PalArch's Journal of Archaeology of Egypt / Egyptology

Survey on Big Data Acquisition tools and techniques

Arockia Jaya J[1]
[1]Associate Professor,
Department of Computer Science and Engineering
Idhaya Engineering College for Women

Dr. Mahalakshmi K[2]
[2]Professor,
Department of Computer Science and Engineering
Kalaignarkarunanidhi Institute of Technology

Arockia Jaya J[1], Dr. Mahalakshmi K[2], Survey on Big Data Acquisition tools and techniques - Palarch's Journal of Archaeology of Egypt/Egyptology 17(9). ISSN 1567-214x,

1. Introduction

In the past years, word big data have been used for labelling attributes by various players. Also, various methods to process architectures for big data were given to meet the various features big data. Altogether, data acquisition is known as the method to gather, filter, as well as clean data prior to storing the data in a data warehouse or any other solutions for storing.

The location of big data acquisition among the overall big data value chain are depicted in Fig. 1. Obtaining big data is very often decided by 4 Vs: volume, velocity, variety, and value. Many data obtaining scenarios provide high-volume, high-velocity, high-variety, yet low-value data, that made it of utmost importance to get adapted as well as time-efficient gathering, filtering, and cleaning algorithm which makes sure that specifically high-value parts of the data usually underwent processing by the data-warehouse analysis. Still few organizations, many data are of more value so that it could be of utmost importance to get new consumers. For such organization, data analyzing, classifying, and packing on a increased data capacities that provide the maximum central role post data obtaining.

The aim of this paper can be divided into 3 parts: Initially, its objective is to find out the current common needs for obtaining data by demonstrating open benchmark frame-work as well as protocol for obtaining huge data for companies. The second aim of this is unveiling the present methods utilized to obtain data in various sectors. At the end, we discuss the needs for obtaining data are got by present methods and feasible methods could happen in the same area.

2. Key Insights for Big Data Acquisition

For obtaining a better insight about the data acquisition, the part will initially show various big data architectures of Oracle, Vivisimo, as well as IBM. This will combine the method for acquiring among the big data processing pipeline.

The historical data show various methods by which the abstraction of big data pipeline. Oracle (2012) depends on the 3-step strategy to process data. Initially, the components of various source of data are obtained and stored in a confined scalable solution for storing like the NoSQL database or the Hadoop Distributed File System (HDFS). Details that have undergone storage follows by reorganizing and is stored in a big data analytics software that is capable of SQL. At last, it undergoes analysis with a big data analytics algorithm.

Velocity[1] depends on various views on a big data. In this the technique is often based

on search. The main component of the architecture is a connector layer, that can handle various data sources. The components of these data sources get collected together in parallel, converted, at last undergoes addition to an index, which builds the basis for data analytics, business intelligence, and overall data- driven applications. Other big players like IBM depend on architectures matching Oracle's. [2]

Around various architectures to big data processing, the base for acquiring data starts with collecting data from potential data sources that are fragmented with the moto of saving the data in scalable, big data-capable data storage. To accomplish this, the following components may be needed:

1. Protocols which permit to gather information for potential sources of data that are fragmented of any type (unstructured, semi-structured, structured)
2. Frameworks that collect the data from the distributed sources by using different protocols
3. Technologies which permit continuous storage of the data obtained from the frameworks

3. Social and Economic Impact of Big Data Acquisition

During the past few years, there is an increase in the sheer volume of data which is steadily generated. One tenth of the current data generated all over the world were within the past couple of years. The source and nature of this data is diverse. It varies starting from data obtained from sensors to data presenting the electronic transactions. There is a huge increase in data produced in social media. There is a diversity in the type of data (structured and unstructured) and semantics of the data. Still the overall data needs to be summed up to assist in finding solutions for the business queries and create an overall picture.

In business these data provide a huge opportunity and ideas in developing novel models for business and to improve the present technologies, hence deriving advantage in business. New methods and applications to handle the big data that are often driven by 4 Vs is utilized to make improved advertisement in market research that are customized as per the user needs. Consider smart metering systems which underwent research in energy sector. More than that, when combined with novel billing systems they could benefit in other segments such as communication and transportation.

Many business sectors are already have been impacted by the big data. Though there are various difficulties expected, there would be a great positive influence on managing the company, making decisions and even in the atmosphere and culture within the company[4].

There are number of limitations pertaining in the big data. For example, there is a need to address the privacy and confidentiality of the data used. Though there are large amount of data generated by various industries, only a minimal data has been used. Adding to that number of such systems are short of real-time needs.

4. Big Data Acquisition: State of the Art

The volume of big data acquisition is done confined to a message queuing paradigm, often known as the streaming paradigm, publish/subscribe paradigm [4], or event processing paradigm [5][6] Here, the fundamental thought is that manifold volatile data sources produce data that are essentially captured, stored, and analyzed by a big data processing system. The novel data created by the data source is sent to the data storage using a data acquisition framework which employs an already defined procedure. The following part explains the dual core technology for big data acquisition.

4.1.1 Protocols

A number of organizations that depends within themselves on big data process have

made SOPs that are specifically prepared for the organizations and among them which cannot be given to public and hence cannot be explained by us. The following section explains the big data that are publicly available.

4.1.1.1 AMQP

The causative factor that lead to the creation of Advanced Message Queuing Protocol (AMQP) was the demand for open protocol that could meet the data needs of large volume companies. For achieving this, 23 companies put on a line of specifications needed for a data acquisition protocol. The resulting AMQP (Advanced Message Queuing Protocol) became an OASIS standard in October 2012. The rationale behind AMQP [6] was to provide a protocol with the following characteristics:

a) **Ubiquity**: Ubiquity of AMQP implies its capability for being applied among various companies that includes present as well as futuristic architectures for acquiring data. AMQP's ubiquity was obtained by doing it to extend easily as well as its implementation simple. The huge volume of frameworks which applies it, includes SwiftMQ, Microsoft Windows Azure Service Bus, Apache Qpid, and Apache ActiveMQ, shows the ease of protocol implementation.

b) **Safety**: The safety characteristic was applied along two varying dimensions. Initially, the protocol permits the combining of message encryption for making sure even intercepted messages cannot undergo easy decoding. Hence, it can be employed to transfer business-critical data. The protocol is strong in tackling the injection of spam, creating the AMQP brokers hard to affect. Then, the AMQP makes sure the messages are durable, which means it permits messages to be transferred even when the sender and receiver are not available simultaneously.

c) **Fidelity**: The 3rd character depends with the continuity of the messages. AMQP takes into account means to make sure that the sender can reveal the semantics of the message hence allowing the receiver better understanding of the details he received. The protocol applies dependable failure semantics which permits the systems to identify errors from the generation of the message at the sender's side prior to the storage of data by the receiver.

d) **Applicability**: The aim of this characteristic is to make sure that AMQP clients as well as brokers can correspond with numerous protocols of the Open Systems Interconnection (OSI) model layers like the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), including Stream Control Transmission Protocol (SCTP). Through these ways, AMQP can be employed in multiple situations as well as companies in which not every protocol of the OSI model layers are needed and employed. Also, the protocol was structured to assist various messaging patterns which includes direct messages, requests/replies, publications/ subscriptions and so on.

e) **Interoperability**: The protocol was structured to be independent of specific situations as well as users. Hence, users and brokers who are completely independent for implementing, architectures, as well as ownership can correspond through AMQP. As described above, numerous frameworks from various companies can implement the protocol.

f) **Manageability**: An important concern while specifying the AMQP is to make sure that all frameworks which implements it could scale with ease. This was obtaining by making sure that AMQP is a tolerate faults as well as prevent losses wire protocol by which data of all types (e.g. XML, audio, video) can be transferred.

To apply these needs, AMQP depends on a type system as well as various 4 layers: a transport layer, a messaging layer, a transaction layer, and a security layer. The type

system depends on primitive types from databases (integers, strings, symbols, etc.), explained varieties from programming, as well as descriptor values which could undergo extension by the users of the protocol. Adding to that, AMQP permits the application of encoding to save symbols as well as values, the definition of compound varieties which has the combined forms of numerous primary varieties.

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" [6]. The fundamental notion that drives the architecture of the transactional messaging approach succeeded by the layer depends on person sending the message which acts as controller during which the receiver does the role of a resource as messages are moved as told by the controller. Through these ways, the decentralized as well as scalable method to process message can be obtained

The last AMQP layer known as security layer, enable to define the method to encrypt the details of AMQP message. The protocol to obtain these goals are needs to be clarified from outside from AMQP. Protocols which are applied to this part includes transport layer security (TSL) as well as simple authentication and security layer (SASL).

Because of its adoption throughout numerous companies as well as its improved flexibility, the AMQP may tend to become the benchmark strategy to process messages in the companies which may not be affordable for implementing their respective protocol. As the data are largely increasing, it provides itself to be the best possible solution for applying services surrounding data streams. The most often used AMQP brokers is RabbitMQ, which is popular because it applies numerous messaging protocols such as JMS.

Java Message Service

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

According to the 1.1 specification JMS "gives a general method for Java programs for creating, sending, receiving and reading messages from the enterprises messaging systems". Administrative tools permit binding of destination and connecting factories to a Java Naming and Directory Interface (JNDI) namespace. A JMS client could use resource injection for accessing the object that was given in the name space that can be later established to form logical connections for the similar object via JMS provider. The JNDI acts as the moderator in this case among various clients who needs to transfer information's. Make a point of the term "client" applied here (as the spec does) to depict both the one who sends and also the one who receives the information. Presently, JMS gives 2 messaging models: point-to-point as well as publisher-subscriber which has a one-to-many kind of connections.

AMQP can adapt with JMS, that is the de facto standard for sending message in the Java world. As AMQP is explained at the format level (i.e. byte stream of octets), JMS is made as standard at API level hence it is not simple for implementation in other programing languages (as the "J" in "JMS" depicts). Adding to that JMS do not facilitate functionality to balance the loads/tolerate faults, error/advisory notification, administration of services, security, wire protocol, or message type repository (database access).

A great benefit of AMQP is, though, the programming language independent of the

application which eliminates vendor-lock in and platform compatibility.

4.1.2 Software Tools

Considering software tools for acquiring data, lot of them are popular and number of them use cases which are accessible throughout web hence we can easily approach them initially. Still, the proper utility of every tools warrants an in-depth understanding of how it works internally as well as the implementing f the software. Various paradigms for acquiring data have been introduced based on the on the scope the tools which they have concentrated. The architectural diagram in Fig. 2 depicts an outline of the total big data workflow giving importance to acquiring data.
In the further part we will see the tools and other details related to data acquisition

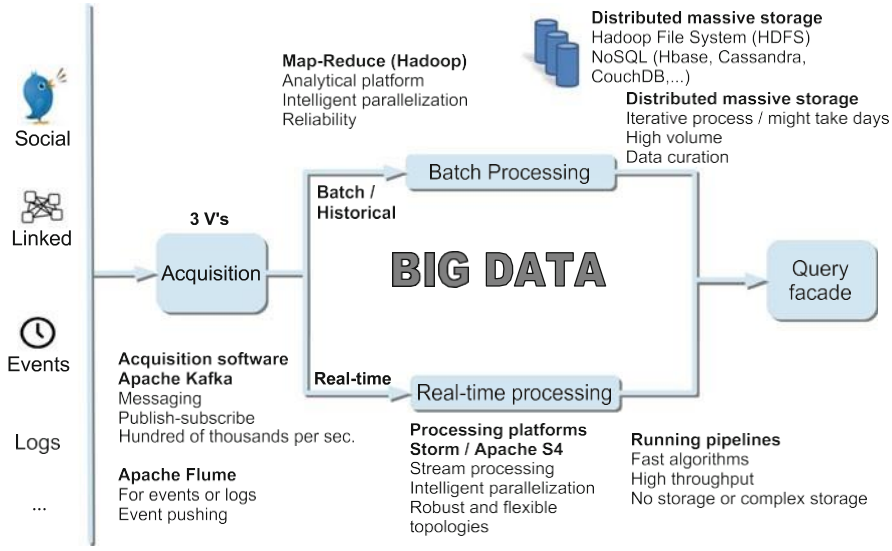

Fig. 4.2 Big data workflow

4.1.2.1 Storm

An open-source framework for the robust distributed real-time activities on continuously flowing data is called STORM. It originated as an open-source project which currently has a huge active community. Storm gives support to various programming languages as well as storage facilities such as relational databases, NoSQL stores. The prime benefit of Storm is utilization in multiple data collection situations such as stream processing as well as distributed RPC to solve intensive computational function on-the-fly, as well as continuous computation application [7]. Various organizations as well as application are making use of Storm for powering a various production system processing data, that includes Groupon, The Weather Channel, fullcontact.com, as well as Twitter. The logical network of Storm has nodes of 3 varieties : a master node known as Nimbus, a set of intermediate Zookeeper nodes, and a set of Supervisor nodes.
The Nimbus: similar to Hadoop's JobTracker: that feeds in the computation for execution, issues code over the cluster, as well as keeps an eye on computation.
The Zookeepers: handling overall cluster coordination. This layer depends on the Apache ZooKeeper project.
The Supervisor Daemon: spawns worker nodes; It can be compared to Hadoop's TaskTracker. Here maximum work of application developers goes into. The worker nodes correspond to Nimbus through the Zookeepers for finding out what to run on the machine, starting as well as stopping workers.
A computation in otherwise known as topology in Storm. When they are employed, topologies function continuously. There are 4 concepts and abstraction layers in Storm:
Streams: unbounded sequence of tuples, which are named lists of values. Values may

be a random object that implements a serialization interface.
Spouts: They are the starting point for computing.
Bolts: which can compute innumerable input streams an generate innumerable output streams. Here the maximum application logic go.
Topologies: The high level abstraction of storm. Fundamentally, a topology is interconnections of network of spouts and bolts linked by edges. All edges are a bolt subscribing to the stream of a spout to the other. Spouts as well bolts are nodes with no state as well as parallel by nature, does number of functions along the cluster. Taking physical point a worker is a Java Virtual Machine (JVM) process with numerous tasks that runs along. Spouts as well as bolts are dispersed along numerous tasks and workers. Storm gives support to numerous stream grouping strategies starting with random grouping to tasks, to field grouping, in which place tuples are summed up by specific fields to the same tasks [8][9].
Storm makes use of a pull model; every bolt retrieves events from its source. Tuples change the overall interconnections confined to a specific period of time otherwise taken as a failure. Hence with respect to recovery the spouts take the responsibility to maintain tuples ready for replay.

4.1.2.2 S4

S4 (simple streaming system that can be scaled) is a dispersed, general-purpose place for the development of applications that processes large flowing data. Initiated by 2008 by Yahoo! Inc., from2 011 it is an Apache Incubator project. S4 is made in such a way to use in commodity hardware, neglecting I/O bottlenecks by depending on an all-in-memory strategy [10].
A stream in S4 is sequential form of elements (events) of tuple-valued keys as well as attributes. A fundamental computation unit PE is depicted by the components as follows: (1) its functionality given by the PE class as well as the related configuration, (2) the event variety used, (3) the keyed attribute in this event, finally (4) the value of the keyed attribute of the consuming events. A PE is expressed by the platform for all the values of key attributes. Pes without keys are important classes of PEs that lacks keyed attribute as well as value. The above PEs takes up all events of the respective variety and specifical at the input layer of an S4 cluster. There is a huge volume of standard PEs accessible to numerous specific tasks like aggregate as well as join. PEs are logical hosts of PEs are the processing nodes (PNs). PNs watch the events, start its functions for incoming events, as well as dispatch events along with the support of the layer used for communicating.
S4 routes all events to processing nodes using a hash function among all the known value 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 does production, consumption, as well as transmission of messages, while PE situations are clones of the respective prototype that has the state and are associated with unique keys [11].
Due to the designing S4 makes sure that every event with a typical value of the keyed attribute arrives at the respective PN as well as inside it is routed to the particular PE instance [12]The present condition of a PE cannot be accessed by other PEs. S4 relies on a push model: events are routed to the next PE quickly. Hence, if a receiver buffer fills the events may be left. Through lossy checkpointing S4 gives state recovery. If the node crashes a novel node does its work from the latest snapshots. The communication layer is from the Apache ZooKeeper project. The cluster and provides failover handling to stand-by nodes is managed by that. PEs are constructed in Java with a very simplified API also they are joined together into the application with the help of a Spring framework.

4.1.2.3 Kafka

Kafka is a dispersed publish-subscribe messaging system constructed for supporting primarily continuous messaging to high-throughput. Kafka's objective is to combine offline as well as online processing giving a technique for a parallel load into Hadoop and ability to divide real-time usage among a cluster of machines. The function for activity stream process develops Kafka almost equal to Apache Flume, still architecture as well as primitives widely vary making Kafka more like the conventional messaging system.

Basically, Kafka was designed at LinkedIn to track the large data of events produced in the site. They are important to detect consumers preferences and for the improvement of data-driven products. The earlier figures provide a simple view of the deployment topology at LinkedIn.

Make a note that an isolated Kafka cluster manages all the functions of data from various sources. This provides a unique pipeline of information for online as well as offline users. This tier functions as a buffer among live activity as well as asynchronous function. Kafka can act such as for replication all data to another data centre so that it can be used offline. Kafka also can make Hadoop analyze offline, track internal operational metrics which gives graphs as input in real time. As we can see the perfect use of Kafka, its publish-subscribe technique would be processing associated stream data, which starts from identifying consumers' activity on huge volume website to relating and giving ranks to activities.

In Kafka, every stream is known as "topic". Topics are divided for scaling. Senders of information gives a key that can find out the partition the information is addressed to. Hence, all the information partitioned by the same key are ensured to be in the same partition. Kafka brokers manage few partitions and get back and save information by the sender.

Kafka user reads from a topic by obtaining the message of every partition of the topic. In case a user wishes to read all messages using particular key, they need to read messages from the specific keyed partition and not the whole topic. Also, any line of the reference in a broker's log file with the help of an offset can be referred. This offset decides the specific topic/partition for the reader. When a reader reads a specific topic then the partition is stepped up.

Kafka gives a minimum one messaging guarantee as well as highly accessible partitions. For storing and cache message Kafka uses file systems, while the data are transcribed at once into a permanent log eliminating the necessity to flush the disk. The protocol is constructed on a message set abstraction when combines, thus grouping the messages. This leads to minimization of the network overhead as well as sequential disk operations. Sender as well as receiver have the same format of message.

4.1.2.4 Flume

Flume is a used for effective collection as well as movement of huge volume data logs. This is easy to access and is built based on the stream of data flow. It is powerful and can tolerate mistakes with reliability mechanisms that can be tuned . It also has multiple fail over as recovery mechanisms. It makes use of a simple data model which can be extended and also permits applications for online analysis. The system has 4 main characters: reliable, scalable, manageable, and extendable

The flume is designed keeping in mind some ideas that when combined can assist in attaining our goal:

- Event: a byte payload that has an choice of having a string headers which depicts the unit of data that Flume can transfer from the source to the destination.
- Flow: The flow of events from the source to destination is known as data flow or flow.
- Client: an interface implementation which functions at source and and gives them to a Flume agent.
- Agent: an independent function which hosts the constituents of flume like the

source, channel, and sink, which hence has the capability for receiving, storing, and forwarding events to next-hop destination.

- Source: an interface implementation which uses the events given to it through particular technique.
- Channel: a temporary store for events, in which events are given to the channel through the source operating within the agent. An event put in a channel remains in that channel till a sink takes it further
- Sink: an interface implementation which can take the events stored in a channel and transport it to the succeeding agent in the flow, or to the final place.

4.1.2.5 Hadoop

An open-source project developing a framework that gives computing on big data with clusters of commodity hardware with reliability, scalability, and dispersed is an Apache Hadoop. This is obtained from Google's MapReduce as well as Google File System (GFS) and transcribed in JAVA. The consumers and supporters of this Hadoop is a huge group of people. They use it for production as well as research conditions in various companies like: Facebook, a9.com, AOL, Baidu, IBM, Imageshack, and Yahoo. The Hadoop project has 4 modules:

i. Hadoop Common: for common utilities used throughout Hadoop.
ii. Hadoop Distributed File System (HDFS): as easily accessible as well as efficient file system.
iii. Hadoop YARN (Yet Another Resource Negotiator): a framework to schedule jobs and manage clusters.
iv. Hadoop MapReduce: a system to process huge volume of data parallelly.Using Hadoop project numerous associated projects have begun. For example, the Apache pig project us constructed on Hadoop for simple transcription and maintenance of Hadoop implementation. Hadoop is highly effective in batch processing. The Apache HBase study's objective is to give an on-time access to big data.

References

[1] 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
[2] Bank of America et al. AMQP v1.0. (2011). Available online at http://www.amqp.org/sites/amqp. org/files/amqp.pdf
[3] Bennett, D., & Harvey, A. (2009). *Publishing Open Government Data. W3C, Technical Report, 2009*. Available online at: http://www.w3.org/TR/gov-data/
[4] 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
[5] in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pp 219–27.
[6] Cugola, G., & Margara, A. (2012). Processing flows of information. *ACM Computing Surveys, 44*
[7] (3), 1–62. doi:10.1145/2187671.2187677.
[8] Davenport, T. H. (2013). *At the Big Data Crossroads: turning towards a smarter travel experi- ence*. Amadeus IT Group. Available online at: http://blogamadeus.com/wp-content/uploads/ Amadeus-Big-Data-Report.pdf
[9] 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
[10] 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
[11] 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.
[12] Hasan, S., & Curry, E. (2014b). Thematic event processing. In *Proceedings of the 15th Interna- tional Middleware Conference on – Middleware '14*, ACM Press, New York, NY, pp. 109–120. doi:10.1145/2663165.2663335.
[13] 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.
[14] 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
[15] 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?
[16] Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Boston, MA: Addison-Wesley Longman Publishing Co.
[17] Madsen, K. (2012) *Storm: Comparison-introduction-concepts, slides*, March. Available online at: http://de.slideshare.net/KasperMadsen/storm-12024820
[18] 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.
[19] 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
[20] 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
[21] 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
[22] 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-revolu tionizing-fraud-detection-in-financial-services/
[23] Vivisimo. (2012). Big Data White Paper.