Journal of King Saud University – Computer and Information Sciences 30 (2018) 431–448

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Big Data technologies: A survey

Ahmed Oussous [a], Fatima-Zahra Benjelloun [a], Ayoub Ait Lahcen [a,b,*], Samir Belfkih [a]

[a] LGS, National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco
[b] LRIT, Unité associée au CNRST URAC 29, Mohammed V University in Rabat, Morocco

ARTICLE INFO

Article history:
Received 2 January 2017
Revised 3 June 2017
Accepted 4 June 2017
Available online 12 June 2017

Keywords:
Big Data
Hadoop
Big Data distributions
Big Data analytics
NoSQL
Machine learning

ABSTRACT

Developing Big Data applications has become increasingly important in the last few years. In fact, several organizations from different sectors depend increasingly on knowledge extracted from huge volumes of data. However, in Big Data context, traditional data techniques and platforms are less efficient. They show a slow responsiveness and lack of scalability, performance and accuracy. To face the complex Big Data challenges, much work has been carried out. As a result, various types of distributions and technologies have been developed. This paper is a review that survey recent technologies developed for Big Data. It aims to help to select and adopt the right combination of different Big Data technologies according to their technological needs and specific applications' requirements. It provides not only a global view of main Big Data technologies but also comparisons according to different system layers such as Data Storage Layer, Data Processing Layer, Data Querying Layer, Data Access Layer and Management Layer. It categorizes and discusses main technologies features, advantages, limits and usages.
© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Contents

* Corresponding author at: LGS, National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco.
E-mail addresses: ahmed.oussous@outlook.com (A. Oussous), fatimazahra.benjelloun@outlook.com (F.-Z. Benjelloun), ayoub.aitlahcen@univ-ibntofail.ac.ma (A. Ait Lahcen), samir.belfkih@univ-ibntofail.ac.ma (S. Belfkih).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2017.06.001
1319-1578/© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Nowadays, large data volumes are daily generated at unprecedented rate from heterogeneous sources (e.g., health, government, social networks, marketing, financial). This is due to many technological trends, including the Internet Of Things, the proliferation of the Cloud Computing (Botta et al., 2016) as well as the spread of smart devices. Behind the scene, powerful systems and distributed applications are supporting such multiple connections systems (e.g., smart grid systems (Chen et al., 2014a), healthcare systems (Kankanhalli et al., 2016), retailing systems like that of Walmart (Schmarzo, 2013), government systems (Stoianov et al., 2013), etc.).

Previously to Big Data revolution, companies could not store all their archives for long periods nor efficiently manage huge data sets. Indeed, traditional technologies have limited storage capacity, rigid management tools and are expensive. They lack of scalability, flexibility and performance needed in Big Data context. In fact, Big Data management requires significant resources, new methods and powerful technologies. More precisely, Big Data require to clean, process, analyze, secure and provide a granular access to massive evolving data sets. Companies and industries are more aware that data analysis is increasingly becoming a vital factor to be competitive, to discover new insight, and to personalize services.

Because of the interesting value that can be extracted from Big Data, many actors in different countries have launched important projects. USA was one of the leaders to catch Big Data opportunity. In March 2012, the Obama Administration launched Big Data Research and Development Initiative (Weiss and Zgorski, 2012) with a budget of 200 million. In Japan, Big Data development became one important axe of the national technological strategy in July 2012 (Chen et al., 2014b). The United Nations issued a report entitled *Big Data for Development: Opportunities and Challenges* (Letouzé, 2012). It aims to outline the main concerns about Big Data challenges and to foster the dialogue about how Big Data can serve the international development.

As a result of the different Big Data projects across the world, many Big Data models, frameworks and new technologies were created to provide more storage capacity, parallel processing and real-time analysis of different heterogeneous sources. In addition, new solutions have been developed to ensure data privacy and security. Compared to traditional technologies, such solutions offer more flexibility, scalability and performance. Furthermore, the cost of most hardware storage and processing solutions is continuously dropping due to the sustainable technological advance (Purcell, 2013).

To extract knowledge from Big Data, various models, programs, softwares, hardwares and technologies have been designed and proposed. They try to ensure more accurate and reliable results for Big Data applications. However, in such environment, it may be time consuming and challenging to choose among numerous technologies. In fact, many parameters should be considered: technological compatibility, deployment complexity, cost, efficiency, performance, reliability, support and security risks. There exist many Big Data surveys in the literature but most of them tend to focus on algorithms and approaches used to process Big Data rather than technologies (Ali et al., 2016; Chen and Zhang, 2014; Chen et al., 2014a) (cf. Section 6). In this paper, we present a survey on recent technologies developed for Big Data. We categorize and deeply compare them not only according to their usage, benefits, limits and features, but also according to their different layers such as Data Storage Layer, Data Processing Layer, Data Querying Layer, Data Access Layer and Management Layer. This helps to better understand the connections among various Big Data technologies and their functioning.

This paper is organized as follows. Section 2 defines Big Data and presents some of its applications. Section 3 identifies and discusses some technical challenges faced in dynamic Big Data environment. Section 4 presents Hadoop framework and compares

some main modules developed on top of it (e.g., data storage, integration, processing and interactive querying). Section 5 presents main Hadoop distributions: Cloudera, Hortonworks Data Platform (HDP), Amazon Elastic MapReduce, MapR, IBM InfoSphere BigInsights, GreenPlums Pivotal HD and Oracle Big Data appliance. Section 6 presents related works.

2. Background

2.1. Big Data definition

Unlike traditional data, the term Big Data refers to large growing data sets that include heterogeneous formats: structured, unstructured and semi-structured data. Big Data has a complex nature that require powerful technologies and advanced algorithms. So the traditional static Business Intelligence tools can no longer be efficient in the case of Big Data applications.

Most data scientists and experts define Big Data by the following three main characteristics (called the 3Vs) (Furht and Villanustre, 2016):

Volume: Large volumes of digital data are generated continuously from millions of devices and applications (ICTs, smartphones, products' codes, social networks, sensors, logs, etc.). According to McAfee et al. (2012), it is estimated that about 2.5 exabytes were generated each day in 2012. This amount is doubling every 40 months approximately. In 2013, the total digital data created, replicated, and consumed was estimated by the International Data Corporation (a company which publishes research reports) as 4.4 Zettabytes (ZB). It is doubling every 2 years. By 2015, digital data grew to 8 ZB (Rajaraman, 2016). According to IDC report, the volume of data will reach to 40 Zeta bytes by 2020 and increase of 400 times by now (Kune et al., 2016).

Velocity: Data are generated in a fast way and should be processed rapidly to extract useful information and relevant insights. For instance, Wallmart (an international discount retail chain) generates more than 2.5 PB of data every hour from its customers transactions. YouTube is another good example that illustrates the fast speed of Big Data.

Variety: Big Data are generated from distributed various sources and in multiple formats (e.g., videos, documents, comments, logs). Large data sets consist of structured and unstructured data, public or private, local or distant, shared or confidential, complete or incomplete, etc.

Emani et al. (2015) and Gandomi and Haider (2015) indicate that more *Vs* and other characteristics have been added by some actors to better define Big Data: *Vision* (a purpose), *Verification* (processed data conforme to some specifications), *Validation* (the purpose is fulfilled), *Value* (pertinent information can be extracted for many sectors), *Complexity* (it is difficult to organize and analyze Big data because of evolving data relationships) and *Immutability* (collected and stored Big data can be permanent if well managed).

2.2. Big Data applications

Here are some examples of Big Data applications:

Smart Grid case: it is crucial to manage in real time the national electronic power consumption and to monitor Smart grids operations. This is achieved through multiple connections among smart meters, sensors, control centers and other infrastructures. Big Data analytics helps to identify at-risk transformers and to detect abnormal behaviors of the connected devices. Grid Utilities can thus choose the best treatment or action. The real-time analysis of the generated Big Data allow to model incident scenarios. This enables to establish strategic preventive plans in order to decrease the corrective costs. In addition, Energy-forecasting analytics help to better manage power demand load, to plan resources, and hence to maximize prots (Stimmel, 2014).

E-health: connected health platforms are already used to personalize health services (e.g., CISCO solution) (Nambiar et al., 2013). Big Data is generated from different heterogeneous sources (e.g., laboratory and clinical data, patients symptoms uploaded from distant sensors, hospitals operations, pharmaceutical data). The advanced analysis of medical data sets has many beneficial applications. It enables to personalize health services (e.g., doctors can monitor online patients symptoms in order to adjust prescription); to adapt public health plans according to population symptoms, disease evolution and other parameters.It is also useful to optimize hospital operations and to decrease health cost expenditure.

Internet of Things (IoT): IoT (Chen et al., 2014b) represents one of the main markets of big data applications. Because of the high variety of objects, the applications of IoT are continuously evolving. Nowadays, there are various Big Data applications supporting for logistic enterprises. In fact, it is possible to track vehicles positions with sensors, wireless adapters, and GPS. Thus, such data driven applications enable companies not only to supervise and manage employees but also to optimize delivery routes. This is by exploiting and combining various information including past driving experience.Smart city is also a hot research area based on the application of IoT data.

Public utilities: Utilities such as water supply organizations are placing sensors in the pipelines to monitor flow of water in the complex water supply networks. It is reported in the Press that Bangalore Water Supply and Sewage Board is implementing a real-time monitoring system to detect leakages, illegal connections and remotely control valves to ensure equitable supply of water to different areas of the city. It helps tp reduce the need for valve operators and to timely identifying and fixing water pipes that are leaking.

Transportation and logistics: (Rajaraman, 2016) Many public road transport companies are using RFID (Radiofrequency Identification) and GPS to track busesand explore interesting data to improve there services... For instance, data collected about the number of passengers using the buses in different routes are used to optimize bus routes and the frequency of trips. various real-time system has been implemented not only to provide passengers with recommendations but also to offer valuable information on when to expect the next bus which will take him to the desired destination. Mining Big Data helps also to improve travelling business by predicting demand about public or private networks. For instance, in India that has one of the largest railway networks in the world, the total number of reserved seats issued every day is around 250,000 and reservation can be made 60 days in advance. Making predictions from such data is a complicated issue because it depends on several factors such as weekends, festivals, night train, starting or intermediate station. By using the machine learning algorithms, it is possible to mine and apply advanced analytics on past and new big data collection. In fact advanced analytics can ensure high accuracy of results regarding many issues.

Political services and government monitoring: Many government such as India and United States are mining data to monitor political trends and analyze population sentiments. There are many applications that combine many data sources: social network communications, personal interviews, and voter compositions. Such systems enable also to detect local issues in addition to national issues. Furthermore, governments may use Big Data systems to optimize the use of valuable resources and utilities. For instance, sensors can be placed in the pipelines of water supply chains to monitor water

flow in large networks. So it is possible for many countries to rely on real-time monitoring system to detect leakages, illegal connections and remotely control valves to ensure equitable supply of water to different areas of the city.

3. Big Data challenges

The mining of Big Data offers many attractive opportunities. However, researchers and professionals are facing several challenges when exploring Big Data sets and when extracting value and knowledge from such mines of information. The difficulties lye at different levels including: data capture, storage, searching, sharing, analysis, management and visualization. Furthermore, there are security and privacy issues especially in distributed data driven applications. Often, the deluge of information and distributed streams surpass our capability to harness. In fact, while the size of Big Data keeps increasing exponentially, the current technological capacity to handle and explore Big Data sets, is only in the relatively lower levels of petabytes, exabytes and zettabytes of data. In this section, we discuss in more details some technological issues still opened for research.

3.1. Big Data management

Data scientists are facing many challenges when dealing with Big Data. One challenge is how to collect, integrate and store, with less hardware and software requirements, tremendous data sets generated from distributed sources (Chen et al., 2014b; Najafabadi et al., 2015a). Another challenge is Big Data management. It is crucial to efficiently manage Big Data in order to facilitate the extraction of reliable insight and to optimize expenses. Indeed, a good data management is the foundation for Big Data analytics. Big Data management means to clean data for reliability, to aggregate data coming from different sources and to encode data for security and privacy. It means also to ensure efficient Big Data storage and a role-based access to multiple distributed endpoints. In other words, Big Data management goal is to ensure reliable data that is easily accessible, manageable, properly stored and secured.

3.2. Big Data cleaning

Those five steps (Cleaning, Aggregation, Encoding, Storage and Access) are not new and are known in the case of traditional data management. The challenge in Big Data is how to manage the complexity of Big Data nature (velocity, volume and variety) (Khan et al., 2014) and process it in a distributed environment with a mix of applications. In fact, for reliable analysis results, it is essential to verify the reliability of sources and data quality before engaging resources. However, data sources may contain noises, errors or incomplete data. The challenge is how to clean such huge data sets and how to decide about which data is reliable, which data is useful.

3.3. Big Data aggregation

Another challenge is to synchronize outside data sources and distributed Big Data plateforms (including applications, repositories, sensors, networks, etc.) with the internal infrastructures of an organization. Most of the time, it is not sufficient to analyze the data generated inside organizations. In order to extract valuable insight and knowledge, it is important to go a step further and to aggregate internal data with external data sources. External data could include third-party sources, information about market fluctuation, weather forecasting and traffic conditions, data from social networks, customers comments and citizen feedbacks. This can help, for instance, to maximize the strength of predictive models used for analytics.

3.4. Imbalanced systems capacities

An important issue is related to the computer architecture and capacity. Indeed, it is known that the CPU performance is doubling each 18 months following the Moore's Law, and the performance of disk drives is also doubling at the same rate. However, the I/O operations do not follow the same performance pattern. (e.i, random I/O speeds have improved moderately while sequential I/O speeds increase with density slowly) (Chen and Zhang, 2014). Consequently, this imbalanced system capacities may slow accessing data and affects the performance and the scalability of Big Data applications. From another angle, we can notice the various devices capacities over a network (e.i, sensors, disks, memories). This may slow down system performance.

3.5. Imbalanced Big Data

Another challenge is classifying imbalanced dataset. This issue has gained lots of attention in the last years. In fact, real-world applications may produce classes with different distributions. The first type of class that are under-presented with negligible number of instances (known as the minority or positive class). The second class that have an abundant number of instances (known as the majority or negative class). Identifying the minority classes is important in various fields such as medical diagnosis (Nahar et al., 2013), software defects detection (Park et al., 2013), Finances (Zhou, 2013), drug discovery (Zhou, 2013) or bioinformatics (Yu et al., 2013).

The classical learning techniques are not adapted to imbalanced data sets. This is because the model construction is based on global search measures without considering the number of instances. Indeed, global rules are usually privileged instead of specific rule so the minority class are neglected during the model construction. Thus, Standard learning techniques do not consider the difference between the number of samples belonging to different classes (del Río et al., 2014). However, the classes which are under-represented may constitute important cases to identify.

In practice, many problem domains have more than two classes with uneven distributions, such as protein fold classification and weld flaw classification (Wang and Yao, 2012). These multi-class imbalance problems pose new challenges that are not observed in two-class problems. In fact, dealing with multi-class tasks with different misclassification costs of classes is harder than dealing with two-class ones. To solve this problem, different methods have been developed and often categorized into two categories. The first one extends some binary classification techniques to make them applicable for multi-class classification problems, e.g., discriminant analysis, decision trees, k-nearest neighbors, Naive Bayes, neural networks, and support vector machines. The second category is known as Decomposition and Ensemble Methods (DEM). It consists of decomposing a multi-class classification problem into a set of binary classification problems that can be solved by Binary Classifiers (BCs), and then classifying a new observation by applying an aggregative strategy on the BCs' predictions (Zhou and Fujita, 2017; Zhou et al., 2017).

3.6. Big Data analytics

Big data brings big opportunities and transformative potential for various sectors; on the other hand, it also presents unprecedented challenges to harnessing such large increasing volumes of data. Advanced data analysis is required to understand the

relationships among features and explore data. For instance, data analysis enables an organization to extract valuable insight and monitor the patterns that may affect positively or negatively the business. Other data driven applications need also real-time analysis, like navigation, social networks,Finance, biomedicine, astronomy, intelligent transport systems. Thus, advanced algorithms and efficient methods of data mining are needed to get accurate results, to monitor the changes in various fields and to predict future observations. However, big data analysis is still challenging for many reasons: the complex nature of Big Data including the 5Vs, the need for scalability and performance to analyze such tremendous heterogenous data sets with real-time responsiveness (Wang et al., 2016; Tsai, 2016).

Nowadays, there are various analytical techniques including data mining, visualization, statistical analysis, and machine learning. Many studies tackle this area by either enhancing the used techniques, proposing new ones or testing the combination of various algorithms and technologies. Thus, Big Data pushed the development of systems architectures, the hardware as well as softwares. However, we still need analytical advancement to face Big Data challenges and stream processing. One of the issues is how to guarantee the timeliness of response when the volume of data is very large? In the following sub-sections, we explore examples of the difficulties encountered when applying current analytical solutions: Machine learning, deep learning, incremental approaches, granular computing.

3.7. Big Data machine learning

The objective of machine learning is to discover knowledge and make intelligent decisions. It is used for many real word applications such as recommendation engines, recognition systems, informatics and data mining, and autonomous control systems (Bishop, 2006). Generally, the field of Machine Learning (ML) is divided into three subdomains: supervised learning, unsupervised learning, and reinforcement learning. Reader may refer to Qiu et al. (2016) for more details about the ML types.

3.7.1. Data Stream learning

Current real-world applications such as sensors networks, credit card transactions, stock management, blog posts and network traffic produce tremendous datasets. Data mining methods are important to discover interesting patterns and to extract value hidden in such huge datasets and streams.

However, traditional data mining techniques such as association mining, clustering and classification lack of efficiency, scalability and accuracy when applied to such Big Data sets in a dynamic environment.

Because of the size, speed and variability of streams, it is not feasible to store them permanently then to analyze them. Thus researchers need to find new ways to optimize analytical techniques, to process data instances in very limited amount of time with limited resources (e.i, memory) and to produce in real-time accurate results.

Furthermore, variability of streams brings unpredictable changes (e.i, changing distribution of instances) in incoming data streams. This concept drift affects the accuracy of classification model trained from past instances. Therefore, several data mining methods were adapted to include drift detection techniques and to cope with changing environment. Classification and clustering are the most studied ones.

Experiments on data streams demonstrated that changes in underlying concept affects the performance of classifier model. Thus, improved analytical methods are needed to detect and adapt to the concept drifts (Jadhav and Deshpande, 2016).

As an example in the current unstable economic environment, enterprises need an efficient Financial Distress Predict (FDP) system. Such system is crucial to improve risk management and support banks in credit decisions. DFDP (Dynamic Financial Distress Prediction) became an important branch of FDP research (Sun et al., 2017). It improves corporate financial risk management. It focuses on how to update the FDP model dynamically when the new sample data batches gradually emerge and FDC (Financial Distress Concept drift) happens over time.

3.7.2. Deep learning

Nowadays, Deep learning constitutes an extremely active research field in machine learning and pattern recognition. It plays an important role in predictive analytics applications such as computer vision, speech recognition and natural language processing (Chen et al., 2014b).

Traditional machine-learning techniques and feature engineering algorithms, are limited in their ability to process natural data in their raw form (Razzak et al., 2017). On the contrary, Deep Learning is more powerful to resolve data analytical and learning problems found in huge data sets. In fact, it helps to automatically extracting complex data representations from large volumes of unsupervised and uncategorized raw data.

Moreover, because deep learning is based on hierarchical learning and extraction of different levels of complex data abstractions, it is suitable to simplify the analysis of large data volumes, semantic indexing, data tagging, information retrieval, and discriminative tasks such a classification and prediction (e.i, a feature extractor that transformed the raw data (such as the pixel values of an image)) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. However, in spite of those advantages, Big Data still presents significant challenges to deep learning (Najafabadi et al., 2015b):

- Huge volumes of Big Data: the training phase is not an easy task for Big Data learning in general and Deep learning specifically. This is because the iterative computations of the learning algorithms are very difficult to be parallelized. Thus, there is still a need to create efficient and scalable parallel algorithms to improve training stage for Deep models
- Heterogeneity: high volumes of data imposes a great challenge for deep learning. It means to handle large number of examples (inputs), large varieties of class types (outputs), and very high dimensionality (attributes). Thus analytical solutions have to deal with running-time complexity and model complexity. In addition to that, such large data volumes make it not feasible to train a deep learning algorithm with a central processor and storage.
- Noisy labels, and non-stationary distribution: because of the disparate origins and heterogenous sources of Big Data, analytical researchers are still facing other challenges such as data incompleteness, missing labeles and noisy labels.
- High velocity: as we know, data are generating at extremely high speed and should be processed in a real-time. In addition to the high velocity, data are often non-stationary and presents a changing distribution over time.

Because of those cited issues, Deep Learning solutions still lack of maturity and need additional extensive research to optimize the analytical results. In summary, research future works should consider how to improve Deep Learning algorithms in order to tackle streaming data analysis, high dimensionality, models scalability. Studies have also to improve formulation of data abstractions, distributed computing, semantic indexing, data tagging, information

retrieval, criteria selection for extracting good data representations, and domain adaptation.

3.7.3. Incremental and ensemble learning

Incremental learning and ensemble learning constitute two learning dynamic strategies. They are fundamental methods in learning from big stream data with concept drift (Zang et al., 2014).

Incremental and ensemble learning are frequently applied to data streams and big data. They tackle various difficulties such as addressing data availability, limited resources. They are adapted to many applications such as stock trend prediction and user profiling. Applying incremental learning enable to produce faster classification or forecasting times while receiving new data.

Many traditional machine learning algorithms inherently support incremental learning, other algorithms can be adapted to facilitate this. Examples of incremental algorithms include decisions trees(IDE4, ID5R), decision rules, neuronal networks, neuronal Gaussian RBF networks(Learn++, ARTMAP) or the incremental SVM.

When comparing those types of algorithms, it is noticed that incremental algorithms are faster. However, ensemble algorithms are more flexible and can better adapt to concept drift. Moreover, not all classification algorithms can be used in incremental learning, but almost every classification algorithms can be used in ensemble algorithms (Zang et al., 2014). Thus, it is recommended to use incremental algorithm in the absence of concept drift or if the concept drift is smooth. On the contrary, ensemble algorithms are recommended to ensure accuracy in the case of huge concept drift or abrupt concept drift. Furthermore, if we have to deal with relatively simple data-stream or a high level of real-time processing, incremental learning is more suitable. However, ensemble learning constitute a better choice in case of complicated or unknown distribution of data streams.

3.7.4. Granular computing

Granular Computing (GrC) (Skowron et al., 2016) is not new, but it has recently became more popular for its use in various Big Data domains. It shows many advantages in the case of intelligent data analysis, pattern recognition, machine learning and uncertain reasoning for huge size of data sets. Indeed, GrC play important roles in the design of decision making models while ensuring acceptable performance.

Technically, GrC constitute a general computation theory based on granules such as classes, clusters, subsets, groups and intervals. Thus, it may be used to build an efficient computational model for complex Big Data applications such as data mining, document analysis, financial gaming, organization and retrieval of huge data bases of multimedia, medical data, remote sensing, biometrics.

Distributed systems require to support different users in understanding big data at different granularity levels. There is also a need to analyze data and present results with different viewpoints. to fulfil those requirements, GrC provides powerful tools for multiple granularity and multiple viewing of data analysis. This enables to better understand and analyze the complexity of various big data sets. Moreover, GrC techniques can serve as effective processing tools for real world intelligent systems and dynamic environment like FDS (Fuzzy Dynamic Decision Systems).GrC enables to tackle the complex issue of evolving attributes and objects in streams over time. Indeed, GrC plays an important role to find simple approximate solution while ensuring cost effectiveness and improved description. For instance, the integration of GrC and computational intelligence has become a hot area of research to develop efficient decision-making models dedicated to resolve complex problems of Big Data.

GrC can be implemented via various technologies such as: fuzzy sets, rough sets, random sets, etc. Fuzzy set techniques provide a novel way to investigate and represent the relation between a set and its members. This is by considering the continuum degree of belonging, namely membership functions (similar to human recognition). Fuzzy information granulation is about a pool of fuzzy granules derived by granulating objects, rather than a single fuzzy granule.

In general, Fuzzy sets have been applied to various areas (Wang et al., 2017) such as control systems, pattern recognition and machine learning. Fuzzy sets enable us to represent and process information at distinct levels of information granularity. More specifically, Fuzzy set techniques play important role in all phases of Big Data value chain: first in handling the uncertainties of raw data, then annotating data and finally preparing specific granular representation of data for artificial intelligent algorithms.

For instance, Huang et al. (2017a) has proved that their proposed model outperform the static algorithms and related incremental algorithms like Zhang's method (Zhang et al., 2012). Indeed, the model ensures a better efficiency and optimizes computation. For that, the both solutions of Huang et al. (2017a) and Huang et al. (2017b) are based on those fundamentals: first a matrix method is used to construct and calculate rough approximations. Second, an incremental method is used to dynamically update rough set approximation.

We notice from the literature that matrices are more and more used for rough data analysis and approximations. This is because matrix structure supports the intuitive description of huge data sets, maintainability and optimized calculations. In fact, to extract knowledge from huge evolving coming streams, the model updates and do computations on just small relation matrices (submatrices) instead of updating the whole relation matrix. Unlike those approaches based on matrix operation, (Luo et al., 2016a) model used probabilistic rough set model with the incremental approach. Their goal is to model imprecise data with tolerance of decision errors in terms of conditional probability and probabilistic parameters.

Consequently, GrC techniques can improve the current big data techniques while tackling big data challenges (e.i, challenges raised by the 5Vs, by pre-processing data or by reconstructing the problem at a certain granular level). However, it is worth noticing that the role of GrC and fuzzy set techniques is to provide a

Table 1
Comparaison between HDFS and Hbase features.

Properties	HDFS	HBase
System	HDFS is a distributed file system appropriate to store large files.	HBase is a distributed non relational database built on the top of HDFS.
Query and search performance	HDFS is not a general purpose file system. It does not provide fast record lookup in files.	It enables fast record lookups (and updates) for large tables.
Storage	HDFS stores large files (gigabytes to terabytes in size) across Hadoop servers.	HBase internally puts the data in indexed "StoreFiles" that exist on HDFS for high-speed lookups.
processing	HDFS is suitable for High Latency operations batch processing.	HBase is built for Low Latency operations.
Access	Data is primarily accessed through MapRe- duce.	HBase provides access to single rows from billions of records.
Input-ouput operations	HDFS is designed for batch processing and hence does not support random reads/writes operations.	HBase enables reads/writes operations. Data is accessed through shell commands, client APIs in Java, REST, Avro or Thrift.

methodology for knowledge abstraction (granulation) and knowledge representation. This is different from the role of other techniques used for big data, like deep learning (Wang et al., 2017).

4. Big Data and Hadoop ecosystem

4.1. Hadoop capabilities

Apache Hadoop is a well known Big Data technology that has an important supporting community. It has been designed to avoid the low performance and the complexity encountered when processing and analyzing Big Data using traditional technologies. One main advantage of Hadoop is its capacity to rapidly process large data sets, thanks to its parallel clusters and distributed file system. In fact, unlike traditional technologies, Hadoop do not copy in memory the whole distant data to execute computations. Instead, Hadoop executes tasks where data are stored. Thus, Hadoop relieves network and servers from a considerable communication load (Usha and Aps, 2014). For instance, it takes just few seconds on Hadoop to query terabytes of data instead of 20 min or more on classic SIEM. Another advantage of Hadoop is its ability to run programs while ensuring fault-tolerance, usually encountered in distributed environment. To guarantee that, it prevent data loss by replicating data on servers.

The power of Hadoop platform is based on two main subcomponents: the Hadoop Distributed File System (HDFS) and the MapReduce framework (explained in the following sections). In addition, users can add modules on top of Hadoop as needed according to their objectives as well as their application requirements (e.g., capacity, performances, reliability, scalability, security). In fact, Hadoop community has contributed to enrich its ecosystem with several open source modules. In parallel, IT venders provide special enterprise hardening features delivered within Hadoop distributions.

4.2. Data Storage Layer: HDFS and HBase

To store data, Hadoop relies on both its file system HDFS and a non relational database called Apache HBase.

4.2.1. Hadoop Distributed File System (HDFS)

HDFS (White, 2012) is a data storage system. It supports up to hundreds of nodes in a cluster and provides a cost-effective and reliable storage capability. It can handle both structured and unstructured data and hold huge volumes (i.e., stored files can be bigger than a terabyte). However, users must be aware that HDFS do not constitute a general purpose file system. This is because HDFS was designed for high-latency operations batch processing. In addition, it does not provide fast record lookup in files. HDFS main advantage is its portability across heterogeneous hardware and software platforms. In addition, HDFS helps to reduce network congestion and increase system performance by moving computations near to data storage. It ensures also data replication for fault-tolerance. Those features explain its wide adoption.

HDFS (Mall et al., 2016) is based on master–slave architecture. It distributes large data across the cluster. In fact, the cluster has a unique master (NameNode) that manages file system operations and many slaves (DataNodes) that manage and coordinate data storage on individual compute nodes. To provide data availability, Hadoop lies on data replication.

4.2.2. HBase

HBase (Prasad and Agarwal, 2016) is a distributed non relational database. It is an open source project that is built on top of HDFS. It is designed for low-latency operations. Hbase is based on column-oriented key/value data model. It has the potential to support high table-update rates and to scale out horizontally in distributed clusters. HBase provides a flexible structured hosting for very large tables in a BigTable-like format.

Tables store data logically in rows and columns (Coronel and Morris, 2016). The benefit of such tables is that they can handle billions of rows and millions of columns. HBase allows many attributes to be grouped into column families so that the elements of a column family are all stored together. This approach is different from a row-oriented relational database, where all columns of a row are stored together. Thus, HBase is more flexible than relational databases. Instead, HBase has the advantage of allowing users to introduce updates to better handle changing applications' requirements. However, HBase has the limitation of not supporting a structured query language like SQL.

Tables of HBase are called HStore and each Hstore has one or more Map-Files stored in HDFS. Each table must have a defined schema with a Primary Key that is used to access the Table. The row is identified by table name and start key while columns may have several versions for the same row key.

Hbase provides many features such us real-time queries, natural language search, consistent access to Big Data sources, linear and modular scalability, automatic and configurable sharding of tables (Dimiduk et al., 2013). It is included in many Big Data solutions and data driven websites such as Facebook's Messaging Platform. HBase includes Zookeeper for coordination services and runs a Zookeeper instance by default. Similarly to HDFS, HBase (Maheswari and Sivagami, 2016) has a MasterNode that manages the cluster and slaves that store parts of the tables and perform operations on data. Table 1 summarizes the differences between HDFS and HBase.

4.3. Data Processing Layer

MapReduce and YARN constitute two options to carry out data processing on Hadoop. They are designed to manage job scheduling, resources and the cluster. It is worth noticing that YARN is more generic than MapReduce.

4.3.1. MapReduce programming model

MapReduce (Lydia and Swarup, 2015) is a framework composed of a programming model and its implementation. It is one of the first essential steps for the new generation of Big Data management and analytics tools. MapReduce has an interesting benefit for Big data applications. In fact, it simplifies the processing of massive volumes of data through its efficient and cost-effective mechanisms. It enables to write programs that can support parallel processing.

In fact, MapReduce programming model uses two subsequent functions that handle data computations: the Map function and the Reduce function.

More precisely, a MapReduce program relies on the following operations:

1. First, the Map function divides the input data (e.g., long text file) into independent data partitions that constitute key-value pairs.
2. Then, the MapReduce framework sent all the key-value pairs into the Mapper that processes each of them individually, throughout several parallel map tasks across the cluster. Each data partition is assigned to a unique compute node. The Mapper outputs one or more intermediate key-value pairs. At this stage, the framework is charged to collect all the intermediate key-value pairs, to sort and group them by key. So the result is many keys with a list of all the associated values.

3. Next, the Reduce function is used to process the intermediate output data. For each unique key, the Reduce function aggregates the values associated to the key according to a predefined program (i.e., filtering, summarizing, sorting, hashing, taking average or finding the maximum). After that, it produces one or more output key-value pairs.
4. Finally, the MapReduce framework store all the output Key-value pairs in an output file.

Within MapReduce paradigm, the NameNode runs a JobTracker instance in order to schedule the different jobs and distribute tasks over the slave nodes. To insure execution reliability, the JobTracker monitors the status of the slave nodes and re-assigns tasks when they failed. Each of the slave nodes runs a TaskTracker instance for the assigned tasks. A TaskTracker instance executes the tasks as specified by the JobTracker and monitors their execution. Each TaskTracker can use multiple JVMs (Java Virtual Machines) to execute several maps or reduce tasks in parallel.

Usually, a Hadoop cluster is composed of a client server, multiple DataNodes and two types of NameNodes (primary and secondary). The role of the client server is first to load data and then to submit MapReduce jobs to the NameNode. The primary Master NameNode is dedicated to coordinate and to manage storage and computations. On the other hand, the secondary master NameNode handles data replication and availability. A unique physical server may handle the three roles (client, master and slaves) in small clusters (less than 40 nodes). However, in medium and large clusters, each role should be assigned to a single server machine.

4.3.2. YARN

YARN is more generic than MapReduce. It provides a better scalability, enhanced parallelism and advanced resource management in comparison to MapReduce. It offers operating system functions for Big Data analytical applications. Hadoop architecture has been changed to incorporate YARN Ressource Manager. In general, YARN works on the top of HDFS. This position enables the parallel execution of multiple applications. It allows also handling both batch processing and real-time interactive processing. YARN is compatible with Application Programming Interface (API) of MapReduce. In fact, users have just to recompile MapReduce jobs in order to run them on YARN.

Unlike MapReduce, YARN (White, 2012) enhances efficiency by splitting the two main functionalities of the JobTracker into two separate daemons: (1) ResourceManager (RM) that allocates and manages resources across the cluster. (2) Application Master (AM) framework with a library. It is designed to schedule tasks, to match them with TaskTrackers and to monitor their progress. AM negotiates also resources with RM and Node Manager. For instance, it ensures task bookkeeping, maintains counters, restarts failed or slow tasks. Thus, Job scheduling entity ensures lifecycle management of all applications executed in a cluster.

4.3.3. Cascading: a MapReduce framework for complex flows

Cascading framework (Mazumder, 2016) is a rich Java API that provides many components for fast and cost-effective Big Data application development, testing and integration. Cascading has interesting benefits.It allows managing advanced queries and handling complex workflows on Hadoop clusters. It supports scalability, portability, integration and test-driven development.

This API adds an abstraction level on the top of Hadoop to simplify complex queries through a cascading concept. In fact, the loaded data are processed and split by a series of functions to get multiple streams called flows. Those flows form acyclic-directed graphs and can be joined together as needed.

The pipe assembly defines the flow to run between the data sources (Source Taps) and the output data (Sink Taps) that are connected to the pipe. A pipe assembly may contain one or more Tuples of a given size.

A cascading flow is written in Java and transformed during the execution into classic MapReduce jobs. Flows are executed on Hadoop clusters and are based on the following process:

A Flow instance is a workflow that first reads the input data from one or many Source Taps, and then processes them by executing a collection of parallel or sequential operations as defined by the pipe assembly. Then, it writes the output data into one or several Sink Taps.

A Tuple represents a set of values (like a database record of SQL table) that can be indexed with Fields and can be stored directly into any Hadoop File format as key/value pair. A tuple should have comparable types in order to facilitate Tuple comparison. Many extensions were added to the Cascading framework to enhance its capabilities, including (Nathan, 2013):

- *Pattern*: used to build predictive big data applications. It provides many machine learning algorithms and enables translating Predictive Model Markup Language (PMML) documents into applications on Hadoop.
- *Scalding*: used as a dynamic programming language to solve functional problems. It is based on Scala language with a simple syntax. This extension is built and maintained by Twitter.
- *Cascalog*: allows to develop application using Java or Clojure (a dynamic programming language based on Lisp dialect). It supports Ad-hoc queries, by running a series of multiple MapReduce jobs to analyze different sources (HDFS, databases and local data). It provides higher level of abstraction than Hive or Pig (cf. 4.4).
- *Lingual*: provides an ANSI-SQL interface for Apache Hadoop and supports a rapid migration of data and workloads to and from Hadoop. Through Lingual, it is easier to integrate the existing Business Intelligence tools and other applications.

4.4. Data Querying Layer: Pig, JAQL and Hive

Apache Pig (Mazumder, 2016) is an open source framework that generates a high level scripting language called Pig Latin. It reduces MapReduce complexity by supporting parallel execution of

Table 2
Hive, Pig and JAQL features.

Properties	Data querying tools		
	Hive	Pig	Jaql
Language	HiveQL (SQL-like)	Pig Latin (script-based language)	JAQL
Type of language	Declarative (SQL dialect)	Data flow	Data flow
Data structures	Suited for structured data	Scalar and complex data types	File-based data
Schema	It has tables' metadata stored in the database	Schema is optionally defined at runtime	Schema is optional
Data Access	JDBC, ODBC	PigServer	Jaql web server
Developer	Facebook	Yahoo	IBM

Table 3
A comparaison between Flume and Chukwa.

Properties	Projects	
	Chukwa	Flume
Real-time	It acquires data for periodic real-time analysis (within minutes)	It focuses on continuous real-time analysis (within seconds)
Architecture	batch system	Continuous stream processing system
Manageability	It distributes information about data flows broadly among its services	It maintains a central list of ongoing data flows, stored redundantly using Zookeeper
Reliability	The agents on each machine are responsible for deciding what data to send. Chukwa uses an end-to-end delivery model that can leverage local on-disk log files for reliability	Robust/Fault tolerant with tunable reliability mechanisms and failover and recovery mechanisms. Flume adopts a hop-by-hop model

MapReduce jobs and workflows on Hadoop. Through its interactive environment, Pig like Hive, simplifies exploring and processing in parallel massive data sets using HDFS (e.g., complex data flow for ETL, various data analysis). Pig allows also interaction with external programs like shell scripts, binaries, and other programming languages. Pig has its own data model called Map Data (a map is a set of key-value pairs). (Krishnan, 2013).

Pig Latin has many advantages.It is based on an intuitive syntax to support an easy development of MapReduce jobs and workflows (simple or nested flows). It reduces the development time while supporting parallelism (Sakr, 2016d). Thus, users can rely on Pig Latin language and several operators to upload and process data. Pig Latin is an alternative to Java programming language with scripts similar to a Directed Acyclic Graph (DAG). In fact, in such DAC, operators that process data constitute nodes while data flows are presented by edges (Loganathan et al., 2014). On the contrary to SQL, Pig does not require a schema and can process semi-structured and unstructured data. It supports more data formats than Hive. Pig can run on both the local environment in a single JVM and the distributed environment on a Hadoop cluster.

JAQL (Beyer et al., 2011) is a declarative language on top of Hadoop that provides a query language and supports massive data processing. It converts high level queries into MapReduce jobs. It was designed to query semi-structured data based on JSONs (JavaScript Object Notation) format. However, it can be used to query other data formats as well as many data types (e.g., XML, comma-separated values (CSV) data, flat files). So, JAQL like Pig does not require a data schema. JAQL provides several in-built functions, core operators and I/O adapters. Such features ensure data processing, storing, translating and data converting into JSON format.

Apache Hive (Shaw et al., 2016) is a data warehouse system designed to simplify the use of Apache Hadoop. In contrast to MapReduce, that manages data within files via HDFS, Hive enables to represent data in a structured database that is more familiar for users. In fact, Hive's data model is mainly based on tables. Such tables represent HDFS directories and are divided into partitions. Each partition is then divided into buckets.

Moreover, Hive provides a SQL-like language called HiveQL (Sakr, 2016a) that enable users to access and manipulate Hadoop-based data stored in HDFS or HBase. Therefore, Hive is suitable for many business applications.

Hive (Bansal et al., 2016) is not suitable for real-time transactions. In fact, it is based on a low-latency operations. Like Hadoop, Hive is designed for large scale processing so even small jobs may take minutes. Indeed, HiveQL transparently converts queries (e.g., ad hoc queries, joins, and summarization) into MapReduce jobs that are processed as batch tasks.

Hive enables also plugging in traditional mappers and reducers when it is not feasible or inefficient to express them in HiveQL.

Unlike most SQL having schema-on-write feature, Hive (Loganathan et al., 2014) has schema-on-read and supports multiple schemas, which defers the application of a schema until you try to read the data. Though the benefit here is that it loads faster, the drawback is that the queries are relatively slower. Hive lacks full SQL support and does not provide row-level inserts, updates or delete. This is where HBase worth investing. Table 2 summarizes and compares some characteristics of Hive, Pig, and JAQL.

4.5. Data Access Layer

4.5.1. Data Ingestion: Sqoop, Flume and Chukwa

Apach Sqoop (Vohra, 2016) is an open source software.It provides a command-line interface (CLI) that ensures an efficient transfer of bulk data between Apache Hadoop and structured datastores (such as relational databases, enterprise data warehouses and NoSQL databases). Sqoop offers many advantages. For instance,it provides fast performance, fault tolerance and optimal system utilization to reduce processing loads to external systems. The transformation of the imported data is done using MapReduce or any other high-level language like Pig, Hive or JAQL (Jain, 2013). It allows easy integration with HBase, Hive and Oozie. When Sqoop imports data from HDFS, the output will be in multiple files. These files may be delimited text files, binary Avro or SequenceFiles containing serialized data. The process of Sqoop Export will read a set of delimited text files from HDFS in parallel, parse them into records, and insert them as new rows in a target database table.

Flume (Hoffman, 2015) is designed to collect, aggregate and transfer data from external machines to HDFS. It has a simple flexible architecture and handles streaming of data flows. Flume is based on a simple extensible data model to handle massive distributed data sources. Flume provides various features including fault-tolerance, tunable reliability mechanism as well as failure-recovery service. Though that Flume complements well Hadoop, it is an independent component that can work on other platforms. It is known for its capacity to run various processes on a single machine. By using Flume, users can stream data from various and high volume sources (like Avro RPC source and syslog) into sinks (such as HDFS and HBase) for real-time analysis (Hoffman, 2013). In addition, Flume provides a query processing engine that can transform each new data batch before it is channeled to the specified sink.

Chukwa (Shireesha and Bhutada, 2016) is a data collection system built on the top of Hadoop. Chukwa's goal is to monitor large distributed systems. It uses HDFS to collect data from various data providers, and MapReduce to analyze the collected data. It inherits Hadoop's scalability and robustness. It provides an interface to display, monitor and analyze results

Chukwa offers a flexible and powerful platform for Big Data. It enables analysts to collect and analyze Big Data sets as well as to monitor and display results. To ensure flexibility, Chukwa is structured as a pipeline of collection, processing stages as well as defined interfaces between stages.

Chukwa is based on four main components: first, it relies on data agents on each machine to emit data. Next, collectors are

Table 4
A comparaison between Strom and Spark.

Properties	Projects	
	Sprak	Storm
Foundation	UC Berkeley	BackType, Twitter
Type	Open source	Open source
Implementation language	Scala	Coljure
Supported languages	Java, Python, R, Scala	Any
Execution model	Batch, streaming	Streaming
Latency	Spark has latency of just few seconds (Deponding on batch size)	Strom has latecy of sub-seconds
Management style	Spark writes data to the storage and requires stateful computations	Storm rools on it own or uses trident and requires stateless computation
Fault Tolerance	Support only exactly once processing mode	Supports 'exaclty once', 'at least once' and 'at most once' processing mode.
Stream sources	HDFS	Spout
Stream Computation	Windows Operations	Bolts
Stream Primitives	Dstream	Tuple
Provisioning	Basic monitoring using ganglia	Apache Ambari
Resources Manger Integration	Messos and Yarn	Mesos
Hadoop Distr	HDP, CDH, MapR	HDP

used to collect data from agents and write it to a stable storage. MapReduce jobs are used to parse and archive data. Users can rely on a friendly interface (HICC) to display results and data. It has a web-portal style (De carvalho et al., 2013; Rabkin and Katz, 2010).

While Flume and Chukwa share similar goals and features, they have some differences resumed in Table 3.

4.5.2. Data streaming: storm and spark

Storm (Mazumder, 2016) is an open source distributed system that has the advantage of handling real time data processing in contrast with Hadoop, which is designed for batch processing.

In comparison to flume, Storm shows better efficiency in implementing complex processing requirements by relying on the Trident API.

Storm is based on a topology composed of a complete network of spouts, bolts, and streams. A spout is a source of streams. A bolt is used to process input streams in order to produce output streams. Hence, Storm is suitable to perform transformations on streams using "spouts" and "bolts".

The ISpout interface of Storm can potentially support any incoming data. In fact, by using Storm (Mazumder, 2016), users can ingest data from various real time synchronous and asynchronous systems (e.i, JMS, Kafka, Shell and Twitter). Based on Bolts,Storm enables to write data to any output system. Storm provides the IBolt interface that supports any type of output systems like JDBC (to store data to any relational database), Sequence Files, Hadoop components like HDFS, Hive, HBase, and other messaging system.

The storm cluster and Hadoop cluster are apparently similar. However, in Storm, it is possible to run different topologies for different storm tasks. Instead, in Hadoop platform, the only option consists in implementing Map Reduce jobs for the corresponding applications. One main difference between Map Reduce jobs and topologies is the following. MapReduce job stops but a topology continues to process messages either all the time or until user terminate (Acharjya and Ahmed, 2016a).

Storm is an easy-to-use, rapid, scalable and fault-tolerant system, if one or more processes fails, Storm will automatically restart it. If the process fails repeatedly, Storm will reroute it to another machine and restart it there. It can be used for many cases (Lyko et al., 2016) such us real-time analytics, online machine learning, continuous computation and distributed RPC.

Storm (Lublinsky et al., 2013) is used to prepare results that can then be analyzed by other Hadoop tools. It can process million tuples per second. Like MapReduce, Storm provides a simplified programming model, which hides the complexity of developing distributed applications.

Apache Spark (Acharjya and Ahmed, 2016a) is an open source distributed processing framework that was created at the UC Berkeley AMPLab. Spark is like Hadoop but it is based on in-memory system to improve performance. It is a recognized analytics platform that ensures a fast, easy-to-use and flexible computing. Spark handles complex analysis on large data sets. Indeed, Spark runs programs up to 100x faster than Hive and Apache Hadoop via MapReduce in-memory system. Spark is based on the Apache Hive codebase. In order to improve system performance, Spark swap out the physical execution engine of Hive. In addition, Spark offers APIs to support a fast application development in various languages including Java, Python and Scala (Karau, 2013). Spark is able to work with all files storage systems that are supported by Hadoop.

Spark's data model (Sakr, 2016b) is based on the Resilient Distributed Dataset (RDD) abstraction. RDDs constitutes a read-only collection of objects stored in system memory across multiple machines. Such objects are available without requiring a disk access. Furthermore, they can be rebuilt if a partition is lost.

The Spark project consists of multiple components for task scheduling, memory management, fault recovery, interacting with storage systems, etc. The components are listed as follows:

- *Spark SQL (Sakr, 2016b)*: One important feature of Spark SQL is that it unifies the two abstractions: relational tables and RDD. So programmers can easily mix SQL commands to query external data sets with complex analytics. Concretely, users can run queries over both imported data from external sources (like Parquet files an Hive Tables) and data stored in existing RDDs. In addition, Spark SQL allows writing RDDs out to Hive tables or Parquet files. It facilitates fast parallel processing of data queries over large distributed data sets for this purpose. It uses a query languages called HiveQL. For a fast application development, Spark has developed the Catalyst framework. This one enable users via Spark SQL to rapidly add new optimizations.
- *Spark streaming (Azarmi, 2016b)*: Spark Streaming is another component that provides automatic parallelization, as well as scalable and fault-tolerant streaming processing. It enables users to stream tasks by writing batch like processes in Java and Scala. It is possible to integrate batch jobs and interactive queries. It runs each streaming computation as a series of short batch jobs on in-memory data stored in RDDs.

Table 5
A Comparison between Mahout and R.

Properties	Analytical Tools	
	Apache Mahout	R
Type	Open source	Open source
Programming language	Java	R language
Arichitecture	Mostly MapReduce, porting to spark	In-memory system
Supported Platform	All Hadoop distributions and other platforms	Hadoop Cloudera Hortonworks Oracle
Features	– Its data model is based on Resilient Distributed Datasets (RDD's). – APIs for rapid application development). – Support SQL, HiveQL and Scala through Spark-SQL. – Efficient query execution by Catalyst framework. – High level tools to interact with data. – Efficient query execution by Catalyst framework. – High level tools to interact with data.	– Programming language. – Libraries with optimized algorithm for machine learning algorithms and graphs.
Key Benefits	– New users can get started with common use cases quickly. – It translates machine learning tasks expressed in Java into MapReduce jobs.	– Limited performance in case of very large data sets (One-node memory). – Supports statistics and machine learning algorithm. – Flexibility to develop programs. – Package for more options.

- *MLlib (*Landset et al., 2015*):* MLlib is a distributed machine learning framework built on top of Spark. For performance, MLlib provides various optimized machine learning algorithms such us classification, regression, clustering, and collaborative filtering. Like Mahout, MLlib is useful for machine learning categories. They offer algorithms for topic modeling and frequent pattern mining. Mlib supports also regression Models. However, Mahout does not support such model. MLlib is relatively young in comparison to Mahout.
- *GraphX, Wendell2014*: GraphX constitutes a library for manipulating graphs and executing graph-parallel computations. Like Spark Streaming and Spark SQL, GraphX extends the features of Spark RDD API. Thus users can create a directed graph with arbitrary properties attached to each vertex and edge. GraphX offers different operators that support graphs manipulation (e.g., subgraph and mapVertices). It provides also a library of graph algorithms (e.g., PageRank and triangle counting).

Table 4 summarizes and compares some characteristics of Storm and Spark.

4.5.3. Storage Management: HCatalog

Apache HCatalog (Wadkar and Siddalingaiah, 2014b) provides a table and storage management service for Hadoop users. It enables interoperability across data processing tools (like Pig, Hive and MapReduce). This is achieved through a shared schema and data type mechanisms. It provides an interface to simplify read and write data operations for any data format (e.g., RCFile, CSV, JSON and SequenceFiles formats) for which a Hive SerDe (serlializer-deserializer) can be written. For that, The system administrator provides the InputFormat, OutputFormat and the SerDe.

The abstracted table of HCatalog provides a relational view of data in HDFS and allows to view disparate data formats in a tabular format. So users do not have to know where and how data is stored. Furthermore, HCatalog supports users with other services. It notifies data availability and provides a REST interface to permit access to Hive Data Definition Language(DDL) operations (Wadkar and Siddalingaiah, 2014b). It also provides a notification service that notifies workflow tools (like Oozie) when new data becomes available in the warehouse.

4.6. Data analytics

Apache Mahout (Mazumder, 2016) is an open source machine learning software library. Mahout can be added on top of Hadoop to execute algorithms via MapReduce. It is designed to work also on other platforms.

Mahout (Dinsmore, 2016) is essentially a set of Java libraries.It has the benefit of ensuring scalable and efficient implementation of large scale machine learning applications and algorithms over large data sets. Indeed, Mahout library provides analytical capabilities and multiple optimized algorithms. For instance, it offers libraries for clustering (like K-means, fuzzy K-means, Mean Shift), classification, collaborative filtering (for predictions and comparisons), frequent pattern mining and text mining (for scanning text and assigning contextual data). Additional tools include topic modeling, dimensionality reduction, text vectorization, similarity measures, a math library, and more. The various companies those who have implemented scalable machine learning algorithms are Google, IBM, Amazon, Yahoo, Twitter and Facebook (Acharjya and Ahmed, 2016a).

As confirmed by Hsieh et al. (2016) and Acharjya and Ahmed (2016a), by integrating Mahout, users do not have to worry about algorithms development. Instead, they can concentrate on their main analytical problem and choose the appropriate analytical models for their applications.

Like Apache Hive (which provides an SQL-like interface to query data in Hadoop distributed file system), Mahout translates machine learning tasks expressed in Java into MapReduce jobs (Manoochehri, 2013).

R (Team, 2000) is a programming language for statistical computing, machine learning and graphics. R is free, open-source software distributed and maintained by the R-project that relies on a community of users, developers and contributors. R programming language includes a well-developed, simple and effective functionalities, including conditionals, loops, user-defined recursive functions and input and output facilities. Many Big Data distributions (like Cloudera, Hortonworks and Oracle) use R to performe analytics.

One drawbacks of R is its limited capacity to handle extremely large datasets because of the one node memory limitations. In fact, R like other high-level languages leads to memory overload because it is based on temporary copies instead of referencing existing objects. Furthermore, R programs are executed in a single thread, and the data must be stored in RAM. Thus, data structures should be no larger than 10–20 percent of a computer's available RAM.

To support a scalable Big Data analysis and resolve memory limitations, R has developed several packages such us *ff* package (offers file-based access to data sets that are too large to be loaded into memory, along with a number of higher-level functions), *bigmemory* Package (memory parallel analysis and data mining of massive data sets), *snow* Package (supports simple parallel computing) and *Teradata Aster R* which runs on the Teradata Aster

Discovery Platform (Brown, 2014), it aims to facilitate the distribution of data analysis over a cluster of machines and to overcome one-node memory limitations in R applications. *pdDR* (programming with Big Data in R) project (Raim, 2013) is another solution that aims to solve this problem, it enables high-level distributed data parallelism in R.

R provides a more complete set of classification models (regarding the types and depth of algorithms) in comparison to Mahout (Ames et al., 2013). However, R is not a rapid solution when compared to other environment because of its object-oriented programming that case memory management problems. Indeed, it may be more practical to use Mahout, Spark, SAS or other frameworks to ensure a better performance of extensive computations (Ames et al., 2013).

Ricardo is part of the eXtreme Analytics Platform (XAP) project of IBM Almaden Research Center designed to handle deep analytics problems. It combines the features of Hadoop with those of R as two integrated partners and components. In fact, Ricardo handles many types of advanced statistical analysis through R functionalities (like K-means, clustering, time-series, SVM classification). It leverages also the parallelism of Hadoop DMS.

At the same time, Ricardo provides large-scale data management capabilities through Hadoop. It supports also Jaql that is a high-level query language. With such combination Ricardo enables R to submit aggregation-processing queries (written in Jaql) to Hadoop. In fact, Ricardo decomposes data analysis algorithms. Small data parts are executed by R while large data parts are executed by Hadoopp/Jaql DMS. This technique minimizes data transfer across the system and ensures system performance. This is for simple trading, but to support complex trading algorithms, multiple iterations are performed over data sets with trading between R and Hadoop at each iteration. Experiments showed that Ricardo improves R's performance and facilitates operations such us data exploration, model building, model evaluation over massive data sets. Table 5 summarizes and compares some characteristics of Mahout and R.

4.7. Management Layer

4.7.1. Coordination and Workflow: Zookeeper, Avro and Oozie

Zookeeper (Lublinsky et al., 2013) is an open source service designed to coordinate applications and clusters in Hadoop environment. It provides several benefits. For instance, Zookeeper supports high performance and data availability. It simplifies also distributed programming and ensures reliable distributed storage. It is implemented in Java and provides APIs for Java and C-based programs. Zookeeper is a distributed application based on a client–server architecture. Zookeeper's server can run across several clusters. Zookeeper has a file system structure that mirrors classic file system tree architectures. Through its simple interface, Zookeeper enables also to implement fast, scalable and reliable cluster coordination services for distributed systems. For instance, it provides the configuration management service that allows a distributed setup, the naming service to find machines within large cluster, the replicated synchronization service to protect data and nodes from lost, the locking service that enables a serialized access to a shared resource as well as the automatic system recovery from failures. ZooKeeper is based on an in-memory data management. Thus, it ensures distributed coordination at a high speed. Zookeeper is increasingly used within Hadoop to provide high availability for the ResourceManager. It is used also by HBase to ensure servers management, bootstrapping, and coordination (George, 2011).

Unlike other components, Apache ZooKeeper (Junqueira and Reed, 2013) can be used outside Hadoop platform. ZooKeeper is used by Twitter, Yahoo and other companies within their distributed systems for configuration management, sharding, locking and other purposes. It is used also by In IBM's Big Insights and Apache Flume.

Apache Avro is a framework for modeling, serializing and making Remote Procedure Calls (RPC) (Shapira et al., 2015). Avro defines a compact and fast binary data format to support data-intensive applications, and provides support for this format in a variety of programming languages such us Java, Scala, C, C++ and Python (Maeda, 2012). Avro ensures efficient data compression and storages at various nodes of Apache Hadoop.

Within Hadoop, Avro passes data from one program or language to another (e.g., from C to Pig). Since data is stored with its schema (self-describing), Avro is compatible with scripting languages. There is a data serialization system at the core of Avro. Avro schemas can contain both simple and complex types. Avro uses JSON as an explicit schema or dynamically generates schemas of the existing Java objects.

Avro offers similar functionality of systems such as Thrift, Protocol Buffers, etc. However, Avro differs from those systems by ensuring: (i) Dynamic typing (data processing without code generation), (ii) untagged data, and (iii) No manually-assigned field IDs when a schema changes (Lublinsky et al., 2013).

Apache Oozie (Islam and Srinivasan, 2015) is a workflow scheduler system designed to run and manage jobs in Hadoop clusters. It is a reliable, extensible and scalable management system that can handle efficient execution of large volume of workflows. The workflow jobs take the form of a Directed Acyclical Graphs (DAGs). Oozie can support various types of Hadoop jobs including MapReduce, Pig, Hive, Sqoop and Distcp jobs (Kamrul Islam and Srinivasan, 2014). One of the main components of Oozie is the Oozie server. This server is based on two main components: a Workflow Engine that stores and runs different types of workflow jobs, and a Coordinator Engine that runs recurrent workflow jobs triggered by a predefined schedule (White, 2012). Oozie enables to track the execution of the workflows. In fact, users can customize Oozie in order to notify the client about the workflow and execution status via Http callbacks (e.g., workflow is complete, workflow enters or exits an action node). Currently, Oozie supports Derby by default in addition to other databases such us HSQL, MySQL, Oracle and PostgreSQL. Oozie provides a collection of APIs library and a command-line interface (CLI) that is based on a client component (Lublinsky et al., 2013).

4.7.2. System Deployment: Ambari, Whirr, BigTop and Hue

Apache Ambari (Wadkar and Siddalingaiah, 2014a) is designed to simplify Hadoop management thanks to an intuitive interface. It supports provisioning, managing, and monitoring Apache Hadoop clusters through an easy-to-use management Web User Interface. The interface is based on RESTful APIs. Ambari supports many Hadoop components such us: HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Moreover, Ambari ensures security over Hadoop clusters using Kerberos authentication protocol. It provides also role-based user authentication, authorization, and auditing functions to manage integrated LDAP and Active Directory.

Apache Whirr (Sammer, 2012) simplify the creation and deployment of clusters in cloud environments such as Amazon's AWS. It provides a collection of libraries for running cloud services. The operator can run Whirr as a command-line tool either locally or within the cloud. Whirr is used to spin up instances and to deploy and configure Hadoop. In addition, Apache Whirr supports provisioning of Hadoop as well as Cassandra, ZooKeeper, HBase, Valdemort (key-value storage), and Hama clusters on the cloud environments.

BigTop (Lovalekar, 2014) supports Hadoop ecosystem. It aims to develop packaging and verify Hadoop-related projects such as

those developed by the Apache community. The goal is to evaluate and to ensure the integrity and the reliability of the system as a whole rather than to evaluate each sub-module individually.

Hue (Chullipparambil, 2016) is a web application for interacting with Hadoop and its ecosystem. Hue aggregates the most common Hadoop components into a single interface. Its main goal is to enable programmers to use Hadoop without worrying about the underlying complexity or using a command line. HUE helps to browse the system, create and manage user accounts, monitor cluster health, create MapReduce jobs, and provides a front end for Hive called Beeswax. Beeswax has Wizards to help create Hive tables, load data, run and manage Hive queries, and download results in Excel format. Hue (Landset et al., 2015) is compatible with any version of Hadoop and is available in all of the major Hadoop distributions.

4.8. Summary

To sum up, this section discussed the Hadoop ecosystem including Hadoop Distributed File System (HDFS), HBaseNoSQL database, Hive data warehouse solution, and the Pig query language for ad hoc analytical requirements. The section provides an overview of other components of Hadoop environment including Hcatalog, Zookeeper, Avro, Oozie, Flume, Sqoop.

One of the major drawbacks to adopting MapReduce paradigm is that the development cycle of MapReduce jobs can take long time and may be complex for some use cases. Thus, advanced scripting languages (like Pig, JAQL) and query languages have been developed to simplify exploring and processing massive datasets. Such improvements reduce the number of code lines and enhance programmer productivity. Furthermore, they offer an intuitive and easy-to use interface to handle access management to different sources of Big Data. In fact, they can handle the storage and processing of various data sets that can be local, distributed over large networks or on the cloud. Another important feature is that they can channel data between data sources to sinks like Hbase and HDFS.

Various methods were also developed to simplify Big Data analysis. For instance R is an easy to use solution that can perform advanced analysis on large data sets via Hadoop. It provides a more complete set of classification models (regarding the types and depth of algorithms) in comparison to Mahout.

However, R clearly has some limitations. In fact, R is not a rapid solution when compared to other environment. This is due to its object-oriented programming that cause some memory management problems. For that reason, it is recommended to use Mahout, Spark, SAS or other rapid frameworks when the performance of extensive computations is a priority.

The current study enable us to recommend to organizations and users to carefully choose among the various available Big Data analytical tools (either open source or proprietary ones). The selection should be based on the importance of each factor: nature of data sets (e.i, volumes, streams, distribution), complexity of analytical problems, algorithms and analytical solutions used, systems capabilities, security and privacy issues, the required performance and scalability in addition to the available budget.

We believe also that it is difficult to afford some commercial tools at a personal level because of the prices and licensing issues.

As a another recommendation,users must be aware that relying on open source solutions may lead to outdating and modifications problem. However, open source systems supports the development and the innovation at a large scale. Organizations have to pay attention when choosing very recent technologies still in production. They have a limited maturity and may lack the support of developer communities or academic researchers.

5. Hadoop distributions

Various IT vendors and communities work to improve and enrich Hadoop infrastructure, tools and services. Sharing Big Data innovations through open source modules is helpful and promotes Big Data technologies. However, the downside is that users may end up with an Hadoop platform composed of various versions of modules from different sources. Because each Hadoop module has its own curve of maturity, there is a risk of versions incompatibility inside Hadoop platform. In addition, the integration of various technologies on the same platform increases security risks. Usually each module is tested. However, most of the time the combination of technologies from different sources may bring hidden risks that are not fully investigated nor tested.

To face those issues, many IT Vendors such us IBM, Cloudera, MapR and Hortonworks have developed their own modules and packaged them into distributions. One of the goals is to ensure compatibility, security and performance of all combined modules. Most of the available Hadoop distributions have been enriched gradually. They include various services such us: distributed storage systems, resource management, coordination services, interactive searching tools, advanced intelligence analysis tools, etc. Furthermore, Hadoop distribution providers offer their own commercial support.

5.1. Cloudera

Cloudera (Azarmi, 2016b) is one of the most used Hadoop distributions. It enables deploying and managing an Enterprise Data Hub powered by Hadoop. It provides many benefits such as a centralized administration tool, a unified batch processing, an interactive SQL, as well as a role-based access control.

In addition to that, Cloudera solutions can be integrated to a wide range of existing infrastructure and can handle disparate workloads and data formats in a single system. Cloudera proposes an easy way for browsing and querying data in Hadoop. In fact, it is possible to realize a real-time interactive querying and visualize the result in a convenient way. In addition, several tools are available to support security and data management.

One of the principle Cloudera modules is Impala (Sakr, 2016c). It constitutes an interesting query language module that is compatible with Hadoop. Impala structures data at rest on a columnar data format. It allows to handle interactive and real-time analysis on Big Data. On the contrary to Hive, Impala do not use MapReduce framework. Instead, it uses its own in-memory processing engine to ensure fast queries over large data sets. Thus, Impala is faster than Hive when returning querying results. Indeed, Impala like AMPLab Shark project (Manoochehri, 2013) can directly use data from existing HDFS and HBase sources. Thus, it minimizes data movement and hence reduces the execution time of "Big queries". Impala duplicates storage for fault-tolerance and enables the integration with major Business Intelligence tools. It includes also the native Hadoop security that is based on Kerberos for authentication and Apache Sentry for role-based authorization (Menon, 2014).

Cloudera (Prasad and Agarwal, 2016) provides also a flexible model that supports structured as well as unstructured data. Cloudera is faster than Hive. For instance, it executes queries at least 10 times faster than Hive/MapReduce. It has been confirmed that in comparison to HiveQL (Hive Query Language), Cloudera ensures 7 to 45 times performance gain for queries with at least one join. Even the aggregation queries have been speed-up by approximately 20–90 times. Cloudera also outperforms HiveQL or MapReduce in terms of real-time responsiveness: in fact, Cloudera Enterprise version reduces response time of queries to seconds instead of minutes using HiveQL or MapReduce.

In spite of all those cited advantages, Cloudera has some disadvantages (Prasad and Agarwal, 2016). For instance, it is not suitable for querying streaming data such as streaming video or continuous sensor data. In addition to that, all joins operations are performed in memory that is limited by the smallest memory node present in the cluster. Furthermore, Cloudera robustness can be affected by the single point failure during query execution. Indeed, it quits the entire query if any host that is executing the query fails. Cloudera Enterprise RTQ does not support internal indexing for files and does not allow to delete individual rows.

5.2. Hortonworks Data Platform

The Hortonworks Data Platform (HDP) (Azarmi, 2016a) is built on Apache Hadoop to handle Big Data storage, querying and processing. It has the advantage of being a rapid, cost-effective and scalable solution. It provides several services for management, monitoring and data integration.in addition to that, HDP has been positioned as a key integration platform since it provides open source management tools and supports connections with some BI platforms.

HDP ensures a distributed storage through DHFS and the non-relational database Hbase. It allows a distributed data processing based on the MapReduce, quering data through Hue and running scripts using Pig. HDP includes Oozie to manage and schedule workflows, as well as Hcatalog to handle Metadata services. Many tools are also available in HDP, including webHDFS, Sqoop, Talend Open Source, Ambari and Zookeeper.

5.3. Amazon Elastic MapReduce (EMR)

Amazon Elastic MapReduce (Amazon EMR) (Sakr, 2016a) is a web-based service built on Hadoop framework. It has the benefit of providing an easy, rapid and effective processing of huge data sets. Furthermore, it simplifies running Hadoop and related Big Data applications on AWS. It removes the cost and complexity of managing Hadoop installation. In addition, it allows resizing on demand the Amazon clusters by extending or shrinking resources. Thus, it is possible to easily extract valuable insight from big data sources without caring about the Hadoop complexity (Aher and Kulkarni, 2015).

This solution is popular in many industries and supports different goals such as log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics. It can handle many data source and types, including clickstream logs, scientific data, etc. Another advantage is that users can connect EMR to several tools like S3 for HDFS, backup recovery for HBase, Dynamo support for Hive. It includes many interesting free components such us Pig and Zookeeper.

5.4. MapR

MapR (Kobielus, 2012) is a commercial distribution for Hadoop designed for enterprises. It has been enhanced to provide a better reliability, performance and ease of use of Big Data storage, processing and especially analysis with machine learning algorithms. It provides a set of components and projects that can be integrated to a wide range of Hadoop ecosystem. MapR does not use HDFS. Indeed, MapR has developed its owns MapR File Systems (MapR-FS) in order to increase performance and enable easy backups. The MapR-FS has the advantage of being compatible with NFS. Thus, data can be easily transferred between them. MapR is based on the standard Hadoop programming model.

5.5. IBM InfoSphere BigInsights

IBM InfoSphere BigInsights is designed to simplify the use of Hadoop in the enterprise environment. It has the required potential to fulfill enterprise needs in terms of Big Data storage, processing, advanced analysis and visualization (Zikopoulos et al., 2011). The Basic Edition of IBM InfoSphere BigInsights includes HDFS, Hbase, MapReduce, Hive, Mahout, Oozie, Pig, ZooKeeper, Hue, and several other open source tools.

IBM InfoSphere BigInsights Enterprise Edition (Zikopoulos et al., 2012) provides additional important services: performance capabilities, reliability feature, built-in resiliency, security management and optimized fault-tolerance. It supports advanced Big Data analysis through adaptive algorithms (e.g., for text processing). In addition, IBM provides a data access layer that can be connected to different data sources (like DB2, Streams, dataStage, JDBC, etc.). It also leverages IBM Infosphere Streams (2013), another tool belonging to the Infosphere set. This IBM distribution has other advantages: first, the possibility to directly store data streams into BigInsights clusters. Second, it supports real-time analytics on data streams. This is achieved through a sink adapter and a source adapter to read data from clusters (Di Martino et al., 2014). IBM facilitates also visualization through Dashboards and Big Sheets (a spreadsheet-like interface for manipulating data in clusters).

5.6. GreenPlum's Pivotal HD

Pivotal HD (Hurwitz et al., 2013) provides advanced database services (HAWQ) with several components, including its own parallel relational database. The platform combines an SQL query engine that provides Massively Parallel Processing (MPP) (Gollapudi, 2013), as well as the power of the Hadoop parallel processing framework. Thus, the Pivotal HD solution can process and analyze disparate large sources with different data formats. The platform is designed to optimize native querying and to ensure dynamic pipelining.

In addition, Hadoop Virtualization Extensions (HVE) tool supports the distribution of the computational work across many virtual servers. Free features are also available for resource and workflow management through Yarn and Zookeeper. To support an easy management and administration, the platform provides a command center to configure, deploy, monitor and manage Big Data applications. For easier data integration, Pivotal HD proposes its own DataLoader besides the open source components Sqoop and Flume.

5.7. Oracle Big Data appliance

Oracle Big Data Appliance (Dijcks, 2012) combines, in one system, the power of optimized industry-standards hardware, Oracle software experience as well as the advantages of Apache Hadoop's open source components. Thus, this solution includes the open source distribution of Cloudera CDH and Cloudera Manager (Segleau et al., 2013).

Oracle Big Data Appliance is presented as a complete solution that provides many advantages: scalable storage, distributed computing, convenient user interface, end-to-end administration, easy-to-deploy system and other features. It supports also the management of intensive Big Data projects.

The Oracle appliance (Murthy et al., 2011) lies on the power of the Oracle Exadata Database Machine as well as the Oracle Exalytics Business Intelligence Machine. The data is loaded into the Oracle NoSQL database. It provides Big Data connectors for high-performance and efficient connectivity. It includes also an open source oracle distribution of R to support advanced analysis.

Table 6
Cloudera, Hortonworks and MapR features.

	Cloudera	Hortonworks	MAPR
Founded Year	Mars 2009	June 2011	2009
License	Multiple versions: Open source and Licensed	Open source	Licensed
GUI	Yes	Yes	Yes
Execution environement	Local or Cloud	Local or Cloud	Local or Cloud (Amazon)
Metadata architecture	Centralized	Centralized	Distributed
Replication	Data	Data	Data + metadata
Management tools	Cloudera Manager	Ambari	MapR Control System
File System Access	HDFS, read-only NFS	HDFS, read-only NFS	HDFS, read/write NFS (POSIX)
SQL Support	Impala	Stinger	Drill
Security	Supports default Kerberos based authentication for Hadoop services.	Supports default Kerberos based authentication for Hadoop services	Supports default Kerberos based authentication for Hadoop services.
Deployement	Deployement with Whirr toolkit. Complex deployment compared to AWS Hadoop or MapR Hadoop.	Deployement with Ambari. Simple Deployment.	Through AWS Management Console.
Maintenance	The maintenance and upgrade requires efforts. Job schulding is done through Oozie.	A set of operational capabilities that provide visibility of the health of the clusters.	Easy to maintain as cluster is managed through AWS Management Console and AWS toolkit.
Cost	Cloudera Standard is free. Cloudera entreprise version is proprietary, needs to be purchased separately. Costs are applicable based on components and tools adopted	HDP is the only completely open Hadoop data platform available. All solutions in HDP are developed as projects through the Apache Software Foundation. There are no proprietary extension	Billing is done through AWS on hourly basis.

The Oracle Big Data Enterprise can be deployed using Oracle Linux and Oracle Java Hotspot virtual machine Hotspot.

5.8. Windows Azure HDInsight

Windows Azure HDInsight (Sarkar, 2014; Nadipalli, 2015) is a cloud platform developed by Microsoft and powered by Apache Hadoop framework. It is designed for Big Data management on the cloud to store, process and analysis any type of large data sources. It provides simplicity, convenient management tools, and open source services for Cloud Big Data projects.Furthermore, it simplifies the processing and intensive analysis of large data sets in a convenient way. It integrates several Microsoft tools such as PowerPivot, PowerView and BI features.

5.9. Choosing a Hadoop distribution

As presented above, several distributions are available to support the use of Apach Hadoop. They are built on Hadoop platform with open source or proprietary components. In our view, the choice of a particular distribution depends on many factors, including use cases, business needs, Big Data analysis goals, problems to solve, as well as the existing infrastructure. Lublinsky et al. (2013) recommends to consider the following parameters when choosing among distributions:

- The technical characteristics: Hadoop version, available components, proposed functionalities and features (scalability, data availability, parallel processing, performance, connectivity with the existing applications and so on).
- The convenience level: easy tools for installation and configuration, user-friendly management interface, possibility to upgrade versions and integrate patches.
- The maintenance needs: clusters' management, the multi- centers support, the disaster-recovery support, and so on.
- The available budget as well as the cost of the selected solution. The cost includes the investments related to the deployment, maintenance and future upgrades and licenses.
- The support proposed to simplify the solution's integration with the existing infrastructure.

Table 6 shows the main characteristics of the major Hadoop distributions: Cloudera, Hortonworks and MAPR.

5.10. Summary

In sum, this section outlines some Big Data distributions. It is noticed that considerable progress has been made to improve Hadoop distributions. Thus, various tools and modules are available to handle Big Data projects in terms of:

- Data integration: distributions provides powerful open source tool to support data uploading and integration into the system. Some of them are based on their own DataLoader.
- Distributed storage: distributions are based either on the standard HDFS as a distributed le system or on their own file system. Distributions that incorporate their own file systems provides more convenience and easier back-ups. The file system is paired most of the time to a non-relational database and or rarely to a parallel relational database.
- Centralized management: we notice that most of the available distributions offer tools for workflow and resource management in parallel to services for coordination and system control. In addition, some of them provide advanced tools to manage the enterprise data hub. Considerable work has been achieved to offer reliable monitoring and diagnosis service of the entire platform through Heatmap, alerts, logs, audits, reports and more.
- Rapid and interactive analysis: some distributions offers machine learning tools, support complex algorithm for scalable advanced analytics.The usefulness of such distributions lies in offering most of the time their own version for searching and querying data in real-time (e.g., using API query, or specific query language). Some of them support even SQL interface to facilitate querying data. This is a strong point that helps SQL familiar users to take advantage from new advances in Big Data. As performance is one of the key issues in Big Data environment, distributions rely either on MapReduce frameworks or a specific in-memory processing engine to achieve a better performance and guarantee rapid queries.
- Security: most Hadoop distributions support native Hadoop security based on Kerberos for authentification and Apache

Sentry coupled to LDAP for fine grained role based access control. This constitute one of their strength as security is a major issue when dealing with distributed Big Data belonging to various proprietaries. Some distributions offer in parallel additional auditing tools and special features to ensure security on the Cloud on the bases of many options such us: VPN IPsec for encrypted connections, virtual private cloud instances, hardware isolation, list of network access control and security groups. In spite of those security solutions, users must be aware that dealing with security and privacy issues at different levels (i.e., data level, storage level and analysis level) is still complicated and rises many challenges (i.e., how to find a balance between security and privacy rules while extracting value and knowledge from continuous streams).

- Visualization: hopefully, current distributions offer various dynamic visualisation tools (e.g., dashboards, Big Sheets,reports, graphs). Thus, they facilitate expressive and intuitive visualization of data, query results, and they support easy Results monitoring. Some Big Data applications go a step forward by formulating recommendations to support decision making in various fields (heath, security, transport and so on).
- Cloud computing services: we notice that other researchers and professionals worked to offer distributions that can be seamlessly integrated to cloud platforms. So they offer many cloud services. This is important for web applications that need important computing resources. It is also useful to enable companies to externalize applications' maintenance and other management overhead to Cloud providers at an affordable cost.

As a summary, the studied distributions and tools enable to extract valuable insight and knowledge from Big Data in its different forms. However, such distributions have clearly some limitations and may differ in their offerings and capacities. We notice that many researchers are still going forward to optimize all the faces of the used technologies and approaches in order to face the increasing multi-streams and Big Data challenges. As a summary, the studied distributions and tools enable to extract valuable insight and knowledge from Big Data in its different forms. However, such distributions have clearly some limitations and may differ in their offerings and capacities. We notice that many researchers are still going forward to optimize all the faces of the used technologies and approaches in order to face the increasing multi-streams and Big Data challenges.

Most of Big Data surveys give an overview on Big Data applications, opportunities and challenges. Others discuss also techniques and methodologies used in Big data environment and how they can help to improve performance, scalability and results accuracy.

Wang et al. (2016) introduces an overview on Big Data including four issues, namely: (i) concepts, characteristics and processing paradigms of Big Data; (ii) the state-of-the-art techniques for decision making in Big Data; (iii) decision making applications of Big Data in social science; and (iv) the current challenges of Big Data as well as possible future directions. Benjelloun et al. (2015) presents several Big Data projects, opportunities, examples and models in many sectors such as healthcare, commerce, tourism and politics. It gives also examples of technologies and solutions developed to face Big Data challenge. Chen and Zhang (2014) presents also a survey about Big Data applications, opportunities, challenges and used techniques. In addition, it discuss methodologies to handle Big Data challenges (e.g., granular computing, cloud computing, bio-inspired computing, and quantum computing). Chen et al. (2014a) introduces the general background of big data and real-world applications. Unlike the previous papers, it discuss Big Data technologies by focusing on the four phases of the value chain of big data, i.e., data generation, data acquisition, data storage, and data analysis. For each phase, they present the general background, discuss the technical challenges, and review the latest advances. But it still does not give reader a sufficient comparison and understanding of the current Big Data technologies. Salleh and Janczewski (2016) provides a literature review on security and privacy issues of big data, while (Benjelloun and Ait Lahcen, 2015) presents Big Data security challenges and a state of the art in methods, mechanisms and solutions used to protect data-intensive information systems.

Sangeetha and Prakash (2017) reviews in more detail big data mining algorithms, data slicing techniques and clustering techniques. It also discusses their advantages and drawbacks as well as their performance and quality measurement. Wu et al. (2014) summarizes the key challenges for Big Data mining. It discuses some key research initiatives and the authors' national research projects in this field. Qiu et al. (2016) gives a brief review of conventional and advanced machine learning algorithms. It presents also the challenges of learning with big data and the corresponding recent solutions. Wang (2016) introduces methods in machine learning, main and new technologies in Big Data, and some applications of machine learning in Big Data. Challenges of machine learning applications in Big Data are also discussed. Skourletopoulos et al. (2017) presents a review of the current big data research. Thus, it explores applications, opportunities and challenges, as well as the state-of-the-art techniques and underlying models that exploit cloud computing technologies, such as the Big Data-as-a-Service (BDaaS) or Analytics-as-a-Service (AaaS).

It is worth mentioning that most of Big Data surveys do not focus on technologies and present algorithms and approaches used to process Big Data. For example, Radha and Rao (2016) presents the characteristics of Big Data applications and state of-the-art tools and techniques to handle data-intensive applications. Khan et al. (2014) Surveys and classifies the various attributes of Big Data, including its nature, definitions, rapid growth rate, volume, management, analysis, and security. This study also proposes a data life cycle that uses the technologies and terminologies of Big Data.

We notice that other articles focus just on one layer of the Big Data system (i.e., storage techniques, analytics tools or visualisation tools), in fact (Acharjya and Ahmed, 2016b) presents the tools used to analyze Big Data. Tsai (2016) presents and discusses big data analytics. Some important open issues and further research directions related to Big Data analytics are also presented.

The survey of Raghav et al. (2016) explains the major prerequisites and challenges that should be addressed by the recent exploration and visualization systems. It also describes about the different techniques and tools currently used for the visualization of large sets of data and their capabilities to support massive volume of data from variety of data sources.

Siddiqa et al. (2016) gives a comprehensive investigation of state-of-the-art storage technologies available for big data. Another paper (Oussous et al., in press) establish a precise picture about NoSQL Databases for Big Data as well as the advantages and disadvantages of the main NoSQL data models and frameworks.

Lee (2017) illustrates the application of data analytics using merchant review data. The impacts of big data on key business performances are then evaluated. Six technical and managerial challenges are discussed.

Rajaraman (2016) explains what is big data, how it is analysed, and give some case studies illustrating the potentials and pitfalls of big data analytics. Fang et al. (2015) presents an overview of big data initiatives, technologies and research in industries and academia, and discusses challenges and potential solutions.

Ali et al. (2016) Highlights the potential and applications of Big Data technologies for the development of many fields. It provides a background on Big Data techniques. It presents also a broad view of approaches used for human development such as Big Data

analytics. It discusses some technical challenges involved in implementing big data.

Luo et al. (2016b) reviews and discuss big data application in four major biomedical subdisciplines: (1) bioinformatics, (2) clinical informatics, (3) imaging informatics, and (4) public health informatics.

Thus, most Big Data surveys have only focused on discussing Big Data opportunities, applications, challenges and issues. Others preferred to survey and study the algorithms and techniques used in such context (i.e., data mining and machine learning). Only few surveys treat Big Data technologies regarding the aspects and layers that constitute a real-world Big Data system. In fact, most of the time, such surveys focus and discusses Big Data technologies from one angle (i.e., Big Data analytics, Big data mining, Big Data storage, Big Data processing or Big data visualisation). On the contrary, in our paper, we tried to treat the subject from the different angles. For that, we introduce fundamentals about Big Data, opportunities and applications. Then we present the main challenges encountered when facing the complexity of imbalanced data sets and difficulties arisen by the 5Vs of Big Data. After that, we present the main solutions. We especially compare the different solutions and technologies dedicated to Big Data. This is done for each layer of a general Big Data application (i.e., storage layer, access, querying, analysis, management layer). We paid attention to determine their advantages, features and facilities as well as their limits. The comparison and tables use many metrics such as scalability, results reliability (fault-tolerance), as well as the performance gain. Thus, the reader can have a comprehensive view about the components of any Big Data processing system and a deep understanding of the technological issues and advantages of the main solutions. Our aim is to help the reader to select the most suitable solution for each layer according to its case.

6. Conclusion

Current Big Data platforms are supported by various processing, analytical tools as well as dynamic visualization. Such platforms enable to extract knowledge and value from complex dynamic environment. They also support decision making through recommendations and automatic detection of anomalies, abnormal behavior or new trends.

In this paper, we have studied Big Data characteristics and deeply discussed the challenges raised by Big Data computing systems. In addition to that, we have explained the value of Big Data mining in several domains. Besides, we have focused on the components and technologies used in each layer of Big Data platforms. Different technologies and distributions have been also compared in terms of their capabilities, advantages and limits. We have also categorized Big Data systems based on their features and services provided to final users. Thus, this paper provides a detailed insight into the architecture, strategies and practices that are currently followed in Big Data computing. In spite of the important developments in Big Data field, we can notice through our comparison of various technologies, that many short comings exist. Most of the time, they are related to adopted architectures and techniques. Thus, further work needs to be carried out in several areas such as data organization, domain specific tools and platform tools in order to create next generation Big Data infrastructures. Hence, technological issues in many Big Data areas can be further studied and constitute an important research topic.

References

Acharjya, D., Ahmed, K.P., 2016a. A survey on big data analytics: challenges, open research issues and tools. Int. J. Adv. Comput. Sci. App. 7, 511–518.

Acharjya, D., Ahmed, K.P., 2016b. A survey on big data analytics: challenges, open research issues and tools. Int. J. Adv. Comput. Sci. App. 7, 511–518.

Aher, S.B., Kulkarni, A.R., 2015. Hadoop mapreduce: a programming model for large scale data processing. Am. J. Comput. Sci. Eng. Surv. (AJCSES) 3, 01–10.

Ali, A., Qadir, J., urRasool, R., urRasool, R., Sathiaseelan, A., Zwitter, A., Crowcroft, J., 2016. Big data for development: applications and techniques. Big Data Anal. 1, 2.

Ames, A., Abbey, R., Thompson, W., 2013. Big Data Analytics Benchmarking SAS, R, and Mahout. SAS Technical Paper.

Azarmi, B., 2016a. The big (data) problem. In: Scalable Big Data Architecture. Springer, pp. 1–16.

Azarmi, B., 2016b. Scalable Big Data Architecture. Springer.

Bansal, H., Mehrotra, S., Chauhan, S., 2016. Apache Hive Cookbook. Packt Publ..

Benjelloun, F.-Z., Ait Lahcen, A., 2015. Big data security: challenges, recommendations and solutions. In: Handbook of Research on Security Considerations in Cloud Computing. IGI Global, pp. 301–313.

Benjelloun, F.-Z., Ait Lahcen, A., Belfkih, S., 2015. An overview of big data opportunities, applications and tools. In: Intelligent Systems and Computer Vision (ISCV), 2015 (pp. 1–6). IEEE.

Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M., Kanne, C.-C., Ozcan, F., Shekita, E.J., 2011. Jaql: a scripting language for large scale semistructured data analysis. In: Proceedings of VLDB Conference.

Bishop, C.M., 2006. Pattern recognition. Mach. Learn. 128, 1–58.

Botta, A., de Donato, W., Persico, V., Pescapé, A., 2016. Integration of cloud computing and internet of things: a survey. Future Gener. Comput. Syst. 56, 684–700.

Brown, M.S., 2014. Data Discovery For Dummies, Teradata Special Edition. John Wiley & Sons Inc..

Chen, C.P., Zhang, C.-Y., 2014. Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. 275, 314–347.

Chen, M., Mao, S., Liu, Y., 2014a. Big data: a survey. Mobile Networks App. 19, 171–209.

Chen, M., Mao, S., Zhang, Y., Leung, V.C., 2014b. Big Data: Related Technologies, Challenges and Future Prospects. Springer.

Chullipparambil, C.P., 2016. Big Data Analytics Using Hadoop Tools (Ph.D. thesis). San Diego State University.

Coronel, C., Morris, S., 2016. Database Systems: Design, Implementation, & Management. Cengage Learning.

De carvalho, O.M., Roloff, E., Navaux, P.O., 2013. A Survey of the state-of-the-art in event processing. In: 11th Workshop on Parallel and Distributed Processing (WSPPD).

Di Martino, B., Aversa, R., Cretella, G., Esposito, A., Kołodziej, J., 2014. Big data (lost) in the cloud. Int. J. Big Data Intell. 1, 3–17.

Dijcks, J.P., 2012. Oracle: Big Data for the Enterprise. Oracle White Paper.

Dimiduk, N., Khurana, A., Ryan, M.H., Stack, M., 2013. HBase in Action. Manning Shelter Island.

Dinsmore, T.W., 2016. Streaming analytics. In: Disruptive Analytics. Springer, pp. 117–144.

Emani, C.K., Cullot, N., Nicolle, C., 2015. Understandable big data: a survey. Comput. Sci. Rev. 17, 70–81.

Fang, H., Zhang, Z., Wang, C.J., Daneshmand, M., Wang, C., Wang, H., 2015. A survey of big data research. IEEE Network 29, 6–9.

Furht, B., Villanustre, F., 2016. Introduction to big data. In: Big Data Technol. App.. Springer International Publishing, Cham, pp. 3–11.

Gandomi, A., Haider, M., 2015. Beyond the hype: big data concepts, methods, and analytics. Int. J. Inf. Manage. 35, 137–144.

George, L., 2011. HBase: The Definitive Guide. O'Reilly Media Inc..

Gollapudi, S., 2013. Getting Started with Greenplum for Big Data Analytics. Packt Publishing Ltd..

Hoffman, S., 2013. Apache Flume: Distributed Log Collection for Hadoop. Packt Publishing Ltd..

Hoffman, S., 2015. Apache Flume: Distributed Log Collection for Hadoop. Packt Publishing Ltd..

Hsieh, M.-Y., Li, G.-L., Liao, M.-H., Chou, W.-K., Li, K.-C., 2016. Accurate analysis of a movie recommendation service with linked data on hadoop and mahout. In: Frontier Computing. Springer, pp. 551–560.

Huang, Y., Li, T., Luo, C., Fujita, H., Horng, S.-J., 2017a. Dynamic variable precision rough set approach for probabilistic set-valued information systems. Knowledge-Based Syst. 122, 131–147.

Huang, Y., Li, T., Luo, C., Fujita, H., Horng, S.-J., 2017b. Matrix-based dynamic updating rough fuzzy approximations for data mining. Knowledge-Based Syst. 119, 273–283.

Hurwitz, J., Nugent, A., Halper, F., Kaufman, M., 2013. Big Data for Dummies. (1st ed.). For Dummies.

Islam, M.K., Srinivasan, A., 2015. Apache Oozie: The Workflow Scheduler for Hadoop. O'Reilly Media Inc..

Jadhav, A., Deshpande, L., 2016. A survey on approaches to efficient classification of data streams using concept drift. Int. J. 4.

Jain, A., 2013. Instant Apache Sqoop. Packt Publishing Ltd..

Junqueira, F., Reed, B., 2013. ZooKeeper: Distributed Process Coordination. O'Reilly Media Inc..

Kamrul Islam, M., Srinivasan, A., 2014. Apache Oozie The Workflow Scheduler for Hadoop. O'Reilly Media Inc..

Kankanhalli, A., Hahn, J., Tan, S., Gao, G., 2016. Big data and analytics in healthcare: introduction to the special section. Inf. Syst. Front. 18, 233–235.

Karau, H., 2013. Fast Data Processing with Spark. Packt Publishing Ltd..

Khan, N., Yaqoob, I., Hashem, I.A.T., Inayat, Z., Mahmoud Ali, W.K., Alam, M., Shiraz, M., Gani, A., 2014. Big data: survey, technologies, opportunities, and challenges. Sci. World J.

Kobielus, J.G., 2012. The forrester wave: Enterprise hadoop solutions, q1 2012. Forrester.
Krishnan, K., 2013. Data Warehousing in the Age of Big Data. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
Kune, R., Konugurthi, P.K., Agarwal, A., Chillarige, R.R., Buyya, R., 2016. The anatomy of big data computing. Software: Pract. Experience 46, 79–105.
Landset, S., Khoshgoftaar, T.M., Richter, A.N., Hasanin, T., 2015. A survey of open source tools for machine learning with big data in the hadoop ecosystem. J. Big Data 2, 1.
Lee, I., 2017. Big data: dimensions, evolution, impacts, and challenges. Bus. Horizons.
Letouzé, E., 2012. Big Data for Development: Challenges & Opportunities. UN Global Pulse.
Loganathan, A., Sinha, A., Muthuramakrishnan, V., Natarajan, S., 2014. A systematic approach to Big Data. Int. J. Comput. Sci. Inf. Technol. 4, 869–878.
Lovalekar, S., 2014. Big Data: an emerging trend in future. Int. J. Comput. Sci. Inf. Technol. 5.
Lublinsky, B., Smith, K.T., Yakubovich, A., 2013. Professional Hadoop Solutions. John Wiley & Sons.
Luo, C., Li, T., Chen, H., Fujita, H., Yi, Z., 2016a. Efficient updating of probabilistic approximations with incremental objects. Knowledge-Based Syst. 109, 71–83.
Luo, J., Wu, M., Gopukumar, D., Zhao, Y., 2016b. Big data application in biomedical research and health care: a literature review. Biomed. Inf. Insights 8, 1.
Lydia, E.L., Swarup, M.B., 2015. Big data analysis using hadoop components like flume, mapreduce, pig and hive. Int. J. Sci. Eng. Comput. Technol. 5, 390.
Lyko, K., Nitzschke, M., Ngomo, A.-C.N., 2016. Big data acquisition. In: New Horizons for a Data-Driven Economy. Springer, pp. 39–61.
Maeda, K., 2012. Comparative survey of object serialization techniques and the programming supports. J. Commun. Comput. 9, 920–928.
Maheswari, N., Sivagami, M., 2016. Large-scale data analytics tools: apache hive, pig, and hbase. In: Data Sci. Big Data Comput.. Springer, pp. 191–220.
Mall, N.N., Rana, S., et al., 2016. Overview of big data and hadoop. Imperial J. Interdiscip. Res. 2.
Manoochehri, M., 2013. Data Just Right: Introduction to Large-scale Data & Analytics. Addison-Wesley.
Mazumder, S., 2016. Big data tools and platforms. In: Big Data Concepts, Theories, and Applications. Springer, pp. 29–128.
McAfee, A., Brynjolfsson, E., et al., 2012. Big Data: the management revolution. Harvard Bus. Rev. 90, 60–68.
Menon, R., 2014. Cloudera Administration Handbook. Packt Publishing Ltd..
Murthy, B., Goel, M., Lee, A., Granholm, D., Cheung, S., 2011. Oracle Exalytics in-Memory Machine: A brief Introduction.
Nadipalli, R., 2015. HDInsight Essentials. Packt Publishing Ltd..
Nahar, J., Imam, T., Tickle, K.S., Chen, Y.-P.P., 2013. Computational intelligence for heart disease diagnosis: a medical knowledge driven approach. Expert Syst. App. 40, 96–104.
Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015a. Deep learning applications and challenges in big data analytics. J. Big Data 2, 1.
Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015b. Deep learning applications and challenges in big data analytics. J. Big Data 2, 1.
Nambiar, R., Bhardwaj, R., Sethi, A., Vargheese, R., 2013. A look at challenges and opportunities of Big Data analytics in healthcare. In: In: 2013 IEEE International Conference on Big Data. IEEE, pp. 17–22.
Nathan, P., 2013. Enterprise Data Workflows with Cascading. O'Reilly Media Inc..
Oussous, A., Benjelloun, F.-Z., AitLahcen, A., Belfkih, S., Comparison and classification of nosql databases for big data. Int. J. Big Data Intell. (In press).
Park, B.-J., Oh, S.-K., Pedrycz, W., 2013. The design of polynomial function-based neural network predictors for detection of software defects. Inf. Sci. 229, 40–57.
Prasad, B.R., Agarwal, S., 2016. Comparative study of big data computing and storage tools: a review. Int. J. Database Theory App. 9, 45–66.
Purcell, B.M., 2013. Big Data using cloud computing. Holy Family Univ. J. Technol. Res.
Qiu, J., Wu, Q., Ding, G., Xu, Y., Feng, S., 2016. A survey of machine learning for big data processing. EURASIP J. Adv. Signal Process. 2016, 1–16.
Rabkin, A., Katz, R.H., 2010. Chukwa: a system for reliable large-scale log collection. In: LISA. vol. 10. pp. 1–15.
Radha, K., Rao, B.T., 2016. A study on big data techniques and applications. Int. J. Adv. Appl. Sci. 5, 101–108.
Raghav, R., Pothula, S., Vengattaraman, T., Ponnurangam, D., Raghav, R., Pothula, S., Vengattaraman, T., Ponnurangam, D., 2016. A survey of data visualization tools for analyzing large volume of data in big data platform. In: International Conference on Communication and Electronics Systems (ICCES). IEEE, pp. 1–6.
Raim, A.M., 2013. Introduction to Distributed Computing with pbdR at the UMBC High Performance Computing Facility. Technical Report HPCF-2013-2, UMBC High Performance Computing Facility, University of Maryland, Baltimore County.
Rajaraman, V., 2016. Big data analytics. Resonance 21, 695–716.
Razzak, M.I., Naz, S., Zaib, A., 2017. Deep learning for medical image processing: Overview, challenges and future. arXiv preprint arXiv:1704.06825.
del Río, S., López, V., Benítez, J.M., Herrera, F., 2014. On the use of mapreduce for imbalanced big data using random forest. Inf. Sci. 285, 112–137.
Sakr, S., 2016. Big data 2.0 processing systems: a survey. Springer Briefs in Computer Science.
Sakr, S., 2016b. General-purpose big data processing systems. In: Big Data 2.0 Processing Systems. Springer, pp. 15–39.
Sakr, S., 2016c. Introduction. In: Big Data 2.0 Processing Systems: A Survey. Springer International Publishing, Cham, pp. 1–13.
Sakr, S., 2016d. Large-scale stream processing systems. In: Big Data 2.0 Processing Systems. Springer, pp. 75–89.
Salleh, K.A., Janczewski, L., 2016. Technological, organizational and environmental security and privacy issues of big data: A literature review. Procedia Comput. Sci. 100, 19–28.
Sammer, E., 2012. Hadoop Operations. O'Reilly Media Inc..
Sangeetha, J., Prakash, V.S.J., 2017. A survey on big data mining techniques. Int. J. Comput. Sci. Inf. Secur. 15, 482.
Sarkar, D., 2014. Introducing HDInsight. In: Pro Microsoft HDInsight. Springer, pp. 1–12.
Schmarzo, B., 2013. Big Data: Understanding How Data Powers Big Business. John Wiley & Sons.
Segleau, D., Laker, K., Stackowiak, R., Mishra, G., Harding, D., Mohiuddin, K., Sun, H., Hornick, M., Nelson, B., Macdonald, B., Plunkett, T., 2013. Oracle Big Data Handbook. McGraw-Hill, Oracle Press.
Shapira, G., Seidman, J., Malaska, T., Grover, M., 2015. Hadoop Application Architectures. O'Reilly Media Inc..
Shaw, S., Vermeulen, A.F., Gupta, A., Kjerrumgaard, D., 2016. Hive architecture. In: Practical Hive. Springer, pp. 37–48.
Shireesha, R., Bhutada, S., 2016. A study of tools, techniques, and trends for big data analytics. IJACTA 4, 152–158.
Siddiqa, A., Karim, A., Gani, A., 2016. Big data storage technologies: a survey. Frontiers 1.
Skourletopoulos, G., Mavromoustakis, C.X., Mastorakis, G., Batalla, J.M., Dobre, C., Panagiotakis, S., Pallis, E., 2017. Big data and cloud computing: a survey of the state-of-the-art and research challenges. In: Advances in Mobile Cloud Computing and Big Data in the 5G Era. Springer, pp. 23–41.
Skowron, A., Jankowski, A., Dutta, S., 2016. Interactive granular computing. Granular. Computing 1, 95–113.
Stimmel, C.L., 2014. Big Data Analytics Strategies for the Smart Grid. CRC Press.
Stoianov, N., Urueña, M., Niemiec, M., Machnik, P., Maestro, G., 2013. Integrated security infrastructures for law enforcement agencies. Multimedia Tools App., 1–16
Sun, J., Fujita, H., Chen, P., Li, H., 2017. Dynamic financial distress prediction with concept drift based on time weighting combined with adaboost support vector machine ensemble. Knowledge-Based Syst. 120, 4–14.
Team, R.C., 2000. R Language Definition. R foundation for statistical computing, Austria.
Tsai, C.-W., Lai, C.-F., Chao, H.-C., Vasilakos, A.V., 2016. Big data analytics. In: Big Data Technologies and Applications. Springer, pp. 13–52.
Usha, D., Aps, A.J., 2014. A survey of Big Data processing in perspective of hadoop and mapreduce. International Journal of Current Engineering and Technology.
Vohra, D., 2016. Using apache sqoop. In: Pro Docker. Springer, pp. 151–183.
Wadkar, S., Siddalingaiah, M., 2014a. Apache Ambari. In: Pro Apache Hadoop. Springer, pp. 399–401.
Wadkar, S., Siddalingaiah, M., 2014b. Hcatalog and hadoop in the enterprise. In: Pro Apache Hadoop. Springer, pp. 271–282.
Wang, H., Xu, Z., Fujita, H., Liu, S., 2016. Towards felicitous decision making: an overview on challenges and trends of big data. Inf. Sci. 367, 747–765.
Wang, H., Xu, Z., Pedrycz, W., 2017. An overview on the roles of fuzzy set techniques in big data processing: trends, challenges and opportunities. Knowledge-Based Syst. 118, 15–30.
Wang, L., 2016. Machine learning in big data. Int. J. Adv. Appl. Sci. 4, 117–123.
Wang, S., Yao, X., 2012. Multiclass imbalance problems: analysis and potential solutions. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 42, 1119–1130.
Weiss, R., Zgorski, L., 2012. Obama Administration Unveils Big Data Initiative: Announces 200 Million in New R&D Investments. Office of Science and Technology Policy, Washington, DC.
White, T., 2012. Hadoop: The Definitive Guide. O'Reilly Media Inc..
Wu, X., Zhu, X., Wu, G.-Q., Ding, W., 2014. Data mining with big data. IEEE Trans. Knowl. Data Eng. 26, 97–107.
Yu, H., Ni, J., Zhao, J., 2013. Acosampling: An ant colony optimization-based undersampling method for classifying imbalanced dna microarray data. Neurocomputing 101, 309–318.
Zang, W., Zhang, P., Zhou, C., Guo, L., 2014. Comparative study between incremental and ensemble learning on data streams: case study. J. Big Data 1, 1–16.
Zhang, J., Li, T., Ruan, D., Liu, D., 2012. Rough sets based matrix approaches with dynamic attribute variation in set-valued information systems. Int. J. Approximate Reasoning 53, 620–635.
Zhou, L., 2013. Performance of corporate bankruptcy prediction models on imbalanced dataset: the effect of sampling methods. Knowledge-Based Syst. 41, 16–25.
Zhou, L., Fujita, H., 2017. Posterior probability based ensemble strategy using optimizing decision directed acyclic graph for multi-class classification. Inf. Sci. 400, 142–156.
Zhou, L., Wang, Q., Fujita, H., 2017. One versus one multi-class classification fusion using optimizing decision directed acyclic graph for predicting listing status of companies. Inf. Fusion 36, 80–89.
Zikopoulos, P., Eaton, C., et al., 2011. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media.
Zikopoulos, P., Parasuraman, K., Deutsch, T., Giles, J., Corrigan, D., et al., 2012. Harness the Power of Big Data The IBM Big Data Platform. McGraw Hill Professional.