i.e. Internet of Things (IoT)) and open data sources.

Different players benefit from big data in the transport sector. Governments and public institutions use an increasing amount of data for traffic control, route planning, and transport management. The private sector exploits increasing amounts of date for route planning and revenue management to gain competitive advantages, save time, and increase fuel efficiency. Individuals increasingly use data via websites, mobile device applications, and GPS information for route planning to increase efficiency and save travel time.

In the manufacturing sector, tools for data acquisition need to mainly process large amounts of sensor data. Those tools need to handle sensor data that may be incompatible with other sensor data and thus data integration challenges need to be tackled, especially when sensor data is passed through multiple companies in a value chain.

Another category of tools needs to address the issue of integrating data produced by sensors in a production environment with data from, e.g. ERP systems within enterprises. This is best achieved when tools produce and consume standardized metadata formats.

4.6.3 Government, Public, Non-profit

Integrating and analysing large amounts of data play an increasingly important role in today's society. Often, however, new discoveries and insights can only be attained by integrating information from dispersed sources. Despite recent advances in structured data publishing on the web (such as using RDF in attributes (RDFa) and the schema.org initiative), the question arises how larger datasets can be published in a manner that makes them easily discoverable and facilitates integration as well as analysis.

One approach for addressing this problem is data portals, which enable organizations to upload and describe datasets using comprehensive metadata schemes. Similar to digital libraries, networks of such data portals can support the description, archiving, and discovery of datasets on the web. Recently, a rapid growth has been seen of data catalogues being made available on the web. The data catalogue registry datacatalogs.org lists 314 data catalogues worldwide. Examples for the increasing popularity of data catalogues are Open Government Data portals, data portals of international organizations and NGOs, as well as scientific data portals. In the public and governmental sector a few catalogues and data hubs can be used to find metadata or at least to find locations (links) to interesting media files such as publicdata.eu.

The public sector is centred around the activities of the citizens. Data acquisition in the public sector includes tax collection, crime statistics, water and air pollution data, weather reports, energy consumption, Internet business regulation: online gaming, online casinos, intellectual property protection, and others.

The open data initiatives of the governments (data.gov, data.gov.uk for open public data, or govdata.de) are recent examples of the increasing importance of public and non-profit data. There exist similar initiatives in many countries. Most data collected by public institutions and governments of these countries is in principle available for reuse. The W3C guidance on opening up government data (Bennett and Harvey 2009) suggests that data should be published as soon as available in the original raw format, then to enhance it with semantics and metadata. However, in many cases governments struggle to publish certain data, due to the fact that the data needs to be strictly non-personal and non-sensitive and compliant with data privacy and protection regulations. Many different sectors and players can benefit from this public data.

The following presents several case studies for implementing big data technologies in different areas of the public sector.

4.6.3.1 Tax Collection Area

One key area for big data solutions is for the tax revenue recovery of millions of dollars per year. The challenge for such an application is to develop a fast, accurate identity resolution and matching capability for a budget-constrained, limited-

staffed state tax department in order to determine where to deploy scarce auditing resources and enhance tax collection efficiency. The main implementation highlights are:

- Rapidly identify exact and close matches
- Enable de-duplication from data entry errors
- High throughput and scalability handles growing data volumes
- Quickly and easily accommodate file format changes, and addition of new data sources

One solution is based on software developed by the Pervasive Software company: the Pervasive DataRush engine, the Pervasive DataMatcher, and the Pervasive Data Integrator. Pervasive DataRush provides simple constructs to:

- Create units of work (processes) that can each individually be made parallel.
- Tie processes together in a dataflow graph (assemblies), but then enable the reuse of complex assemblies as simple operators in other applications.
- Further tie operators into new, broader dataflow applications.
- Run a compiler that can traverse all sub-assemblies while executing customizers to automatically define parallel execution strategies based on then-current resources and/or more complex heuristics (this will only improve over time).

This is achieved using techniques such as fuzzy matching, record linking, and the ability to match any combination of fields in a dataset. Other key techniques include data integration and Extract, Transform, Load (ETL) processes that save and store all design metadata in an open XML-based design repository for easy metadata interchange and reuse. This enables fast implementation and deployment and reduces the cost of the entire integration process.

4.6.3.2 Energy Consumption

An article reports on the problems in the regulation of energy consumption. The main issue is that when energy is put on the distribution network it must be used at that time. Energy providers are experimenting with storage devices to assist with this problem, but they are nascent and expensive. Therefore the problem is tackled with smart metering devices.

When collecting data from smart metering devices, the first challenge is to store the large volume of data. For example, assuming that 1 million collection devices retrieve 5 kB of data per single collection, the potential data volume growth in a year can be up to 2920 TB.

The consequential challenges are to analyse this huge volume of data, cross-reference that data with customer information, network distribution, and capacity information by segment, local weather information, and energy spot market cost data.

Harnessing this data will allow the utilities to better understand the cost structure and strategic options within their network, which could include:

- Adding generation capacity versus purchasing energy off the spot market (e.g. renewables such as wind, solar, electric cars during off-peak hours)
- Investing in energy storage devices within the network to offset peak usage and reduce spot purchases and costs
- Provide incentives to individual consumers, or groups of consumers, to change energy consumption behaviours

One such approach from the Lavastorm company is a project that explores analytics problems with innovative companies such as FalbygdensEnergi AB (FEAB) and Sweco. To answer key questions, the Lavastorm Analytic Platform is utilized. The Lavastorm Analytics Engine is a self-service business analytics solution that empowers analysts to rapidly acquire, transform, analyse, and visualize data, and share key insights and trusted answers to business questions with non-technical managers and executives. The engine offers an integrated set of analytics capabilities that enables analysts to independently explore enterprise data from multiple data sources, create and share trusted analytic models, produce accurate forecasts, and uncover previously hidden insights in a single, highly visual and scalable environment.

4.6.4 Media and Entertainment

Media and entertainment is centred on knowledge included in the media files. With the significant growth of media files and associated metadata, due to evolution of the Internet and the social web, data acquisition in this sector has become a substantial challenge.

According to a Quantum report, managing and sharing content can be a challenge, especially for media and entertainment industries. With the need to access video footage, audio files, high-resolution images, and other content, a reliable and effective data sharing solution is required.

Commonly used tools in the media and entertainment sector include:

- Specialized file systems that are used as a high-performance alternative to NAS and network shares
- Specialized archiving technologies that allow the creation of a digital archive that reduces costs and protects content
- Specialized clients that enable both LAN-based applications and SAN-based applications to share a single content pool
- Various specialized storage solutions (for high-performance file sharing, cost-effective near-line storage, offline data retention, for high-speed primary storage)

Digital on-demand services have radically changed the importance of schedules for both consumers and broadcasters. The largest media corporations have already invested heavily in the technical infrastructure to support the storage and streaming

of content. For example, the number of legal music download and streaming sites, and Internet radio services, has increased rapidly in the last few years—consumers have an almost-bewildering choice of options depending on what music genres, subscription options, devices, Digital rights management (DRM) they like. Over 391 million tracks were sold in Europe in 2012, and 75 million tracks played on online radio stations.

According to Eurostat, there has been a massive increase in household access to broadband in the years since 2006. Across the "EU27" (EU member states and six other countries in the European geographical area) broadband penetration was at around 30 % in 2006 but stood at 72 % in 2012. For households with high-speed broadband, media streaming is a very attractive way of consuming content. Equally, faster upload speeds mean that people can create their own videos for social media platforms.

There has been a huge shift away from mass, anonymized mainstream media, towards on-demand, personalized experiences. Large-scale shared consumer experiences such as major sporting events, reality shows, and soap operas are now popular. Consumers expect to be able to watch or listen to whatever they want, whenever they want.

Streaming services put control in the hands of users who choose when to consume their favourite shows, web content, or music. The largest media corporations have already invested heavily in the technical infrastructure to support the storage and streaming of content.

Media companies hold significant amounts of personal data, whether on customers, suppliers, content, or their own employees. Companies have responsibility not just for themselves as data controllers, but also their cloud service providers (data processors). Many large and small media organizations have already suffered catastrophic data breaches—two of the most high-profile casualties were Sony and LinkedIn. They incurred not only the costs of fixing their data breaches, but also fines from data protection bodies such as the Information Commissioner's Office (ICO) in the UK.

4.6.5 Finance and Insurance

Integrating large amounts of data with business intelligence systems for analysis plays an important role in financial and insurance sectors. Some of the major areas for acquiring data in these sectors are exchange markets, investments, banking, customer profiles, and behaviour.

According to McKinsey Global Institute Analysis, "Financial Services has the most to gain from big data". For ease of capturing and value potential, "financial players get the highest marks for value creation opportunities". Banks can add value by improving a number of products, e.g., customizing UX, improved targeting, adapting business models, reducing portfolio losses and capital costs, office efficiencies, and new value propositions. Some of the publicly available financial data

are provided by international statistical agencies like Eurostat, World Bank, European Central Bank, International Monetary Fund, International Financial Corporation, Organization for Economic Co-operation and Development. While these data sources are not as time sensitive in comparison to exchange markets, they provide valuable complementary data.

Fraud detection is an important topic in finance. According to the Global Fraud Study 2014, a typical organization loses about 5 % of revenues each year to fraud. The banking and financial services sector has a great number of frauds. Approximately 30 % of fraud schemes were detected by tip off and up to 10 % by accident, but only up to 1 % by IT controls (ACFE 2014). Better and improved fraud detection methods rely on real-time analysis of big data (Sensmeier 2013). For more accurate and less intrusive fraud detection method, banks and financial service institutions are increasingly using algorithms that rely on real-time data about transactions. These technologies make use of large volumes of data being generated at a high velocity and from hybrid sources. Often, data from mobile sources and social data such as geographical information is used for prediction and detection (Krishnamurthy 2013). By using machine-learning algorithms, modern systems are able to detect fraud more reliably and faster (Sensmeier 2013). But there are limitations for such systems. Because financial services operate in a regulatory environment, the use of customer data is subject to privacy laws and regulations.

4.7 Conclusions

Data acquisition is an important process and enables the subsequent tools of the data value chain to do their work properly (e.g. data analysis tools). The state of the art regarding data acquisition tools showed that there are plenty of tools and protocols, including open-source solutions that support the process of data acquisition. Many of these tools have been developed and are operational within production environments or major players such as Facebook or Amazon.

Nonetheless there are many open challenges to successfully deploy effective big data solutions for data acquisition in the different sectors (see section “Future Requirements and Emerging Trends for Big Data Acquisition”). The main issue remains producing highly scalable robust solutions for today and researching next generation systems for the ever-increasing industrial requirements.

References

ACFE Association of Certified Fraud Examiners. (2014). *Report to the nations on occupational fraud and abuse*, Global fraud Study 2014. Available online at: http://www.acfe.com/rttn/docs/2014-report-to-nations.pdf

Bank of America et al. AMQP v1.0. (2011). Available online at http://www.amqp.org/sites/amqp.org/files/amqp.pdf

Bennett, D., & Harvey, A. (2009). *Publishing Open Government Data. W3C, Technical Report, 2009*. Available online at: http://www.w3.org/TR/gov-data/

Bradic, A. (2011) *S4: Distributed stream computing platform*, Slides@Software Scalability Belgrad. Available online at: http://de.slideshare.net/alekbr/s4-stream-computing-platform

Carzaniga, A., Rosenblum, D. S., & Wolf, A. L. (2000). Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pp 219–27.

Cugola, G., & Margara, A. (2012). Processing flows of information. *ACM Computing Surveys, 44* (3), 1–62. doi:10.1145/2187671.2187677.

Davenport, T. H. (2013). *At the Big Data Crossroads: turning towards a smarter travel experience*. Amadeus IT Group. Available online at: http://blogamadeus.com/wp-content/uploads/Amadeus-Big-Data-Report.pdf

DHL Solutions & Innovation Trend Research. (2013). *Big Data in Logistics. A DHL perspective on how to move beyond the hype*. Available online at: http://www.delivering-tomorrow.com/wp-content/uploads/2014/02/CSI_Studie_BIG_DATA_FINAL-ONLINE.pdf

Gabriel, G. (2012) Storm: The Hadoop of Realtime Stream Processing. *PyConUs*. Available online at http://pyvideo.org/video/675/storm-the-hadoop-of-realtime-stream-processing

Hasan, S., & Curry, E. (2014a). Approximate semantic matching of events for the internet of things. *ACM Transactions on Internet Technology* 14(1):1–23. doi:10.1145/2633684.

Hasan, S., & Curry, E. (2014b). Thematic event processing. In *Proceedings of the 15th International Middleware Conference on – Middleware '14*, ACM Press, New York, NY, pp. 109–120. doi:10.1145/2663165.2663335.

Hasan, S., & Curry, E. (2015). Thingsonomy: Tackling variety in internet of things events. *IEEE Internet Computing, 19*(2), 10–18. doi:10.1109/MIC.2015.26.

IBM. (2013). *Architecture of the IBM Big Data Platform*. Available online at http://public.dhe.ibm.com/software/data/sw-library/big-data/ibm-bigdata-platform-19-04-2012.pdf

Krishnamurthy, K. (2013). *Leveraging big data to revolutionize fraud detection, information week bank systems & technology*. Available online at: http://www.banktech.com/leveraging-big-data-to-revolutionize-fraud-detection/a/d-id/1296473?

Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Boston, MA: Addison-Wesley Longman Publishing Co.

Madsen, K. (2012) *Storm: Comparison-introduction-concepts, slides*, March. Available online at: http://de.slideshare.net/KasperMadsen/storm-12024820

McAfee, A., & Brynjolfsson, E. (2012). Big Data: The management revolution. *Harvard Business Review, 90*(10), 60–66. Available online at http://automotivedigest.com/wp-content/uploads/2013/01/BigDataR1210Cf2.pdf.

Neumeyer, L. (2011). Apache S4: A distributed stream computing platform, *Slides Stanford Infolab*, Nov. Available online at: http://de.slideshare.net/leoneu/20111104-s4-overview

Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. (2011). S4: Distributed stream computing platform, *KDCloud*. Available online at: http://www.4lunas.org/pub/2010-s4.pdf

Oracle. (2012). *Oracle information architecture: An architect's guide to big data*. http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf

Sensmeier, L. (2013). How Big Data is revolutionizing Fraud Dedection in Financial Services. *Hortonworks Blog*. Available online at: http://hortonworks.com/blog/how-big-data-is-revolutionizing-fraud-detection-in-financial-services/

Vivisimo. (2012). Big Data White Paper.