BIG DATA ANALYSIS IN HEALTHCARE: APACHE HADOOP, APACHE SPARK AND APACHE FLINK

Elham Nazari[1], Mohammad Hasan Shahriari[2], Hamed Tabesh[1*]

[1]Department of Medical Informatics, Faculty of Medicine, Mashhad University of Medical Sciences, Mashhad, Iran.
[2]Department of Electrical Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

Article Info

Article type:
Review

Article History:
Received: 2019-03-03
Revised: 2019-05-02
Accepted: 2019-06-02

**** Corresponding author:***
Hamed Tabesh
Department of Medical Informatics, Faculty of Medicine, Mashhad University of Medical Sciences, Mashhad, Iran.
Email: tabeshh@mums.ac.ir

A B S T R A C T

Introduction:

Health care data is increasing. The correct analysis of such data will improve the quality of care and reduce costs. This kind of data has certain features such as high volume, variety, high-speed production, etc. It makes it impossible to analyze with ordinary hardware and software platforms. Choosing the right platform for managing this kind of data is very important.

The purpose of this study is to introduce and compare the most popular and most widely used platform for processing Big Data, Apache Hadoop MapReduce, and the two Apache Spark and Apache Flink platforms, which have recently been featured with great prominence.

Material and Methods:

This study is a survey whose content is based on the subject matter search of the Proquest, PubMed, Google Scholar, Science Direct, Scopus, IranMedex, Irandoc, Magiran, ParsMedline and Scientific Information Database (SID) databases, as well as Web reviews, specialized books with related keywords and standard. Finally, 80 articles related to the subject of the study were reviewed.

Results:

The findings showed that each of the studied platforms has features, such as data processing, support for different languages, processing speed, computational model, memory management, optimization, delay, error tolerance, scalability, performance, compatibility, Security and so on. Overall, the findings showed that the Apache Hadoop environment has simplicity, error detection, and scalability management based on clusters, but because its processing is based on batch processing, it works for slow complex analyzes and does not support stream processing, Apache Spark is also distributed as a computational platform that can process a Big Data set in memory with a very fast response time, the Apache Flink allows users to store data in memory and load them multiple times and provide a complex Fault Tolerance mechanism Continuously retrieves data stream status.

Conclusion:

The application of Big Data analysis and processing platforms varies according to the needs. In other words, it can be said that each technology is complementary, each of which is applicable in a particular field and cannot be separated from one another and depending on the purpose and the expected expectation, and the platform must be selected for analysis or whether custom tools are designed on these platforms.

Keywords:

Big Data Analysis, Apache Hadoop, Apache Spark, Apache Flink, Healthcare.

How to cite this paper

Nazari E, Shahriari MH, Tabesh H. Big Data Analysis in Healthcare: Apache Hadoop, Apache spark and Apache Flink. Front Health Inform. 2019; 8(1): e14. DOI: 10.30699/fhi.v8i1.180

INTRODUCTION

With the development of new technologies, health care data is increasing. Estimated in 2012, the data is about 200 petabyte (PB), estimated to reach 250000 PB by 2020 [1]. Analysis of these data is very important for acquiring knowledge, for extracting useful information and for discovering hidden data patterns. And in the area of health care will improve the quality of services, reduce costs and reduce errors [2, 3]. This kind of data has many features: including high volume, variety, scalability, complexity, high speed production and uncertainty, which makes it possible to use common data mining techniques and typical software and hardware to analyze this type of data [3-7].

A Big Data analysis is a process that organizes the data collected from various sources and then analyzes the data sets in order to discover the facts and meaningful patterns [8].

Large-scale data analyzes have many uses in the field of health: for example, early diagnosis of diseases such as breast cancer, in the processing of medical images and medical signals for providing high-quality diagnosis, monitoring patient symptoms, tracking chronic diseases such as diabetes, preventing incidence of contagious diseases, education through social networks, genetic data analysis and personalized (precision) medicine. Examples of this type of data are omics data, including genomics, transcriptomics, proteomics and pharmacogenomics, biomedical data, web data and data in various electronic health records (EHRs) and hospital information systems (HISs). Data contained in the EHR and HIS contain rich data including demographic characteristics, test results, diagnosis and information of each individual [9-17]. Therefore, analysis of health data has been considered with regard to its importance, so that it has led scholars and scientists to create structures, methodologies, approaches and new approaches for managing, controlling and processing Big Data [18].

In recent years, many tools have been introduced for Big Data analysis. We intend to introduce the tools provided by the Apache Software Foundation and then compare them with each other after defining the Big Data and its features. Some of the tools available for Big Data analysis are Apache Hadoop [19], Spark [20], and Flink [21], the focus of these tools is on batch processing or stream processing. Mostly batch processing tools are based on the Apache Hadoop Infrastructure such as Apache Mahout. Data stream analysis programs are often used for real-time analysis. Spark and Flink are examples of data flow analysis platforms. The interactive analysis process allows users to interact directly in real time to conduct their analysis. For example, "Hive and Drill" are cluster platforms that support interactive analysis. These tools help researchers develop Big Data project [22].

MATERIALS AND METHODS

The Big Data is a term used for data with a volume greater than 1018 or Exabyte, and storage, management, sharing, analysis, and visualization of this type of data is difficult due to its characteristics [23, 24]. The analysis of this type of data includes these steps:

- Acquisition
- Information extraction and cleaning
- Data integration
- Modeling and analysis
- Interpretation and deployment [25].

The Big Data is defined by the attributes. These features are, in fact, the challenges that the Big Data analysis has to address and need to be managed. At the beginning of the emergence of this type of data, three features were raised; in studies of 8 characteristics for the big data, and in 2017, 42 characteristics were proposed, and it is expected to reach 120 characteristics by the year 2021 [26]. Further, these descriptions are some important features:

- Volume: refers to the production of high-volume data that requires a lot of storage space.
- Velocity: Data rates are unpredictable.
- Variety: Variety of data and its various formats. Data may fall into three categories: structured, semi-structured, and unstructured. They can also have different types of images, videos and audio.
- Veracity: refers to bias, noise and abnormality in large data. Extreme noise, incomplete, inaccurate, inaccurate or extra data, or, in other words, data quality imply this characteristic.
- Vagueness: refers to the vagueness of the information.
- Versatility: Different for different contexts [4-7].

Apache Haddop is a suite of open source software that facilitates solving issues with Big Data through the use of a large number of computers. Many people regard the two Hadoop and MapReduce as similar, while this is not true [27]. In fact, Hadoop uses the MapReduce software model to provide a framework for storing and processing Big Data. While Hadoop was originally designed to use

computing on weak and medium systems, it was gradually being used in high-end hardware [28].

In recent years, many projects have been developed to complete or modify the Hadoop, for this purpose the term "Hadoop Ecosystem" is used to refer to projects and related products. In Fig 1 the Hadoop Ecosystem is shown [27].

To fully understand the Hadoop, you need to look at both yourself and the ecosystem around it. The Hadoop project consists of four main components [27]:

1) Hadoop Distributed File System (HDFS): A file system for storing huge volumes of data across a cluster of systems. HDFS has master-slave architecture. It provides high-throughput and error tolerance systems, which holds more than three copies of each data block.

2) MapReduce data processing engine: The distributed programming and processing model is distributed.

3) Resource Management Layer, YARN (also known as MapReduce Version 2): The new model is a distributed job and places jobs among the cluster. This model provides a distinction between infrastructure and programming model.

4) Commons libraries used in different parts of the Hadoop that are also used elsewhere. Some of these libraries have been implemented in java, including compression codecs, I/O utilities, and error detection [19-25].

The Hadoop Ecosystem consists of several projects around the main components mentioned above. These projects are designed to help researchers and experts in all stages of the analysis and machine learning workflow. The general structure of the ecosystem consists of three layers: the storage layer, the processing layer and the management layer [26-40].

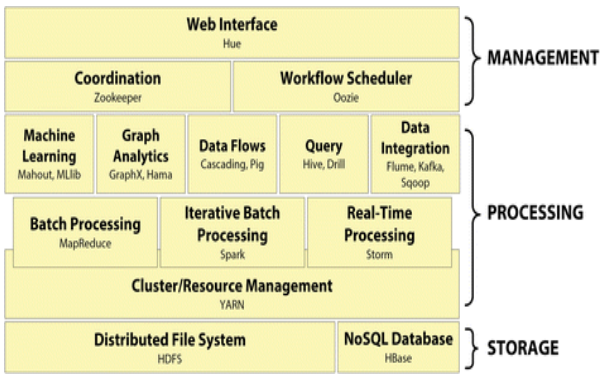

Fig 1: Hadoop Ecosystem [27]

MapReduce is a Google-generated programming model for Big Data processing based on the "divide and conquer" method [41, 42]. The divide and conquer method is implemented in two steps: Map and Reduction. The steps in performing the MapReduce model are shown in Fig 2 [43].

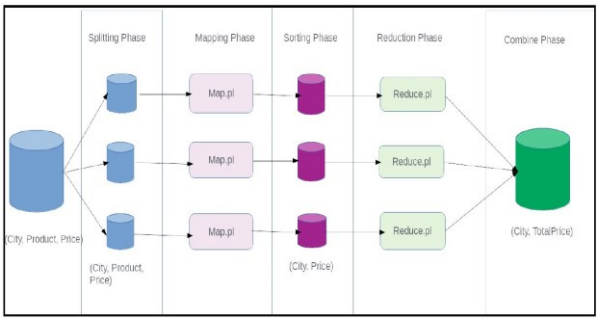

Fig 2: Steps to Perform the MapReduce Model [43]

MapReduce programming enables a large amount of data to be processed in parallel. Based on this model, each software is a sequence of MapReduce operations consisting of a Map stage and a Reduce step that is used to process a large number of independent data. These two main actions are applied to a key, value [44]:

Mapping step: The main node takes the input, dividing it into smaller issues. Then they distribute them between nodes that are tasked with doing things. This node may also repeat the same thing, in which case we have a multi-level structure. Ultimately, these sub-issues are processed and sent to the original node.

Step Reduce: Now, the main node that receives the responses and the results combines them to provide output. Meanwhile, actions may be performed on results, such as filtering, summarizing or converting.

These two main actions are applied to a key, value. The Map function takes a tidy pair of data and converts it into a list of ordered pairs:

Map (k1, v1) -> list (k2, v2)

Then, the MapReduce framework collects all pairs with the same key from all the lists and groups them together. Then, for each key generated, a group is created. Now the Reduce function applies to each group:

Reduce (k2, list (v2)) -> list (v3)

The MapReduce framework now converts a list of (key, values) into a list of values. The device should be able to process a list of (key, values) in the main memory.

One of the important features of MapReduce is that the failed nodes are categorized automatically and the complexity of the error tolerance is hidden from the viewpoint of the programmers [44].

RESULTS

Advantages of MapReduce are parallel processing, high scalability, low hardware cost, large file processing, maximum operational capability, and data locality [42, 44, 45].

Disadvantages of MapReduce are:

- One challenge and two-step dataflow challenge. It does not directly support tasks with different data flows.
- The user must repeatedly copy the join and filtering and aggregation codes manually, which will waste time, create program errors, reduce readability and impede optimization.
- Even for typical operations such as Filtering and Projection, custom codes are written that cause problems with reuse and maintenance.
- The vague nature of the Map and Reduce functions hinders the system's ability to be optimized.
- No support for streaming data [42, 44-51]

Apache Spark is a text-based framework that was presented at the University of Berkeley in 2009. The existence of multiple advantages has made the processing engine a powerful and useful process for macro processing, and distinguishes it from other tools, such as Hadoop and Storm [18, 20, 52-55].

All features provided by Apache Spark are built on top of the core [56]. For Spark, there are three Java APIs, Python and Scala. The Spark core is an API location that defines the resilient distributed RDD dataset, which is the concept of Spark's original programming [57]. Its key features are [56]:

1. Responsible for the essential I/O capabilities.

2. Its important role in planning and observing Spark cluster.

3. Recovering errors by using computations in memory solves the complexity of MapReduce.

Advantages of Apache Spark are:

- Easily installed.
- Open source professionals like Intel, IBM, Databricks, Cloudera and MapR have officially announced that they support and support Apache Spark's standards as the standard engine for large data analysis.
- Eliminates the needs of large-format processing with various data (text data, data charts, etc.) and also manages data sources properly.
- Surely, 10 to 100 times faster than the Hadoop because of processing in memory, it also works better with MapReduce in executing the program on the disk.
- Supports various programming languages from Python to Scala and Java. The system has a set of over 80 high-level operators and can be interactively used for querying data within the shell.
- For duplicate processing, interactive processing and event processing.
- Can be integrated with Hadoop Distributed File System (HDFS), Hbase, and Casendra and other storage systems [18, 46, 50, 52, 57-59].

The main issue is whether with the arrival of Spark, we have to leave the Hadoop, and do they differ from each other? In the answer, Spark and Hadoop cannot be considered completely separate from each other and with the advent of new tools from the past, we must say that Spark has integrated with the Hadoop and has overcome its problems [50, 51]. Spark does not use MapReduce as its executable engine, but is well integrated with the Hadoop. Because it can run in yarn and work with the Hadoop and HDFS data format. There is no way to create security in Spark, and it needs to be connected to the security mechanisms in YARN, such as Kerberos. As a result, Spark can be more powerful in combination with Hadoop [20, 54, 60].

Another high-level Apache project is the Flink project, which has exactly the same mission as Spark in the Hadoop Ecosystem, and has been introduced as an alternative to the MapReduce Hadoop model. Apache Flink is an open source stream processor. Flink provides speed, efficiency, and precision for mass processing, and can handle batch processing or even direct and stream processing.

Many of the concepts of the two tools are similar, but we will see Flink as a more diverse and lighter option. For example, both data streams in Spark and Flink guarantee that each record is processed exactly once. As a result, any duplicates that may be available will be deleted. Compared to other processing systems, such as Storm, they have a very high operational capability, but both have a low error overhead.

The same flake with Spark can do structured query language (SQL), graph, machine learning and stream processing. It also works with NoSQL, relational database management system (RDBMS), like the SQL server and MongoDB. Flink is a combination of MapReduce based on memory-based disk and spark. One of Fink's advantages over Spark is the following:

Since Flink contains a memory management system,

memory processing will be faster than Spark [18, 28].

It has better performance for repetitive processes. Duplicate processing runs on a node independently of the cluster, which will increase the speed.

Can run classic MapReduce processing and also integrate with Apache TEZ [61].

Spark thanks to the micro-classification architecture provides near-real-time flow, while Apache Flink provides real-time stream for real-time. Due to pure stream architecture based on the Kappa architecture [48, 51, 59].

Flink has a pipeline nature in processing data and chooses the best method for doing it. In Fig 3, the architecture and components of the Flink are shown [62].

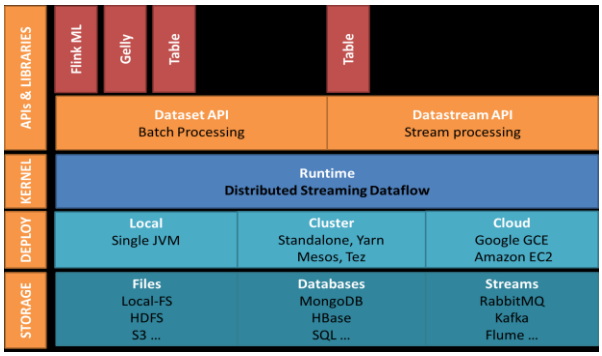

Fig 3: Architecture and components of Apache Flink [42]

Table 1 shows the differences between the Hadoop, Spark, and Flink and they are examined more precisely:

Table 1: The Comparison of the features of Hadoop, Spark and Flink [2, 18, 20, 42, 43, 45-48, 51, 52, 57, 59, 62-66]

Attributes	Apache Flink	Apache Spark	Apache Hadoop
Data processing	Provides stream and batch processing	Batch processing, also supports stream processing	Batch processing
Language support	Java, Scala, Python and R	Java, Scala, Python and R	First of all Java, but other languages like Groovy, Ruby, C ++, C, Python, Perl are also supported
Extended language	Java, Scala	Scala	Java
Processing speed	It processes faster than spark because of its underlying engine	Processes 100 times faster than MapReduce because processing is done in memory	Data processing is much slower than Spark and Flink
Computational model	Flink has adopted a continuous stream based on an operator-driven stream model. A continuous flow operator processes data quickly, without delay, in the collection. Flink can order that only some of the information that has actually been changed be processed. Therefore, job performance is significantly increased	Spark provides a near-real-time stream due to micro-batching architecture. Repeats your data batchwise. Each iteration should be planned and implemented separately	MapReduce selects the batch model. Batch essentially processes data in rest mode, captures a large amount of data, then processes and then writes in the output and does not support iterative processing.
Memory Management	Provides automatic storage management	Provides customizable memory management. In recent versions of Spark, automatic memory management is also possible	Provides customizable memory management. In other words, you can permanently or dynamically manage memory
Optimization	Flink comes with an optimizer that is independent of the programming interface. Flink Optimizer acts as a relational databases optimizer.	Job should be optimized manually. There is an optimizer named Catalyst, which is made in Scala language.	Job should be optimized manually
Delay	With the Apache Flink configuration, data execution time is achieved with low latency and high speed. Flink can process data (at very high speeds and large volumes) in milliseconds	The Hadoop is relatively faster because it stores a lot of input data in memory by the RDD and keeps the average data in memory, and eventually writes the data after it is completed or, if necessary, writes to the disk	The delay in the Hadoop is greater than Spark and Flink. The reason for this delay is support for various formats and structures and a huge amount of data.
Fault tolerance	The fault tolerance mechanism in Apache Flink is based on Chandy-Lamport distributed snapshots. This mechanism is lightweight, which keeps the interest rate high and at the same time ensures strong adaptability.	Uses the Flexible Distributed Dataset (RDDs). So that it will retrieve the program after no error, without the need for code or additional settings	It is very robust against the error and there is no need to reboot the program in case of any errors.
Scalability	Scalability is very high.	Scalability is very high.	MapReduce has potentially scalability that can be used in products with several thousand nodes

Attributes	Apache Flink	Apache Spark	Apache Hadoop
Performance	Flint's performance is superior to any other information processing system. Flink used repetitive closed loop operators to accelerate machine learning and graph processing.	Stream processing is not as efficient as it uses micro-batch processing.	The Hadoop's performance of Spark and Flink is slower.
Delete Repeat	Processes each record exactly once and then duplicates it.	Processes each record exactly once and then duplicates it	Not available
Compatibility	It is fully compatible with the Hadoop, it can process data stored in the Hadoop and support all file formats / input formats.	Hadoop and Spark are compatible with each other, and Spark will share all mapping-down compatibility for data sources, file formats, and business intelligence tools through Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC).	MapReduce and Spark are compatible with each other
Security	Through the Hadoop / Kerberos infrastructure there is support for user authentication. If Flink is running on YARN, Flink will confirm with its Kerberos Tokens that the user has YARN, HBase, and HDFS.	Authentication is only through authentication password. If SPARC runs on the Hadoop, it uses HDFS ACLS and file-level permissions. In addition, Spark can run on YARN and allow Kerberos to authenticate.	It supports Kerberos authentication, which is manageable somewhat difficult.
Cost	To run in memory, it requires high Random Access Memory (RAM), so it costs a lot.	It needs high RAM, because it needs a higher cost	It runs on cheap hardware.
Abstraction	Dataset abstraction for batch processing and DataStreams abstraction for stream processing	Spark RDD abstraction for batch processing and DStream abstraction for stream processing	There is no abstraction.
Visualization	Provides a web interface for sending and executing jobs. An implementation plan can be displayed on this interface.	Provides a web interface for sending and executing jobs in which the implementation plan can be visualized. Flink and Spark are both connected to Apache zeppelin, which provides data analysis as well as discovery, visualization and collaboration.	The Zoom Data visualization tool can be connected directly to HDFS, as well as to SQL-on-Hadoop, Impala, Hive, Spark SQL, Presto, and more.
Easy to use	It is easy because it has high level operators.	It is easy because it has high level operators.	MapReduce developers need to code each operation, which makes it very difficult, but add-ons like Pig and Hive make it a bit easier to work with.
Real-time analysis	Basically used for real-time processing. However, it also provides quick batch processing.	Supports real-time processing.	MapReduce fails in the face of real-time data processing.
SQL Support	Table Application Programming Interface (API) is a descriptive language similar to SQL that supports a data format such as DSL.	Using Spark-SQL executes SQL commands.	Ability to execute SQL commands with Apache Hive is possible.

DISCUSSION

Considering the necessity of analyzing health data that is increasing day by day [25] and the importance of considering the appropriate software platform, this study examined and compared the three platforms Apache Hadoop, Apache Spark and Apache Flink. The results showed that, depending on the needs, the efficiency of the data analysis and processing platforms varies, in other words, it can be said that each technology is complementary and each one is applicable in a particular field and cannot be separated from one another. For example, when the volume discussion was raised, Hadoop first implemented MapReduce, which has a high processing speed parallel to the volume Then, with the advancement of technology, a variety of discussions came about, and other tools came into being and were based on different aspects. If there is a need for real-time data processing and stream in a single event, Spark will be the appropriate option it will not be a must and should be used with a Flink, so the use of each technology depends on the need and that we can combine these tools together to achieve the desired purpose. The results of this study can help researchers and those who are seeking Big Data analytics in the field of health and medical care in choosing the appropriate platform.

Big Data analysis improves health care services and reduces costs. The results of well-conducted studies and projects in the field of health care in the context

of the Big Data analysis illustrate this fact. According to a report, these analyzes will cost $340 to $450 billion in various prevention, diagnosis and treatment departments [67, 68].

One of the most famous recently implemented projects is IBM Watson. In this study, Watson's physician will help in identifying symptoms and factors associated with patient diagnosis and treatment and making better decisions.

In the field of health care, 80% of the complex data (MRI images, medical notes, etc.) are used to perform these analyzes, based on the need of different platforms. Hadoop helps researchers and doctors get a glimpse of data that has never been possible before.

It finds correlations in the data with many variables, which is a very difficult task for humans and can be effective in the discovery and prevention of diseases and the treatment of chronic diseases. One MapReduce demo, this demo helps write a program that can remove duplicate CT scan images from a 100 million photos database. Anticipated wearable technologies of 26.54% in care for the elderly and in intensive care during the period 2020-2016 will create a change in the field of health care. Collected data can be stored in Hadoop and analyzed using MapReduce and Spark and will save costs [69].

Hadoop is well placed to upgrade hospital services, especially when hospitals sit at the bedside with sensors for checking the status of blood pressure, cholesterol, and so on, while using the RDBMS, it's no longer possible to get this information in the long run. Production is saved. More than 40 percent of people have admitted that high insurance costs are due to the large number of fraudulent complaints that cost more than one billion. Insurance companies use a hypothetical environment to reduce these scams. They use historical and real-time data on medical complaints, wages, and so on.

At Texas Hospital, Hadoop was used in electronic medical records (EMRs) and found that patients in their 30-day treatment period needed additional care and treatment, and with the help of the Hadoop platform, they could reduce the readmission rate from 26 to 21; this means that using Hadoop in EMR, they could reduce the readmission rate by 5%.

96% of US hospitals have EHR, while in Asia and India, this is very low. The EHR is a rich source of data analysis and well-managed patient planning changes and changes. In India, a hospital called AIIMS uses the Big Data analysis to improve the quality of services [70].

In addition to using the platform's capabilities, it is possible to add and use tools for the needs of these platforms to carry out the analysis of the database. In a study, the Medoop tool, a medical platform based on Hadoop-based, was proposed. This system uses the features of scalability, high reliability and high throughput in the Hadoop [71]. In the study, a Hadoop image processing interface (HIFI) tool was developed for MapReduce Image-based activities [72]. In a study, the SparkSeq tool was also proposed, which was also added to Spark, with the goal of analyzing the genomic data and the expression of the cloud-based gene expression for the purpose of discovering translated strings for a type of cancer or another tool called Thunderbuilt for analysis of large-scale, neural data [73, 74]. A study from Apache Spark has been used to analyze functional MRI data and avoid frequent write-ups on disk [75]. Flink has also been used to monitor electrocardiogram (ECG), magnetic resonance imaging (MRI) reading, wearable sensor monitoring, and other cyber-physical systems, and is also useful in analyzing genomic data and has been reported to have a high fault tolerance [76, 77].

CONCLUSION

It is suggested that future studies introduce other platforms and make more comparisons with different platforms and their capabilities for managing Big Data in the field of health. It is also suggested that, according to the target, custom tools should be designed and incorporated into existing platforms to be used.

AUTHOR'S CONTRIBUTION

All the authors approved the final version of the manuscript.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest regarding the publication of this study.

FINANCIAL DISCLOSURE

No financial interests related to the material of this manuscript have been declared.

REFERENCES

1. Hermon R, Williams PA. Big data in healthcare: What is it used for? 3rd Australian eHealth Informatics and Security Conference; 2014.
2. Chen M, Mao S, Liu Y. Big data: A survey. Mobile Netw Appl. 2014; 19(2): 171-209.
3. Ristevski B, Chen M. Big data analytics in medicine and healthcare. J Integr Bioinform. 2018; 15(3): 1-5. PMID: 29746254 DOI: 10.1515/jib-2017-0030 [PubMed]
4. Mooney SJ, Pejaver V. Big data in public health: Terminology, machine learning, and privacy. Annu Rev Public Health. 2018; 39: 95-112. PMID:

29261408 DOI: 10.1146/annurev-publhealth-040617-014208 [PubMed]

5. Jin X, Wah BW, Cheng X, Wang Y. Significance and challenges of big data research. Big Data Research. 2015; 2(2): 59-64.
6. Bello-Orgaz G, Jung JJ, Camacho D. Social big data: Recent achievements and new challenges. Information Fusion. 2016; 28: 45-59.
7. Arockia Panimalar S, Varnekha Shree S, Veneshia Kathrine A. The 17 V's of big data. International Research Journal of Engineering and Technology. 2017; 4(9): 329-33.
8. Goga K, Xhafa F, Terzo O. VM deployment methods for DaaS model in clouds. In: Barolli L, Xhafa F, Javaid N, Spaho E, Kolici V. (eds) Advances in internet, data & web technologies. Lecture notes on data engineering and communications technologies, vol 17. Springer, Cham; 2018.
9. Khan AS, Fleischauer A, Casani J, Groseclose SL. The next public health revolution: Public health information fusion and social networks. Am J Public Health. 2010; 100(7): 1237-42. PMID: 20530760 DOI: 10.2105/AJPH.2009.180489 [PubMed]
10. Velikova M, Lucas PJF, Samulski M, Karssemeijer N. A probabilistic framework for image information fusion with an application to mammographic analysis. Medical Image Analysis. 2012; 16(4): 865-75.
11. Antink CH, Leonhardt S, Walter M. A synthesizer framework for multimodal cardiorespiratory signals. Biomedical Physics & Engineering Express. 2017; 3(3): 035028.
12. Sung W-T, Chang K-Y. Evidence-based multi-sensor information fusion for remote health care systems. Sensors and Actuators A: Physical. 2013; 204: 1-19.
13. Benke K, Benke G. Artificial intelligence and big data in public health. Int J Environ Res Public Health. 2018; 15(12): 2796-805. PMID: 30544648 DOI: 10.3390/ijerph15122796 [PubMed]
14. Kim W-J. Knowledge-based diagnosis and prediction using big data and deep learning in precision medicine. Investig Clin Urol. 2018; 59(2): 69–71. PMID: 29520381 DOI: 10.4111/icu.2018.59.2.69 [PubMed]
15. Lee CH, Yoon H-J. Medical big data: Promise and challenges. Kidney Res Clin Pract. 2017; 36(1): 3–11. PMID: 28392994 DOI: 10.23876/j.krcp.2017.36.1.3 [PubMed]
16. Archenaa J, Anita EM. A survey of big data analytics in healthcare and government. Procedia Computer Science. 2015; 50: 408-13.
17. Andreu-Perez J, Poon CCY, Merrifield RD, Wong STC, Yang G-Z. Big data for health. IEEE Journal of Biomedical and Health Informatics. 2015; 19(4): 1193-208.
18. Verma A, Mansuri AH, Jain N. Big data management processing with hadoop MapReduce and spark technology: A comparison. Symposium on Colossal Data Analysis and Networking. 2016; IEEE.
19. Taylor RC. An overview of the hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics. 2010; 11(12): S1.
20. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, et al. Apache spark: A unified engine for big data processing. Communications of the ACM. 2016; 59(11): 56-65.
21. Carbone P, Ewen S, Haridi S. Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. 2015.
22. García-Gil D, Ramírez-Gallego S, García S, Herrera F. A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. Big Data Analytics. 2017; 2(1): 1-11.
23. O'Driscoll A, Daugelaite J, Sleator RD. 'Big data', hadoop and cloud computing in genomics. J Biomed Inform. 2013; 46(5): 774-81. PMID: 23872175 DOI: 10.1016/j.jbi.2013.07.001 [PubMed]
24. Sagiroglu S, Sinanc D, editors. Big data: A review. International Conference on Collaboration Technologies and Systems (CTS). 2013: IEEE.
25. Jagadish H, Gehrke J, Labrinidis A, Papakonstantinou Y, Patel JM, Ramakrishnan R, et al. Big data and its technical challenges. Communications of the ACM. 2014; 57(7): 86-94.
26. Shafer T. The 42 V's of big data and data science [Internet]. 2017. [cited: 15 May 2019] Available from: https://www. kdnuggets.com/2017/04/42-vs-big-data-data-science.html.
27. Landset S, Khoshgoftaar TM, Richter AN, Hasanin T. A survey of open source tools for machine learning with big data in the hadoop ecosystem. Journal of Big Data. 2015; 2(1): 24-60.
28. Dunning T, Friedman E. Real world hadoop. O'Reilly Media; USA: 2015.
29. Hoffman S. Apache Flume: distributed log collection for hadoop. Packt Publishing Ltd; 2013.
30. Garg N. Apache kafka. Packt Publishing Ltd; 2013.
31. Ting K, Cecho JJ. Apache sqoop cookbook: Unlocking hadoop for your relational database. O'Reilly Media; USA: 2013.
32. White T. Hadoop: The definitive guide. O'Reilly Media; USA: 2012.
33. Hausenblas M, Nadeau J. Apache drill: Interactive ad-hoc analysis at scale. Big Data. 2013; 1(2): 100-4. PMID: 27442064 DOI: 10.1089/big.2013.0011 [PubMed]
34. Fernández A, del Río S, López V, Bawakid A, del Jesus MJ, Benítez JM, et al. Big data with cloud computing: An insight on the computing environment, MapReduce, and programming frameworks. Data Mining and Knowledge Discovery. 2014; 4(5): 380-409.
35. Wu D, Sakr S, Zhu L. Big data programming models.

In: Zomaya A, Sakr S. (eds) Handbook of Big Data Technologies. Springer; Cham: 2017.

36. Pol UR. Big data analysis: Comparison of hadoop mapreduce, pig and hive. International Journal of Innovative Research in Science, Engineering and Technology. 2016; 5(6): 9687-93.
37. Oozie. Apache oozie workflow scheduler for hadoop. [Internet] 2019. [cited: 1 Jul 2019]. Available from: https://oozie.apache.org/
38. Olasz A, Thai BN, Kristóf D. A new initiative for tiling, stitching and processing geospatial big data in distributed computing environments. ISPRS Ann Photogramm Remote Sens Spatial Inf Sci. 2016; 3(4): 111-8.
39. Masiane M, Warren L. CS5604 front-end user interface team. [Internet] 2016 [cited: 1 Jul 2019] Available from: https://vtechworks.lib.vt.edu/handle/10919/70935
40. Shrivastava A, Deshpande T. Hadoop blueprints. Packt Publishing; 2016.
41. Sinha S. What is a hadoop ecosystem? [Internet]. 2017 [cited: 1 Jul 2019]. Available from: https://www.quora.com/What-is-a-Hadoop-ecosystem.
42. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM. 2008; 51(1): 107-13.
43. Kumar VN, Shindgikar P. Modern big data processing with hadoop: Expert techniques for architecting end-to-end big data solutions to get valuable insights. Packt Publishing; 2018.
44. Thomas L, Syama R. Survey on MapReduce scheduling algorithms. International Journal of Computer Applications. 2014; 95(23): 9-13.
45. Team D. Hadoop vs spark vs flink: Big data frameworks comparison [Internet]. 2016 [cited: 1 Jul 2019]. Available from: https://www.data-flair.training/blogs/hadoop-vs-spark-vs-flink./
46. Carbone P, Katsifodimos A, Ewen S, Markl V, Haridi S, Tzoumas K. Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. 2015; 36(4): 28-38.
47. Chintapalli S, Dagit D, Evans B, Farivar R, Graves T, Holderbaugh M, et al. Benchmarking streaming computation engines: Storm, flink and spark streaming. International Parallel and Distributed Processing Symposium Workshops, IEEE; 2016.
48. Frampton, M., Mastering Apache Spark. Packt Publishing; 2015.
49. Monteith JY, McGregor JD, Ingram JE. Hadoop and its evolving ecosystem. 5th International Workshop on Software Ecosystems. Citeseer; 2013.
50. Parsian M. Data algorithms: Recipes for scaling up with hadoop and spark. O'Reilly Media, USA; 2015.
51. Singh D, Reddy CK. A survey on platforms for big data analytics. Journal of Big Data, 2015. 2(1): 8-28.
52. Estrada R, Ruiz I. Big data SMACK: A guide to apache spark, mesos, akka, cassandra, and kafka. Apress; 2016.
53. Meng X. Mllib: Scalable machine learning on spark. Spark Workshop; 2014.
54. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, et al. Fast and interactive analytics over hadoop data with Spark. Usenix Login. 2012; 37(4): 45-51.
55. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I, et al. Spark: Cluster computing with working sets. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing; 2010.
56. Team, D. Apache spark ecosystem: Complete spark components guide [Internet]. 2017 [cited: 1 Dec 2018]. Available from: https://data-flair.training/blogs/apache-spark-ecosystem-components
57. Safabakhsh M. Apache spark [Internet]. 2018 [cited: 1 Jul 2019]. Available from: http://myhadoop.ir/?page_id=131.
58. Penchikala S. Big data processing with apache spark–Part 1: Introduction [Internet]. 2015 [cited: 1 Jul 2019]. Available from: https://www.infoq.com/articles/apache-spark-introduction.
59. Shoro AG, Soomro TR. Big data analysis: Apache spark perspective. Global Journal of Computer Science and Technology. 2015; 15(1): 7-14.
60. Kupisz B, Unold O. Collaborative filtering recommendation algorithm based on hadoop and spark. International Conference on Industrial Technology. IEEE; 2015.
61. Saha B, Shah H, Seth S, Vijayaraghavan G, Murthy A, Curino C. Apache tez: A unifying framework for modeling and building data processing applications. International Conference on Management of Data. ACM; 2015.
62. Team D. Flink tutorial: A comprehensive guide for apache flink [Internet]. 2018 [cited: 1 Jan 2019]. Available from: https://data-flair.training/blogs/flink-tutorial./
63. Tsai C-W, Lai C-F, Chao H-C, Vasilakos AV. Big data analytics: A survey. Journal of Big Data. 2015; 2(1): 21-53.
64. Oussous A, Benjelloun F-Z, Lahcen AA, Belfkih S. Big data technologies: A survey. Journal of King Saud University-Computer and Information Sciences. 2018; 30(4): 431-48.
65. Ramírez-Gallego S, Fernández A, García S, Chen M, Herrera F. Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. Information Fusion. 2018; 42: 51-61.
66. Ferranti A, Marcelloni F, Segatori A, Antonelli M, Ducange P. A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. Information Sciences. 2017; 415: 319-40.
67. Nazari E, Pour R, Tabesh H. Comprehensive overview

of decision-fusion technique in healthcare: A scoping review protocol. Iran J Med Inform. 2018; 7(1): e7.

68. Kayyali B, Knott D, Van Kuiken S. The big-data revolution in US health care: Accelerating value and innovation. Mc Kinsey & Company. 2013; 2(8): 1-13.

69. Poojary P. Big data in healthcare: How hadoop is revolutionizing healthcare analytics [Internet]. 2019 [cited: 15 May 2019]. Available from: https://www.edureka.co/blog/hadoop-big-data-in-healthcare.

70. HDFS Tutorial Team. How big data is solving healthcare problems successfully? [Internet]. 2016 [cited: 15 May 2019]. Available from: https://www.hdfstutorial.com/blog/big-data-application-in-healthcare/

71. Lijun W, Yongfeng H, Ji C, Ke Z, Chunhua L. Medoop: A medical information platform based on hadoop. International Conference on e-Health Networking, Applications and Services. IEEE; 2013.

72. Sweeney C, Liu L, Arietta S, Lawrence J. HIPI: A hadoop image processing interface for image-based mapreduce tasks. International Journal of Recent Trends in Engineering & Research. 2011; 2(11): 557-62.

73. Wiewiórka MS, Messina A, Pacholewska A, Maffioletti S, Gawrysiak P, Okoniewski MJ. SparkSeq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision. Bioinformatics. 2014; 30(18): 2652-3. PMID: 24845651 DOI: 10.1093/bioinformatics/btu343 [PubMed]

74. Freeman J, Vladimirov N, Kawashima T, Mu Y, Sofroniew NJ, Bennett DV, et al. Mapping brain activity at scale with cluster computing. Nat Methods. 2014; 11(9): 941-50. PMID: 25068736 DOI: 10.1038/nmeth.3041 [PubMed]

75. Boubela RN, Kalcher K, Huf W, Našel C, Moser E. Big data approaches for the analysis of large-scale fMRI data using apache spark and GPU processing: a demonstration on resting-state fMRI data from the human connectome project. Front Neurosci. 2016; 9: 492. PMID: 26778951 DOI: 10.3389/fnins.2015.00492 [PubMed]

76. Versaci F, Pireddu L, Zanetti G. Scalable genomics: From raw data to aligned reads on Apache YARN. International Conference on Big Data. IEEE; 2016.

77. Harerimana G, Jang B, Kim JW, Park HK. Health big data analytics: A technology survey. IEEE Access. 2018; 6: 65661-78.