NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696
VOLUME 4, ISSUE 9, Sep.-2017

COMPONENTS AND DEVELOPMENT IN BIG DATA SYSTEM: A SURVEY

Sudhir Allam,
Sr. Data Scientist, Department of Information Technology, USA

ABSTRACT
This paper examines the principal components and develops the big data method. Through the expansion of distributed processing platforms, new Big Data research application items often have a variety of features. When people come into touch with Big Data systems early on, it is difficult for them to make choices [1]. The components designed for the big data framework based on specifications are presented in this paper. Conventional databases may not handle unstructured data and high amounts of real-time data sets. Various databases contribute to large data and it is difficult, utilizing conventional database methods, to archive, maintain, process, interpret, visualize, and derive valuable information from such data sets [2]. However, there are also technical aspects to the refining of massive heterogeneous Big Data trends. The components explored in this paper involve those in the data access layer, data collection layer, data analysis, and data processing layer of the big data system

Keywords: Big Data system, Hadoop, MapReduce, HBase, NoSQL databases

INTRODUCTION
In our everyday lives, big data applications are commonly utilized. The primary objective and central aspect of Big Data is data analytics. With numerous data from different digital outlets, the role of data analytics has increased dramatically all these years [2]. This huge quantity of data produced at a ferocious rate and in all sorts of configurations is what we consider big data nowadays. But the storage of these data on the traditional structures we have for a long time is not feasible. A large variety of computational tools are accessible for analyzing complex structured and unstructured data, each configured to manage particular data/workload categories, in line with the need to store and evaluate the growth of complex data in real-time. This paper aims at providing a common view of the whole Big Data structure, which consists of many phases and main components of any level of big data processing [3]. This paper helps in understanding the different components of the big data systems like distributed file systems and MapReduce enabled NoSQL databases concerning some data storage parameters [4]. Big data systems are a special application environment built on the integration of computer engines owing to their diverse modules and periodic changes. The products and their advancements in integrating different data sources can be found in this article. The precise layer of these separated components is dependent on their normal case of usage [4].

RESEARCH PROBLEM
The main problem that this explorative study aims at solving is to identify and explore the development of big data system components. Material science and heterogeneous computation have developed a lot in recent years. The use of hybrid architecture as the framework for Big Data applications improves business awareness [5]. Following the unprecedented storage layer approaches, high-performance computer chips are designed for the device. Fixing conventional database output bottlenecks with operating systems has been a trend. The data produced today cannot be processed by traditional databases, since the majority of today's data are semi-structured or unstructured [5]. However, conventional programs were only configured to accommodate well-designed columns and rows of organized data. Today's data are housed in many silos. It can be a challenging job to get them together and analyze them for trends [6]. For years the computing ability of application servers was multiplying, however, due to its restricted size and speed, databases have lagged. Today, however, since several systems generate large data for processing, different big data device components may be beneficial.

LITERATURE REVIEW

The section examines in detail the components of a big data system and its development. It introduces some common features for each big data system layer and their usage for the application of the layer and the main algorithms or techniques.

Database Access Components

Components in data access in big data systems usually include clear connections to data servers, with a special focus on real-time querying, searching, and distribution of data. They go beyond data management. The three major categories which are divided depending on model calculations and interfaces are a multi-dimensional query feature, SQL-based computing component, and the real-time querying component [7].

Real-Time Querying Component

Apache Drill/Dremel, Facebook Presto, and Impala are by far the most common real-time query components. In November 2013, Facebook published Presto's source code. It was a form of distributed SQL query tool is developed to perform high-performance real-time data processing. It provides hierarchical database language (SQL) as an essential operating interface that contains dynamic queries, aggregates, attachments, and window capabilities [7]. Presto provided a basic storing data abstract model to accommodate queries that contain data in various applications (like HBase, HDFS, and Scribe). It uses custom execution software and answer operators to enable SQL syntax [8]. Impala is a Cloudera developed dynamic SQL Big Data query platform focused on the Dremel system and massively parallel MPP (Massive Parallel Processing) design of Google [9]. Impala uses a distributed query engine composed of a query planner, a query organizer, and a query exec engine.

SQL-Based Computing Component

This component is intended to assist data scientists in their work. Its services recognize the complexity of data storage and computation capabilities by providing SQL-like languages [9]. These items aren't authentic SQL query solutions but rather have a SQL-like command interface to perform operations with each computer simulation (like Hive's MapReduce tasks and Spark SQL's resilient distributed data collection (RDD) [11].

1) Hive: Hadoop is a data storage framework built on Hadoop. It transforms hierarchical files and folders to database tables and contains an important SQL query feature. Hive is a data storage facility that turns Configuration files into MapReduce work. Users will simply use SQL-type statements to perform complex MapReduce statistics because the learning process is limited [12].

2)Pig: Pig is a Hadoop-based data analysis platform.

This is a Hadoop-based data analysis application. Pig Latin is utilized and is equivalent to SQL. It acts as a compiler that converts SQL-like queries to a series of simplified MapReduce operations [12]. Pig is an innovative, big data parallel processing application that is simple to use. Pig's central configuration and process of execution are the same as Hive's. The MapReduce engine implements both of these products, converting the SQL/SQL statements into physical and logical plans. Pig, unlike Hive, has its very own syntax called Pig Latin, and in terms of programming definitions and data, structures are substantially different from ANSI SQL [12].

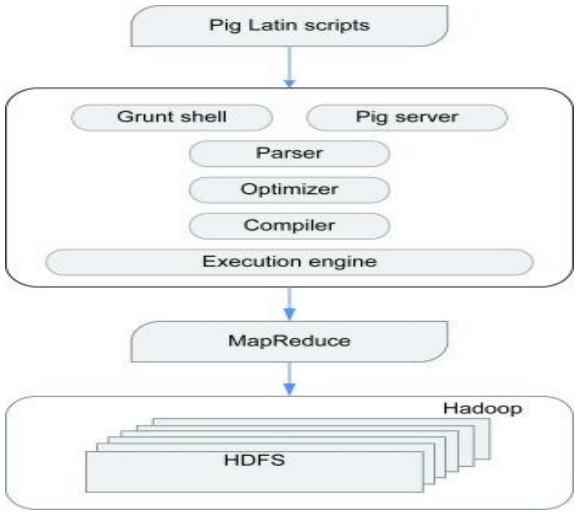

Fig i: Apache Pig Architecture

a) Programming ideas: Pig Latin is a descriptive language of programming, while SQL is a language of information flow programming. Pig defines the data processing and conversion procedure, and only the data to be converted is specified by SQL. Pig files are similar to SQL query proposals, converting concise results to a level into process steps.

b) Data structure: The data is used in the table and linked to the SQL scheme. The data structure of Pig is ad hoc since it decides schemes during recovery. Pig often enables complex nested data structures in comparison to the flat table architecture of SQL [13]. User design (UDF) and streaming (UDF adaptation in many languages) increase the tool's usability and power.

3)Spark SQL: Shark is Spark SQL's precursor. Because Hive runs on Hadoop, in the computer step MapReduce, a large number of medium disk landing processes use a lot of I/O. Just like with Hive, Spark SQL provides a fast start-up platform for engineers who are familiar with relational databases but are unfamiliar with Spark RDD. Spark community developers have revised Hive's optimizer and implementation strategy to access Spark, an in-memory device [13]. The Shark SQL project has eliminated the specialized components of Hive, for example, the syntax parser and query optimizer, thus boosting system accessibility and operational efficiency and improving feature enlargement considerably. Spark can use many optimization techniques to increase performance, including in-memory column space and byte code formation [13]. This model also makes adapting the SQL parsers, architects, and optimizers to suit their own needs even simpler for developers.

Multi-dimensional query component

Apache Kylin is a multi-dimensional query component. Even though database systems have largely replaced traditional systems, relational databases as the basic data processing services for Big Data architectures, database systems remain the bedrock of the underlying platform [14]. Reasonable Online analytical processing (ROLAP) is an improvement in the OLAP approach that performs deep, multi-dimensional processing of related database results. Such query modules may perform a multi-dimensional activity in real-time on the output of a database engine system [14]. Kylin is an optimizing Hadoop/Spark database that uses ANSI SQL to provide the most often queried user interface. Just a few of the data analytics that Kylin will specifically combine with Tableau, Qlik, SensePower BI/Excel, as well as SuperSet [14]. The users deal with Hadoop data with Kylin, which outperforms Hive, in fractions of a second. Kylin's most significant feature is that it helps users to define and pre-create a data cube in Kylin utilizing big data papers.

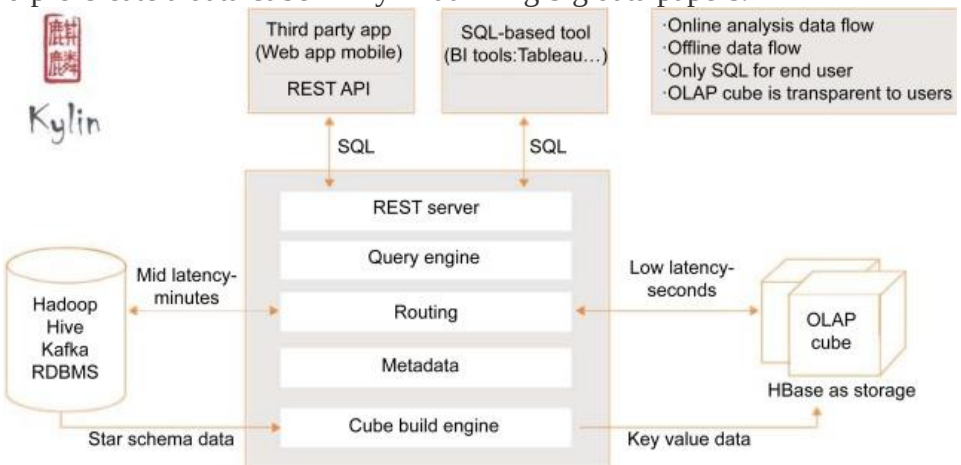

Fig ii: Architecture of the Apache Kylin

Data collecting components

Data collection is particularly critical in the Big Data loop. The Internet contains nearly unlimited material about a variety of topics. The activities of users provide a range of knowledge that helps to enhance infrastructure and to identify demand. We can include some of the most popular elements according to the different aspects of these programs.

1. HDFS

The Hadoop file system (HDFS)is a distributed file system that holds massive files in a streamed data access format and operates on commodity hardware. It's a file system that allows data to be shared by several servers in a network system. This makes it possible to satisfy requirements for low latency for access to records, storing huge amounts of small files, writing many users, and arbitrary modification of files. HDFS has a most common filesystem for offline collection purposes [14]. The standard master/slave architecture is used for HDFS. A single NameNode and several DataNodes are part of the HDFS cluster. NameNode is the master server that holds the domain of the file system, runs namespace transactions, and maps blocks into DataNodes. As with DataNodes, they are processed for the administration of storage resources connected to the nodes they operate. Access control and crisis backup are enforced through the block duplication process, the file within the storage structure is divided into blocks and stored in a series of DataNodes [14].

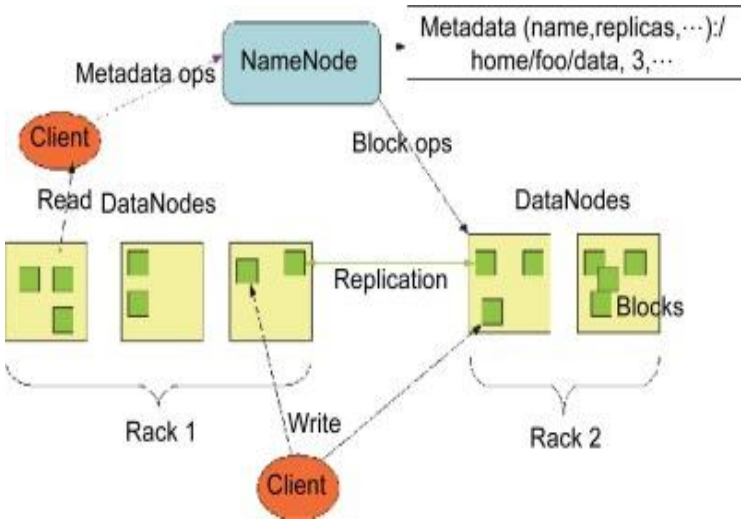

Fig iii. HDFS system architecture.

2. HBase

HBase is a column-oriented, distributed open-source database that incorporates the HDFS-based BigTable[15]. HBase is a Hadoop subproject of Apache. Unlike a standard relational database, HBase is an unstructured database and is column-based instead of row-based.

A standard HBase cluster comprises an HMaster and many HRegionServer-led slave nodes. The log module for each node of HRegionServer and multiple Regions are separated by keys into a section in the entire registered table. HRegion's minimum storage unit and load balance are often allocated. The HBase co-operation mode is also a traditional master-slave model, consisting of many layers which process various workloads. HMaster works with ZooKeeper, the cluster management framework that allocates HRregion to and balances each HRegionserver, discovers and relocates invalid HRegionServer, collects HDFS waste, and handles schema Upgrade queries.

3.NoSQL storage system

NoSQL applies to a database that is not related. With the growth of Internet web 2.0 technologies, in vast scales of data sharing, the conventional relational database cannot meet the ultra-wide and strong competitive requirements. To address the challenges of large data collection, particularly Big Data application problems, multiple types of data were developed in the NoSQL database [16]. NoSQL is difficult to describe since any form has its advantages and its focal point. Recent years have seen significant growth in the memory database. This archive saves much of the contents of your memory rather than on your computer or other external storage. The most popular in this field are LevelDB and RocksDB [16]. The primary problem with blockchain technologies is version management. OrpheusDB offers version control facilities in the data set as a Highlight while ForkBase uses reusable database (SIRI) conceptually invariant indexes for blockchain and fork-able frameworks as an effective storage engine.

Components in data analysis

The data analysis module's components are primarily modules designed for various programming languages. R and Python are connected, signifying a programming stack built on scripting. Both utilize data processing statistical techniques [17]. The primary distinction between the two methodology stacks is that Python is more oriented toward engineering advancement and includes direct support for established algorithms such as word segmentation and clustering that are needed for several projects, while R is more oriented toward statistical drawing and includes native support for visualization. The R programming language makes very little references to its functional architecture, preferring to function as an interface to other systems or Big Data engines. Both languages are focused on statistical sampling. Since the scripting language is dependent on the characteristics of its execution environment, it has minimal support for large-scale data. They sometimes need collaboration with C++ or other more powerful and lower-level languages.

Due to its advanced cross-platform infrastructure, comprehensive development suite assistance, and solid engineering management framework, the Java virtual machine (JVM)-based domain is significantly more systematic. Numerous data analysis applications have appeared, the majority of which are built on Apache Hadoop and Apache Spark [17]. Both have a comprehensive environment for designing suites and a robust and fully functional interface package.

Components in the data processing

Big Data is an umbrella concept for technology that allows the collection, organization, and processing of massive datasets that cannot be solved using conventional methods. With the increase in prominence, the size and value of various types of computation have grown significantly in recent years.

1)Apache Hadoop

Apache Hadoop is a distributed batch processing platform. It was the first Big Data system, and the open-source ecosystem has significantly enhanced it. As per Google's articles and publications, Hadoop reimplemented the algorithms and modules, resulting in the following components that function together to process batch info.

a) HDFS: HDFS is a module of the distributed file system that coordinates storage resources and data replication through cluster nodes [17]. Due to its performance, HDFS is commonly used as the storage mechanism in Big Data frameworks.

b) YARN: YARN is the Hadoop stack's cluster management portion, in charge of organizing and handling underlying resources and scheduling job execution.

c) MapReduce: MapReduce is the native batch processing engine included with Hadoop. Its computing strategy is focused on the map and shuffles and reduces the algorithm [17]. This technique is typically slow due to its heavy reliance on permanent storage and many readings and writing operations per job. However, since storage space is one of the most available cloud tools, MapReduce is capable of processing massive datasets at a low cost. Since Hadoop clusters do not use memory, they will operate on less costly hardware than other alternatives. Hadoop has tremendous scalability capacity and has even been used in a production cluster of thousands of nodes thanks to the cluster manager YARN.

2)Apache storm

Apache Storm is a low-latency stream processing system. Apache Storm satisfies near-real-day computing requirements. It is capable of processing vast amounts of data and delivering performance with minimal latency. Apache Storm provides stream processing through topologies, a scheduler focused on guided acyclic graphs (DAGs) [17]. These topologies imply that each data object would be subjected to the transformations and measures. Streams, spouts, and bolts include topologies. The stream is an infinite amount of data that continually enters the device. Spouts are data stream origins located at the topology's tip.

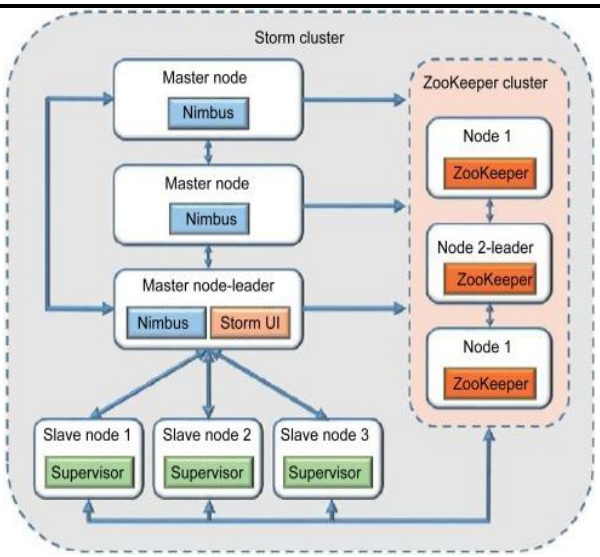

Fig iv: Structure of Storm Cluster

3)Apache Spark

To accommodate a variety of use cases inside application scenes, frameworks employ plugins and integrated interfaces to manage batch and stream workloads. Apache Spark is a platform for batch processing that combines SQL, streaming, and advanced analytics. Spark streaming is the part that manages the encoding of streams[18]. With regards to batch processing, Apache Spark focuses on accelerating batch processing workloads through the usage of completely in-memory computing and optimization. Unlike MapReduce, Spark processes all data in memory and communicates with the storage layer only at the start and end of the operation to initialize the data in memory and persist the final output. [18]. Memory is used to handle all intermediate outcomes. Several exceptions, such as cache or OOM, can cause the device to perform write operations. Spark implements in-memory computing using a model named RDD. These immutable constructs remain in memory and display arrays of data that can be traced back to their source, preventing each time access to disk storage. It will improve performance on disk-related activities by generating DAGs to display all the operations on the data that must be performed and their relationships [18]. The scheduler can delegate workloads based on the order of the nodes in DAGs, which will improve the intelligence of the coordinate work between the processors. Spark streaming, a module that buffers the stream in sub-second intervals, is Apache Spark's stream processing solution.

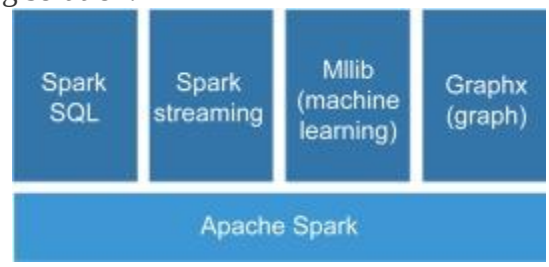

Fig iv: Apache Spark

THE FUTURE IN THE UNITED STATES

Big data has been a top concern for several technology firms in the United States. Enterprises and customers alike are becoming more interested in harnessing the potential of big data and using data analytics technologies. There are several reasons to assume that the future effect would be significant and widespread since there is a clear collection of levers that companies may use to generate demand through industries. The effect is magnified even as economic impact is measured in terms of demand surplus rather than conventional indicators of productivity. [19] Although certain markets are primed for greater gains, businesses in all industries will benefit from big data. Indeed, along with talent, technology, energy/shale, and trade, big data is projected to be one of the five standouts for US development.

ECONOMIC BENEFITS FOR THE UNITED STATES

This study would be critical for businesses and the government in the United States in deciding which components can be implemented in their operations to increase device performance [19]. Numerous businesses in the United States are embracing large data analytics. For the retail sector, $30-55 billion in total savings are projected in the United States through the adoption of big data tools to supply chain, operational activities, and merchandise sales, including hundreds of billions in additional earnings shifts. In consumer products, digital data is expected to generate $520 billion to approximately $1.5 trillion in value globally through advancements in product design, business processes, and promotion. In manufacturing, using big data through R&D, distribution, and supply chain management could result in savings of between $125 and 270 billion for the United States [19]. Most of the importance mentioned previously relates to economic increases or rises in GDP per capita. Big data allows enterprises to increase their productivity by managing labor, infrastructure, and procedures – while lowering the inputs required to achieve a specified production amount. However, some of the importance described previously requires increasing performance, more and better output, for a specified amount of input, which is typically more difficult to calculate.

CONCLUSION

This article addressed the components of a big data system and its development. It discussed the significance and specifics of each aspect. The various elements have varying degrees of importance for various businesses and ventures. The modern Big Data platform architecture faces the following challenges in terms of achieving a unified structure: to support as many application scenarios as possible, each product is often designed to be extremely broad and often requires additional calculation models to adapt to the need.

REFERENCES

1) N. Kaur and S. Sood, "Efficient Resource Management System Based on 4Vs of Big Data Streams", Big Data Research, vol. 9, pp. 98-106, 2015.
2) D. Agrawal, S. Chawla, A. K. Elmagarmid, et al., “Road to freedom in big data analytics,” in Proc. of the 19th Intl. Conf. on Extending Database Technology, pp. 479-484, 2016.
3) G. Aletti and A. Micheletti, "A clustering algorithm for multivariate data streams with correlated components", Journal of Big Data, vol. 4, no. 1, 2016.
4) E. Kolker and E. Kolker, "Healthcare Analytics: Creating a Prioritized Improvement System with Performance Benchmarking", Big Data, vol. 2, no. 1, pp. 50-54, 2014.
5) M. Naimur Rahman, A. Esmailpour and J. Zhao, "Machine Learning with Big Data An Efficient Electricity Generation Forecasting System", Big Data Research, vol. 5, pp. 9-15, 2016.
6) Gog, M. Schwarzkopf, N. Crooks, M. P. Grosvenor, A. Clement, and S. Hand, “Musketeer: All for one, one for all in data processing systems,” in Proc. of the 10th European Conf. on Computer Systems, 2015, DOI: 10.1145/2741948.2741968
7) Thusoo, Z. Shao, S. Anthony, et al., “Data warehousing and analytics infrastructure at Facebook,” in Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, pp. 1013-1020,2010.
8) Sheth, “Transforming big data into smart data: Deriving value via harnessing volume, variety, and velocity using semantic techniques and technologies,” in Proc. of IEEE 30th Intl. Conf. on Data Engineering, p. 2, 2014.
9) Thusoo, Z. Shao, S. Anthony, et al., “Data warehousing and analytics infrastructure at Facebook,” in Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, 2010, pp. 1013-1020,2010.
10) X.-R. Meng, J. Bradley, B. Yavuz, et al, “MLlib: Machine learning in Apache Spark The Journal of Machine Learning Research, 17 (1) (20, pp. 1235-1241,2016.
11) N. Elgendy and A. Elragal, “Big data analytics: A literature review paper,” in Proc. of Industrial Conf. on Data Mining, pp. 214-227,2014.
12) Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, “Pig Latin: A not-so-foreign language for data processing,” in Proc. of ACM SIGMOD Intl. Conf. on Management of Data, 2008, pp. 1099-1110.
13) M. Armbrust, R.-S. Xin, C. Lian, et al., “Spark SQL: Relational data processing in Spark,” in Proc. of ACM SIGMOD Intl. Conf. on Management of Data,pp. 1383-1394,2015.

14) M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing, pp. 1-7,2010.
15) L.-Q. Xu, S.-L. Huang, S.-L. Hui, A. J. Elmore, and A. Parameswaran, "OrpheusDB: A lightweight approach to relational dataset versioning," in Proc. ACM Intl. Conf. on Management of Data, pp. 1655-1658, 2016.
16) S.-M. Wu, K.-H. Lin, and L.-P. Chang, "KVSSD: Close integration of LSM trees and flash translation layer for write-efficient KV store," in Proc. of Design, Automation & Test in Europe Conf. & Exhibition, pp. 563-568.2016.
17) J. Dean, "Challenges in building large-scale information retrieval systems: Invited talk," in Proc. of the 2nd ACM Intl. Conf. on Web Search and Data Mining, 2009, p. 1.
18) Y. M. Zaharia, R.-S. Xin, P. Wendell, et al., "Apache Spark: A unified analytics engine for large-scale data processing", Communications of the ACM, vol. 59, no. 11, 2016, DOI: 10.1145/2934664
19) A.Byers, "Big Data, Big Economic Impact? I/S: A Journal of law and policy for the information society", vol. 10, no. 3 (2015), 757-764, 2015.