REVIEW ARTICLE

European Journal of Information Technologies and Computer Science
www.ej-compute.org

Real-Time Big Data Analytics for Data Stream Challenges: An Overview

Alaa Abdelraheem Hassan and Tarig Mohammed Hassan

ABSTRACT

The conventional approach of evaluating massive data is inappropriate for real-time analysis; therefore, analyzing big data in a data stream remains a critical issue for numerous applications. It is critical in real-time big data analytics to process data at the point where they are arriving at a quick reaction and good decision making, necessitating the development of a novel architecture that allows for real-time processing at high speed and low latency. Processing and analysing a data stream in real-time is critical for a variety of applications; however, handling a large amount of data from a variety of sources, such as sensor networks, web traffic, social media, video streams, and other sources, is a considerable difficulty. The main goal of this paper is to give an overview of the current architecture for real time big data analytics, real-time data stream processing methods available, including their system architectures Lambda, kappa, and delta large data stream processing.

Keywords: Apache spark, Apache storm, Delta, Hadoop, Kappa, Lambda.

Published Online: July 25, 2022

ISSN: 2736-5492

DOI :10.24018/ejcompute.2022.2.4.62

A. A. Hassan
Khartoum University, Khartoum, Sudan.
(e-mail: aloosh-131@hotmail.com)
T. M. Hassan
Khartoum University, Khartoum, Sudan.
(e-mail: tarig_harbi71@hotmail.com)

I. Introduction

Big Data refers to rapidly growing datasets with different sizes and complexities beyond the capability of traditional data base tools, properly manage and carefully analyse them. Huge amounts of data are created and captured in big quantities every day from many sorts of applications such as network sensors, public web, corporate applications, and other technologies. The flow of data is entering in such a rapid and complicated manner and must be managed, stored, and analyzed to offer knowledge necessary to improve those applications.

Some applications were rely on time in their logic, such as data stream or streaming data management, processing, and analysing. These applications and their analytics must deal with data in real time and decisions must be made within a certain time frame.

Businesses and enterprises can use Big Data analytics to learn more about their position and performance, and they can manipulate knowledge to improve decision-making process. The key difficulty of the big data stream is performance - much study in recent years has focused on the performance problem [1].

II. Background

A continuous transfer of information from one location to another over an extended period of time is common in real-time applications [2]. A stream is a series of connected data, such as video and audio data. The term "data stream processing" refers to the process of processing these types of events. Event stream processing is a simplified version of composite event processing CEP that refers to processing a single stream. Streaming data, also known as event stream processing, is a continuous flow of data created by numerous sources and is a core idea in real-time analytics systems. Data streams can be processed, stored, analyzed, and displayed in real-time utilizing stream processing Technology [3]. The term "streaming" refers to never-ending data streams with no beginning or end that provide a constant stream of data that may be processed.

Data streams are generated in a variety of forms and volumes by a variety of sources. They can all be collected to seamlessly acquire real-time information and analytics from a single source of data, from applications, networking devices, and server log files to website activity, banking, transactions, and internet of things (IOT) sensors.

The concept of an event is central to many modeling and implementation methodologies for growing real-time applications. An event is occurring in the system environment for which event-driven computing can provide a more appropriate reaction and higher throughput [3].

EP stands for event processing, which is computing that performs actions on events as they are reported in a system that monitors or listens to events in the environment. Reading, producing, altering, and processing events are all common information processing operations [4].

Many event processing applications are built to interact with events from the environment, and their design adheres to event-driven architecture principles.

For situations when action must be done as so as on feasible, event stream processing is critical. This is why event stream processing environments are often described as real-time processing.

A. Event Stream Processing and Data Stream Processing

Stream processing and Complex Event Processing are the two primary aspects of the EP (CEP). The first step is event processing, which may be used for filtering, enrichment, classification, and joining, among other continuous analytics operations. CEP detects and reports composite events by looking for patterns in sequences of basic events. For operations on events that are distinct from the application logic, CEP offers event processing logic. This can save money on development and maintenance. The logic of event processing is frequently expressed in area-specific languages referred to as event processing languages (EPLs). Infosphere Streams [5], Tibco Stream Base [6], and Oracle Stream Explorer [7] are examples of enterprise scale CEP systems built by big corporate vendors like IBM, Tibco, and Oracle.

The development of various data stream processing and analytics platforms is influenced by these systems.

An event processing system (EPS) uses event type and patterns to process data related to events entering from the environment. Event type identifies properties of events that occur in the environment. The time stamp when an event occurred and various data kinds connected with event load are characteristics of an event type, but event patterns indicate relationships between events in the actual world [8].

B. Real Time Big Data Systems

A real-time system is one in which IT systems must process events as they occur and within a set time interval. Depending on the system, this time period is in the mille, micro, or even nanosecond range. Real-time systems are frequently assumed to be those in which accuracy is dependent on timeliness [9]. The following are the most significant qualities of a real-time system:

The period between an event occurring in the system's environment and the start of its processing is referred to as latency.

Availability refers to a system's ability to perform its function when it is needed. High availability is required for real-time systems; otherwise, events are not handled immediately and are difficult to store or buffer for subsequent processing, particularly when dealing with high volume and high velocity data streams.

Also Horizontal Scalability, Mean the system's ability to add servers to an existing pool in order to increase capacity and performance. For real-time systems to ensure that data is processed in predefined time intervals, the ability to dynamically add new servers as data volume and workload demands is vital [9].

C. Batch Processing

When a collection of transactions is accumulated over time, it is a time-saving approach of processing massive volumes of data. Data is gathered, inputted, and processed before batch results are generated. The basic purpose of a batch processing system is to keep all of the tasks in a batch running at the same time.

D. Real Time Processing

Real-time processing systems are extremely rapid and responsive. These systems are employed in situations when a huge number of events need to be accepted and handled quickly. Real-time processing necessitates rapid transactions and is characterized by prompt responses.

Data must be downloaded in batches before it can be processed, stored, or analyzed in batch data processing methods, whereas streaming data comes in constantly and may be treated at the same time [13].

III. Real Time Big Data Analytic Platform

The lowest layer of the diagram displays the infrastructure platforms that are necessary to provide computing resources for the development of analytics systems. IBM, Oracle, and Tico's traditional enterprise distributed systems, as well as cloud PaaS [15], [16], are examples.

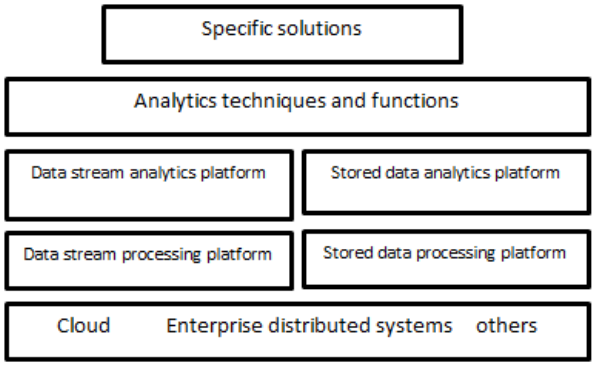

Fig. 1. Real time analytic platform [16].

The second tier illustrates big data platforms, which are software platforms that handle enormous amounts of data, either batch or stream data, leveraging the underlying resources provided by infrastructure Platforms. Stored data is handled by Hadoop and Spark, while stream data is analyzed using Kafka and Storm. The analytics approaches and tools are depicted in the following tier. This layer, which may be thought of as a library of techniques to generate analytics solutions for specific business issues, involves text processing and machine learning methods and models, either real-time or stored data layers. It's tough to fit all of today's technologies and solutions onto a single layer. Microsoft Azure and Amazon provide components for the majority of these tiers [16].

A. Big data Stream Processing Tools

Data stream processing techniques such as Apache Hadoop, Apache Spark, and Apache Storm demonstrate that earlier approaches, like Map Reduce, do not allow for real-time processing despite the ability to analyse a high volume of data and the speed with which the data arrives [14].

1) Hadoop

Hadoop is an open-source software platform for scalable, dependable, distributed computing and large-scale processing [17]. It was built as a batch processing system to handle processing massive data collections of structured, unstructured, and semi-structured data.

As a result, it does not meet the performance requirements for real-time data analytics. By combining Hadoop with the Apache Storm environment, Hadoop may be used for real-time processing in this case.

Hadoop is made up of several components [18]:

1) Map Reduce is a model environment for distributed data processing that works on huge clusters of commodity processors. In batch mode, Map Reduce was utilized to handle large amounts of data [18], [19].
2) HDFS (Hadoop Distributed File System) is a distributed file system for large clusters of computers. HDFS is a file system for storing massive amounts of data.
3) YARN (Yet another Resource Negotiator) is the Hadoop Yarn administrator who is in charge of cluster management in the system.

Hadoop has a master-slave design with two servers, which are the foundations of the Map Reduce framework, a single master node, and a number of worker nodes [20]. HDFS is a distributed, scalable, and robust storage system that can easily collaborate with Map Reduce [21]. Through the network, it provides a substantial amount of aggregate bandwidth. HDFS is made up of two parts: a master node called name node and data nodes.

Map Reduce, created by Google in 2004, is a fault-tolerant framework for processing massive volumes of data in parallel on big clusters of computers. [22], [23]. It is divided into two phases: the map and reduce phase [22], and the reduce phase [24].

Due to its elasticity and scalability, Map Reduce is a fault-tolerant framework for batch processing of massive volumes of data. But it is not a viable meth od for real-time processing.

Many key components of the Hadoop ecosystem, including as hive, zookeeper, Kafka, and Flume, remain to deal with various databases and data types.

The Apache Software Foundation created Kafka, an open-source message broker system built in Scala [25]. Kafka processes reads and writes from thousands of clients at a high rate of gigabytes per second. To provide high availability and horizontal scalability, data streams are segregated and scattered throughout a cluster of computers [25], [26]. Kafka relies on Zookeeper for political coordination of processing nodes in the sensitive Hadoop environment. For configuration management, naming, and distributed synchronization, Zookeeper obtains a centralized service.

Flume is a distributed, dependable, and available service for gathering, aggregating, and transporting huge volumes of log data in a professional manner. It features a straightforward and adaptable design based on streaming data flows [27]. It has configurable reliability and recovery methods and is resilient and fault resistant. It employs a simple data model that enables online analytic applications.

Flume and Kafka are the event mainstays for real-time processing, and they each offer their own set of features. Kafka is ideal for high-volume communications systems that need scalability and availability. Flume is more suitable for data input and basic event processing, however it is not suitable for CEP applications. Flume and Kafka are used in tandem by several real-time applications to govern their own characteristics [28], [29].

2) SPARK

Spark is a distributed computing framework that is open-source [30]. It's a collection of useful tools and software components organized into a logical framework. AMP Lab at the University of California, Berkeley, usually develops and designs it.

Spark conducts a cluster-level data read and all relevant analytical operations by publishing the findings to the same level.

Spark is developed in Scala, although it can also be used with Java and Python [31].

The fundamental distinction between Map Reduce and Spark in Hadoop is that Map Reduce works in stages, whereas Spark works on all data at the same time. Batch processing can be up to ten times quicker, while in-memory analysis can be up to a hundred times faster. All task analysis activities are normally performed in memory and in real time by Spark [32]. It only uses disks when its memory becomes insufficient. Hadoop, on the other hand, writes data to disk after each successful process.

Spark, on the other hand, lacks a file management mechanism. Hadoop Distributed File System is used.

3) STORM

Storm [34] is a distributed, fault-tolerant, and processing-guaranteed real-time processing system. Storm was founded at Back Type, a private company that was purchased by Twitter in 2011. It is an Eclipse Public License-licensed open-source project.

Storm completes what Hadoop achieved for batch processing in real-time by making cognitive processing of unlimited data flows clear and trustworthy. The storm is simple to use and may be implemented in any computer language [34], [35].

A storm cluster typically consists of three essential nodes: The Hadoop Job Tracker's equivalent is Nimbus.

Supervisor: Responsible for starting and finishing the creative process.

Zookeeper**:** A node that acts as the storm cluster's common coordinating service.

Storm uses a variety of topologies known as spouts and bolts, whereas Hadoop uses Map Reduce jobs. Storm is a real-time distributed computing system that can manage massive amounts of data.

In stream computing, Spout reads from a queuing broker such as Kafka, but it may also construct its own stream.

Bolt: any number of input streams can be handled, as well as any number of new output streams [37]-[38].

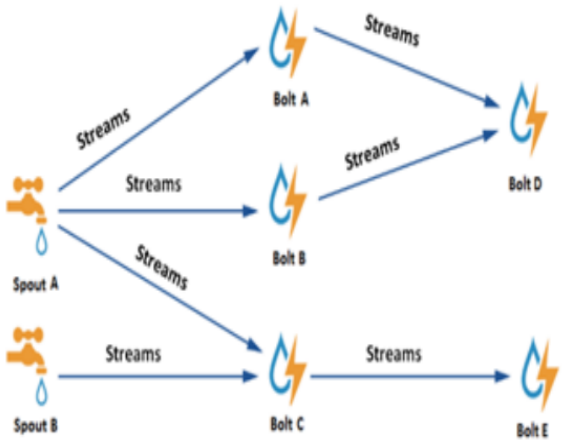

Fig. 2. Spout and bolt technology [36].

European Journal of Information Technologies and Computer Science
www.ej-compute.org

IV. Related Work

Millions of Likes are added to Facebook every second throughout the world. Millions of new websites are launched, and a large number of presidential tweets are sent out. All these multiple internet technology performers produce very large data point and information in the form of streamed data. While processing a vast quantity of data from many sources, including as sensor networks, online traffic, social media, video streams, and other sources, remains a substantial difficulty [39], processing a data stream in real time remains a vital issue for multiple applications.

Hadoop is a fault-tolerant framework with excellent scalability. It processes large amounts of data in batches and provides useful insight into more historical data. Map Reduce is unsuited for real-time stream processing, since processing data as soon as it arrives is critical [39].

Lambda, Kappa, and Delta are three current designs that are often based on real-time processing:

A. Lambda Architecture

An architecture that combines real-time and batch processing in a single framework gives minimal latency and excellent outcomes [40].Serving layer: Combines the batch and speed layer's findings [41].

Batch layer allows for complicated pre-computation of large volumes of data in batches. Speed layer only processes current data to update the services layer's high latency. Because frequent updates would create delay, this layer only updates once and is used for initial processing.

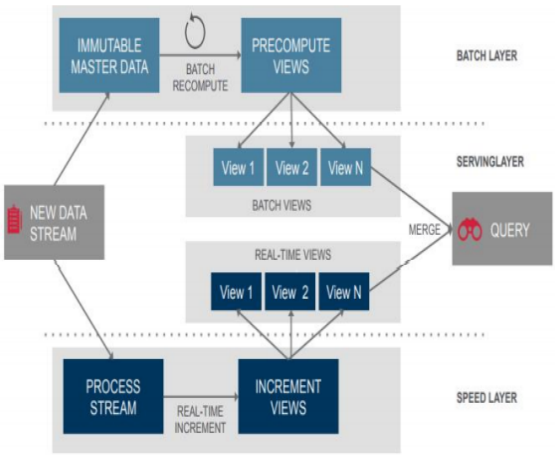

Fig. 3 Lambda architecture [34].

Map Reduce is a type of batch processing that may be utilized in a variety of situations [41]-[42]. The Serving layer then computes in real-time, with the ultimate objective of reducing latency by properly completing real-time computations as soon as data arrives.

The lambda design reveals a number of issues [43]:

In the real-time and batch layers, the business logic is implemented twice. In a distributed system, the code was generated twice [44].

When Lambda does asynchronous processing correctly, the outputs are not immediately comparable to contemporary data.

The outcome of a sophisticated Operational is massive, and it takes a long time to complete.

Managing two operating systems for batch and speed layers is a significant operational burden.

B. Kappa Architecture

Kappa architecture [45] is a lambda architecture simplification. Jay Kreps created it in 2014. A Kappa architecture is similar to a lambda design, but without the batch processing mechanism. Data is sent fast using the streaming technology [45]. Append-only approach for storing data in a Kappa Architecture in a permanent log, rather than using a relational database like SQ. Data is fed to computation from this log and then communicated to supplemental storage for serving [41].

Kappa differs from lambda in that it uses only one piece of code for two levels, reducing system complexity [45]. Figure 13 shows the Kappa design, which is made up of two layers: a stream processing layer that conducts stream processing tasks and a serving layer for querying the results [4].

Customers may build their systems on top of the processing framework, which benefits Kappa while also satisfying the requirement for data integrity and streaming processing because real-time processing is possible. The Kappa architecture was built using Storm, Spark, and Kafka [47].

C. Delta Architecture

Delta architecture often consists of numerous processing layers, each of which has various processing components that are independent of one another. Layers function in a loosely connected fashion as both consumers and producers.

For example, Kafka may be used at this tier to create a low-latency append messing system. Logs store a large amount of data in a natural sequential sequence over a period of time. At the time, the phrase "communication medium" meant a specific implementation of a log [48].

The log stores data from a number of sources, allowing for quick query responses and data manipulation for various analysis objectives. Although query-answering is time-saving, data from the store does not allow for real-time analysis like data from the log. [48] Although query-answering services for data stores might be useful, we must remember that unlike logs, data from stores cannot be utilized for real-time analysis.

When compared to the delta design, the main distinction between lambda and kappa architectures is that.

Delta is a data lake, which means it gathers all of the data into one convenient location. The Lambda architecture is used for long-term batch data processing. Though the lambda design employs HDFS as a data lake, the data lake's most important feature is immutability. This might be viewed as a fault in the batch layer's data processing [49].

Any new streaming data is treated as an incremental record rather than as a new record in the Delta architecture. Furthermore, deal with patterns similar to Lambda, with the exception that Delta does not consider the data lake to be immutable, and batch processing can quickly alter the data lake's current data structures [50].

The delta symbol was also used to represent gradual change. When new data is produced, modified, or streamed since the last processing time, incremental changes in data processing occur organically. With current technology, file systems directly handle CRUD (Create-Read-Update-Delete) activities.

The issue is that the HDFS-based data lake is designed with immutability, which results in performance overhead in the

layers [51].

Data Lake is considered immutable by the Delta architecture, which treats incoming data as delta records rather than append-only contemporary records.

V. Discussion

Lambda, Kappa and Delta are three architectural styles covered in this study. Architecture of big data processing is critical for real-time analytics. Lambda's architecture is made up of three levels: layer of batch, layer of query and layer of querying. The batch and speed layers' results are merged. And the stratum of speed: To compensate for the significant latency of the services layer updates, only the most current data is processed.

In the real-time and batch layers, the business logic is implemented twice. The programmers must maintain code on two different distributed systems. It manages the Master Dataset, which is a set of immutable, append-only, and exclusive raw data, as well as batch views. Kappa Architecture is a permanent append-only log. Data is streamed from the log to a computing system and then transferred to auxiliary storage for serving the log. Kappa uses only one code route, which decreases system complexity and performance. It does not need a batch layer to prove findings from a real-time layer.

The table below compares and contrasts the three architectures.

Table I: Compares and Contrasts of the Three Architechtures

Criteria	Lambda	Kappa	Delta
Architecture	Immutable	Immutable	none immutable
Layers	Real-time layer, batch, and serving	Serving and stream processing layer	Several layers
Processing data	Real and Batch	real-time	Batch and real time
Processing guarantees	batch mode, but approximate streaming mode approximate streaming mode	With consistency exactly once.	batch mode, but approximate streaming mode
Re-processing paradigm	In every batch cycle	Just when code change	In every layer
Scalability	yes	yes	yes
fault tolerance	yes	yes	yes
permanent storage	yes	yes	yes
Real-time	It is not correct	Accurate	Not accurate

VI. Conclusion

Because of the sophisticated calculations of historical data collected, as well as technical improvements and the situation of everyday life, real-time big data analytics is critical. The need for rapid access to information and the ability to make decisions is expanding. Allow society to reap the rewards and make timely, well-informed decisions. A well-conducted literature survey benefits from real-time data analytics. Future researchers and data scientists will be able to improve their businesses and technologies by categorizing real-time data analytics methods according to their various areas of application and tools used by categorizing real-time data analytics methods according to their various areas of application and tools used.

Processing and analysing a vast volume of data in real time (data stream) while balancing throughput, reaction time, and data quality has become a significant issue with the recent emergence of real time data streaming and big data.

The three designs are more concerned with managing performance difficulties by balancing throughput and reaction time than with data quality issues and data analysis findings.

The map reduction technology in Hadoop provides a distributed computing platform for processing huge datasets on larger clusters; however it does not support real-time processing. A robust system that satisfies real-time specification standards is necessary to overcome the constraints of the old system.

Data stream technology was explored in this paper to evaluate which application fields can benefit from real-time big data applications, as well as how to organize, manage, analyse data streams. To build effective real-time big data applications, a number of difficulties must be addressed, including real-time event (data) transfer, real-time scenario (exceptions) recognition, real-time decision making, and executing real-time responses. The first step in creating more effective and efficient apps in this industry is to thoroughly appreciate the complex nature of these apps and the challenges that lie ahead.

The development and usage of real-time big data applications will be greatly aided by successful ways to addressing these difficulties, which will give several benefits such as saving lives, improving quality of life, decreasing risks, and increasing profitability.

This paper's main purpose is to provide a real-time processing and analytic architecture for real-time large data analytics. Apache Storm helps you to process large amounts of data quickly and efficiently. In addition, a variety of open-source Real-time processing technologies, like as Hadoop, Storm, and Spark, is frequently used in this design.

References

[1] Laney D. 3D data management: Controlling data volume, velocity and variety. *META group research note.* 2001; 6(70): 1.

[2] Gantz J, Reinsel D. Extracting value from chaos. IDC iview. 2011; 1142(2011): 1-2.

[3] Hariri RH, Fredericks EM, Bowers KM. Uncertainty in big data analytics: survey, opportunities, and challenges. *Journal of Big Data*. 2019; 6(1): 1-6.

[4] Data IB, Hub A. Extracting business value from the 4 V's of big data. 2016; 19: 2017.

[5] Snow D. Dwaine Snow's Thoughts on Databases and Data Management. 2012.

[6] Gandomi A, Haider M. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*. 2015; 35(2): 137-44.

[7] Pokorný J, Škoda P, Zelinka I, Bednárek D, Zavoral F, Kruliš M, et al. Big data movement: a challenge in data processing. InBig Data in Complex Systems 2015: 29-69.

[8] Chen M, Mao S, Liu Y. Big data: A survey. *Mobile Networks and Applications*. 2014; 19(2): 171-209.

[9] Bakshi K. Considerations for big data: Architecture and approach. In2012 IEEE aerospace conference 2012: 1-7.

[10] Elgendy N, Elragal A. Big data analytics: a literature review paper. InIndustrial conference on data mining 2014: 214-227.

[11] Plattner H, Zeier A. In-memory data management: technology and applications. Springer Science & Business Media; 2012.
[12] Watson HJ. Tutorial: Big data analytics: Concepts, technologies, and applications. *Communications of the Association for Information Systems.* 2014; 34(1): 65.
[13] Zhang L, Stoffel A, Behrisch M, Mittelstadt S, Schreck T, Pompl R, et al. Visual analytics for the big data era—A comparative review of state-of-the-art commercial systems. In2012 IEEE Conference on Visual Analytics Science and Technology (VAST) 2012: 173-182.
[14] Elgendy N. Big Data Analytics in Support of the Decision Making Process. M. S. Thesis. German University. 2013.
[15] He Y, Lee R, Huai Y, Shao Z, Jain N, Zhang X, et al. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In2011 IEEE 27th International Conference on Data Engineering 2011: 1199-1208.
[16] Tallat R, Latif RM, Ali G, Zaheer AN, Farhan M, Shah SU. Visualization and Analytics of Biological Data by Using Different Tools and Techniques. In2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST) 2019: 291-303.
[17] Manyika J, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Hung Byers A. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute. 2011.
[18] Shen Z, Wei J, Sundaresan N, Ma KL. Visual analysis of massive web session data. InIEEE symposium on large data analysis and visualization (LDAV) 2012: 65-72.
[19] Mohamed S, Ismail O, Hogan O. Data equity: Unlocking the value of big data. London, UK: Centre for Economics and Business Research. 2012.
[20] Unit EI. The deciding factor: Big data & decision making. Capgemini Reports. 2012: 1-24.
[21] Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. *Journal of Big Data*. 2015; 2(1): 1-21.
[22] Qiu J, Wu Q, Ding G, Xu Y, Feng S. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*. 2016; 2016(1): 1-6.
[23] Kaur N, Singh G. A Review Paper on Data Mining And Big Data. *International Journal of Advanced Research in Computer Science.* 2017; 8(4).
[24] Rao JN, Ramesh M. A Review on Data Mining & Big Data. Machine Learning Techniques. *Int. J. Recent Technol. Eng*. 2019; 7: 914-6.
[25] Tseng FM, Hu YC. Comparing four bankruptcy prediction models: Logit, quadratic interval logit, neural and fuzzy neural networks. *Expert Systems with Applications*. 2010 Mar 15; 37(3): 1846-53.
[26] Ratra R, Gulia P. Big data tools and techniques: A roadmap for predictive analytics. *International Journal of Engineering and Advanced Technology (IJEAT).* 2019; 9(2): 4986-92.
[27] Marcu OC, Costan A, Antoniu G, Pérez-Hernández M, Tudoran R, Bortoli S, et al. Storage and Ingestion Systems in Support of Stream Processing: A Survey. Ph. D. Thesis. INRIA Rennes-Bretagne Atlantique and University of Rennes 1.
[28] Etzion O, Niblett P. Event Processing in Action, Stamford.
[29] Linington PF, Milosevic Z, Tanaka A, Vallecillo A. Building enterprise systems with ODP: an introduction to open distributed processing. CRC Press; 2011.
[30] Luckham DC. Event processing for business: organizing the real-time enterprise. John Wiley & Sons; 2011.
[31] Milosevic Z, Chen W, Berry A, Rabhi FA, Buyya R, Calheiros RN, Dastjerdi AV. Real-time analytics. *Big Data: Principles and Paradigms.* 2016: 39-61.
[32] Murphy BM, O'Driscoll C, Boylan GB, Lightbody G, Marnane WP. Stream computing for biomedical signal processing: A QRS complex detection case-study. In2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) 2015: 5928-5931.
[33] Brown PC. Architecting Complex-Event Processing Solutions with TIBCO®. Addison-Wesley; 2013.
[34] Margara A, Cugola G, Tamburrelli G. Learning from the past: automated rule generation for complex event processing. InProceedings of the 8th ACM international conference on distributed event-based systems 2014: 47-58.
[35] Ellis B. Real-time analytics: Techniques to analyze and visualize streaming data. John Wiley & Sons; 2014.
[36] Chrysos G, Papapetrou O, Pnevmatikatos D, Dollas A, Garofalakis M. Data stream statistics over sliding windows: How to summarize 150 Million updates per second on a single node. In2019 29th International Conference on Field Programmable Logic and Applications (FPL) 2019: 278-285.
[37] Braud RE. Query-based debugging of distributed systems. University of California, San Diego; 2010.
[38] Traub J, Grulich PM, Cuéllar AR, Breß S, Katsifodimos A, Rabl T, et al. Efficient Window Aggregation with General Stream Slicing. InEDBT 2019; 19: 97-108.
[39] Grimaila MR, Myers J, Mills RF, Peterson G. Design and analysis of a dynamically configured log-based distributed security event detection methodology. *The Journal of Defense Modeling and Simulation.* 2012; 9(3): 219-41.
[40] Chen W, Rabhi FA. Enabling user-driven rule management in event data analysis. *Information Systems Frontiers*. 2016; 18(3): 511-28.
[41] Lassinantti J, Ståhlbröst A, Runardotter M. Relevant social groups for open data use and engagement. *Government Information Quarterly*. 2019; 36(1): 98-111.
[42] Zaharia M, Das T, Li H, Hunter T, Shenker S, Stoica I. Discretized streams: Fault-tolerant streaming computation at scale. InProceedings of the twenty-fourth ACM symposium on operating systems principles 2013: 423-438.
[43] 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): 1-36.
[44] Ounacer S, Talhaoui MA, Ardchir S, Daif A, Azouazi M. A new architecture for real time data stream processing. *International Journal of Advanced Computer Science and Applications*. 2017; 8(11): 44-51.
[45] Rahman H, Begum S, Ahmed MU. Ins and outs of big data: A review. InInternational Conference on IoT Technologies for HealthCare 2016: 44-51.
[46] Mohammed EA, Far BH, Naugler C. Applications of the MapReduce programming framework to clinical big data analysis: current landscape and future trends. *BioData mining.* 2014; 7(1): 1-23.
[47] Khezr SN, Navimipour NJ. MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research. *Journal of Grid Computing*. 2017; 15(3): 295-321.
[48] Lakhe B, Lakhe. Practical Hadoop Migration. Berkeley: Apress; 2016.
[49] Grover P, Kar AK. Big data analytics: A review on theoretical contributions and tools used in literature. *Global Journal of Flexible Systems Management*. 2017; 18(3): 203-29.
[50] Kreps J. Questioning the lambda architecture. Online article, July. 2014; 205.
[51] Salloum S, Dautov R, Chen X, Peng PX, Huang JZ. Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*. 2016; 1(3): 145-64.