IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 June 2023, accepted 12 July 2023, date of publication 14 July 2023, date of current version 19 July 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3295694*

SURVEY

# Real-Time Analytics: Concepts, Architectures, and ML/AI Considerations

**WEISI CHEN[1], (Member, IEEE), ZORAN MILOSEVIC[2,4], (Senior Member, IEEE), FETHI A. RABHI[3], AND ANDREW BERRY[2]**

[1]School of Software Engineering, Xiamen University of Technology, Jimei, Xiamen, Fujian 361024 China
[2]Deontik, Brisbane, QLD 4032, Australia
[3]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia
[4]Best Practice Software, Brisbane, QLD 4000, Australia

Corresponding author: Weisi Chen (chenweisi@xmut.edu.cn)

This work was supported in part by the Fujian Provincial Natural Science Foundation of China under Grant 2022J05291, and in part by the Xiamen Scientific Research Funding for High-Level Overseas Chinese Scholars.

**ABSTRACT** With the advancement in intelligent devices, social media, and the Internet of Things, staggering amounts of new data are being generated, and the pace is continuously accelerating. Real-time analytics (RTA) has emerged as a distinct branch of big data analytics focusing on the velocity aspect of big data, in which data is prepared, processed, and analyzed as it arrives, intending to generate insights and create business value in near real-time. The objective of this paper is to provide an overview of key concepts and architectural approaches for designing RTA solutions, including the relevant infrastructure, processing, and analytics platforms, as well as analytics techniques and tools with the most up-to-date machine learning and artificial intelligence considerations, and position these in the context of the most prominent platforms and analytics techniques. The paper develops a logical analytics stack to support the description of key functionality and relationships between relevant components in RTA solutions based on a thorough literature review and industrial practice. This provides practitioners with guidance in selecting the most appropriate solutions for their RTA problems, including the application of emerging AI technologies in this context. The paper discusses the complex event processing technology that has influenced many recent data streaming solutions in the analytics stack and highlights the integration of machine learning and artificial intelligence into RTA solutions. Some real-life application scenarios in the finance and health domains are presented, including several of the authors' earlier contributions, to demonstrate the utilization of the techniques and technologies discussed in this paper. Future research directions and remaining challenges are discussed.

**INDEX TERMS** Real-time analytics, data streaming, big data analytics, complex event processing, machine learning.

## I. INTRODUCTION

### A. BACKGROUND

With the recent advancement in intelligent devices, social media, sensor networks, and the Internet of Things (IoT), staggering amounts of new data are being generated every second, and the pace is continuously accelerating. This has led to fast-evolving developments in big data analytics in which data is turned into insights by preparing, processing, and analyzing data as it arrives, with the continuing advent of novel technologies and techniques to deal with the challenges posed by big data. It has been asserted that 'the fresher the data, the more valuable it is,' especially for data-driven platforms like Uber [1]. This need has driven the development of real-time analytics (RTA), a distinct branch of big data analytics emphasizing the velocity aspect of big data. It is increasingly prominent in both academia and industry due to the growing demand to deal with the proliferation of incoming data and event streams and the need to take immediate or near-immediate action in response to data triggers. Some examples of RTA include real-time financial data quality control, real-time decision-making in financial trading, real-time

The associate editor coordinating the review of this manuscript and approving it for publication was Chong Leong Gan.

health-related data monitoring from wearable devices, and real-time advertisement recommendation to promote sales. RTA is sometimes referred to as fast data analytics [2], the nature of which has introduced challenges in processing it in real-time, particularly when applying complex data analysis techniques.

Other increasingly prevalent disciplines related to RTA are artificial intelligence (AI) and machine learning (ML), which have been extensively employed in various areas in the big data era. In particular, ML is a data-driven approach to knowledge and insight discovery which has achieved remarkable success in recent years, particularly in well-known fields like computer vision and natural language processing. Large-scale data collection, model or function identification through training, and prediction (e.g., classification, regression, or clustering) of fresh data are all components of ML. Traditional techniques, including regression algorithms, decision trees, random forests, and support vector machines (SVM), are examples of ML models. More sophisticated artificial neural network-based deep learning (DL) models are also available, such as multi-layer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), autoencoders (AE), and the emerging large language models (LLM). ML and DL models have been extensively employed in big data analytics on almost all fronts of academia and industry, for instance, computer vision [3], natural language processing [4], finance [5], and healthcare [6].

There is an overwhelming volume of research work conducted on big data analytics or ML/AI across many disciplines in recent years as well as many high-profile, real-life applications. However, the combination of real-time analytics with ML/AI techniques is still insufficiently studied in the literature. Commonly, practitioners find it challenging to get started with the design and implementation of RTA software solutions, especially when the integration of ML/AI technologies into RTA solutions is required.

### B. RESEARCH OBJECTIVES AND CONTRIBUTIONS

The objective of this paper is to provide an overview of key concepts and architectural approaches for designing RTA solutions, including the relevant infrastructure, processing, and analytics platforms, as well as analytics techniques and tools with the most up-to-date ML and AI considerations and to position them in the context of the most prominent platforms and analytics techniques in the recent decade.

This paper attempts to answer the following motivating research questions:

RQ1: In accordance with recent research and industrial practice, what are the key concepts, prominent software architectures, techniques, and tools available for RTA solutions?

RQ2: How can software engineering practitioners design and implement their own RTA solutions?

RQ3: What are the methods to integrate ML or AI into RTA solutions?

The contributions of this article are as follows:

- Providing an overview of key concepts and architectural approaches for designing RTA solutions.
- Developing a logical analytics stack to support the description of key functionality and relationships between relevant components in RTA solutions based on a thorough literature review and industrial practice. The most up-to-date solutions are positioned in relation to various stack layers.
- Highlighting mechanisms for integrating RTA with ML and AI.
- Presenting some real-life case studies in finance and health domains to demonstrate the utilization of the techniques and technologies discussed.
- Discussing key challenges and future research directions.

### C. STRUCTURE OF THE PAPER

This review paper is structured as follows. Section II describes the process of literature selection according to the scope of this review and provides some quantitative and qualitative analysis of the selected literature. Section III explains the key concepts and requirements of RTA solutions, which sets the scene for subsequent sections. Section IV introduces a hierarchy of abstractions called the analytics stack for big data based on our earlier work presented in [7]. Each layer of the stack is described in the following sections: Section V describes real-time data processing platforms for big data environments, referred to as data stream processing platforms; Section VI explains data stream analytics platforms that can utilize analytics techniques against real-time data streams; Section VII discusses the key analytics tools that provide specific data analytics solutions based on the platforms discussed in Sections IV and V, with a focus on ML and AI integration considerations. Section VIII describes case studies in the finance and health domains that use specific analytics platforms, techniques, and tools. Section IX discusses the lessons learned, remaining challenges, and future research agenda, and finally, Section X concludes the paper.

## II. RELATED WORK

In the discovery phase of this review, we have adopted the systematic literature review methodology, aiming to develop and implement a rigorous and repeatable process that will give a thorough and unbiased review of the body of literature already in existence. The digital library used is the EI Compendex database provided by Engineering Village. The rationale for this selection is that EI Compendex includes information from a number of leading repositories, such as IEEE and ACM journals, and focuses on engineering research aspects, fitting the scope of our paper. It also holds high standards by indexing trustworthy and peer-reviewed sources. We have collected journal articles on real-time analytics

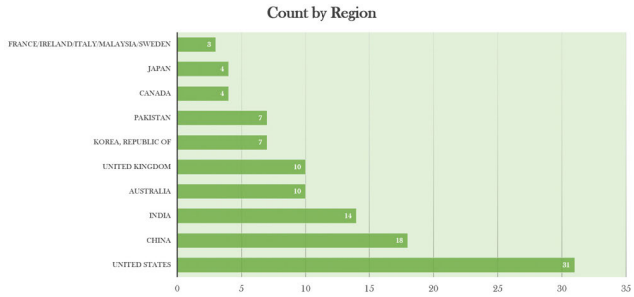


**FIGURE 1.** Selected journal papers by region (Top regions with more than two papers published).

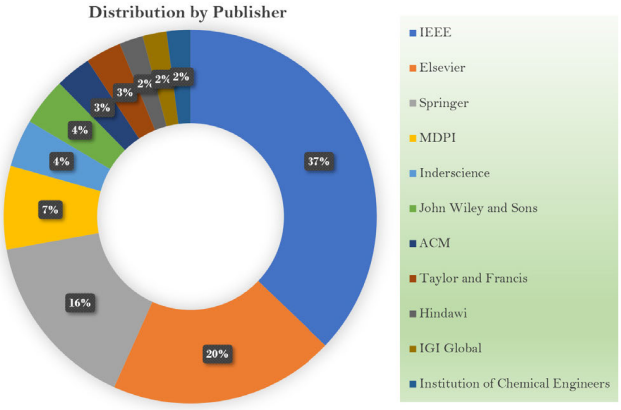


**FIGURE 2.** Selected journal papers by publisher (Top publishers with more than one paper published).

(RTA) within the EI Compendex database by Engineering Village in the last decade (from 2014 to 2023) written in English, and 198 articles were identified. Further, we have limited the publication classification to computer science and technology-related topics only, resulting in 108 articles. Figure 1 and Figure 2 describe the distribution of these 108 publications based on the region and the publisher, respectively. To be specific, the United States, China, India, and the United Kingdom have published the most papers in recent years on RTA. IEEE, Elsevier, Springer, MDPI, Wiley, and Inderscience are the leading publishers that have published the most RTA-related papers in recent years.

Figure 3 shows the yearly number of publications of these 108 papers, which presents an upward trend of RTA studies in recent years, especially from 2017, indicating the increasing prominence of RTA. Note that the year 2023 is not complete at the time of writing (17 June 2023). This figure also shows the number of related review papers by year next to the total number of publications.

Most of the selected papers are research papers concentrating on concrete real-time data analysis tasks and applications in various areas, such as IoT, smart cities, and health. The objective of such research is commonly analyzing certain data and generating insights and conclusions from the data in real-time. Interestingly, very few of them discuss the software

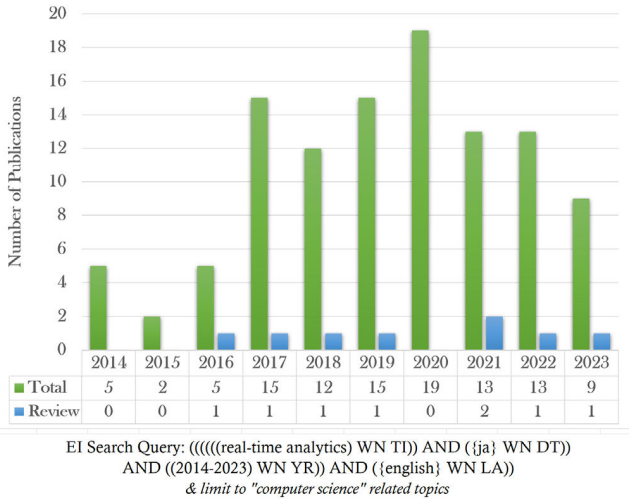


**FIGURE 3.** Selected journal papers by year (2014-2023).

design, architecture, and implementation of RTA solutions, and most of these papers focus on the ''analytics'' element rather than the ''real-time'' element. Among these publications, there are eight review/survey papers related to RTA, but with different focuses. Table 1 lists and compares these articles.

As can be seen from Table 1, each review article in recent years has focused on a unique aspect of utilizing RTA. None of these reviews have details about real-life scenarios in practice. A review of the most up-to-date software architecture articles describing components and techniques that can serve as a guide for practitioners is still lacking. To the best of our knowledge, there is no such literature focusing on the practical software architecture and implementation aspects of RTA, with a thorough discussion on how ML and AI can be integrated into RTA platforms. Thus, the aforementioned three research questions in Sections I-B are yet to be answered.

In addition to the review papers, the rest of the paper will primarily focus on 72 of the selected papers (8 review papers and 64 research papers) in the last five years (2019-2023) to reflect the most up-to-date technologies. Table 2 lists all 72 publications.

In the next section, we will review the current state-of-the-art RTA solutions in the most recent literature, from a software perspective, with practicality and the integrations of ML/AI in mind, attempting to address the three defined research questions. In addition, our review takes industrial practice into account, which further distinguishes our contribution from the existing literature.

## III. CONCEPTS AND REQUIREMENTS FOR REAL-TIME ANALYTICS

This section introduces key concepts and requirements for real-time analytics (RTA) solutions in a big data context and sets the scene for discussion in the subsequent sections.

**TABLE 1.** Review articles in recent years.

| Ref | Year | Content | Perspective |
|---|---|---|---|
| [8] | 2016 | A review covering real-time data analytics algorithms for data collection, data analysis, and response access. | Algorithms |
| [9] | 2017 | A survey on RTA of massive IoT data, with a focus on network methodologies. | Network |
| [10] | 2018 | A review of IoT big data analytics, with a focus on IoT aspects such as sensors, flexibility, security, and privacy. | IoT |
| [11] | 2019 | A survey on real-time anomaly detection techniques. | Anomaly detection |
| [12] | 2021 | A review of RTA technologies, especially data stream mining platforms, and paradigms, from a user's perspective. | User |
| [13] | 2021 | A review of emerging wearable systems for real-time sensing and analytics. | Wearable Systems |
| [14] | 2022 | A review of real-time visual analytics for in-home medical rehabilitation of stroke patients. | Medical visual analytics |
| [15] | 2023 | A review of RTA methods, especially infrastructure and database technologies, with a discussion on their benefits and limitations. | Infrastructure and commercial solutions |

**TABLE 2.** Reviewed literature in this paper.

| Type | References |
|---|---|
| Review Articles | [8-15] |
| Research Articles | [3, 16-78] |

## A. KEY CONCEPTS

### 1) EVENT, EVENT PATTERN, AND STREAMS

The concept of an event plays a fundamental role in RTA systems. An event is a timestamped record that represents an environment change in the system. Many real-time systems detect and report events using sensors, e.g. [40], [72], followed by real-time analysis of the reported events to identify patterns denoting certain opportunities or threats. To enhance the performance, some applications distribute event data across multiple processors to enable parallel computation and horizontal scaling, e.g., [79]. Note that the timestamp of event data usually signifies the time of event occurrence or, in some cases, the time of arrival at the processing system. The former is preferred, particularly in distributed systems where event data can be delayed or lost. Events arriving at a processing system are most often referred to as a data stream. In many scenarios, event data streams are akin to time series data, so time series processing techniques can be adapted to the handling of event data [26].

For ease of computation and processing in RTA applications, an event can be described by an event type encapsulating common attributes of events, typically including the timestamp and the nature of various data items associated with the event payload. Event type can also describe the wire format of a message emitted by a physical sensor or associated with a particular messaging system.

An event pattern is one of the pivotal concepts adopted in many current RTA solutions, particularly complex event processing, which will be discussed in the next subsection. An event pattern abstracts relationships between events. Real-life examples of event patterns include unusual structuring of deposits of large amounts of money into multiple small transactions at banks, indicating a possible money laundering activity; unusual pathology orders in eHealth systems; and 'pairs trading,' meaning two stocks with a high correlation between rise and fall. Akin to event type, RTA applications require the definition of event pattern types to describe and detect occurrences of event patterns in the environment.

### 2) COMPLEX EVENT AND EVENT STREAM PROCESSING

Event processing generally means performing operations on events as they are fed into the system that proactively monitors the ingress of events from the environment. Common operations on event data may include reading, creating, integrating, transforming, and processing events [80]. Event integration from multiple sources generally involves an ETL (Extraction, Transformation, and Loading) process [81] used in both on-premise and cloud environments.

Early research efforts in event processing were driven by the need to simulate event applications and distributed systems, e.g., the Stanford Rapide project. Some of these efforts were later used for the construction of distributed applications using event-driven approaches [7]. Another type of application involves different sites processing a stream over an extended time. These streams may be a continuous flow of text, image, audio, and video or a sequence of data related to real-life events, which could be emitted by devices like sensors or generated by software systems like a stock ticker. The processing of these streams is commonly defined as data stream processing or event stream processing.

More recent research into this space has addressed the need to perform more complex computations over a vast number of events from a number of data sources, aiming to find relationships (e.g., temporal, causal relationships) between events, interpret the implications, and generate insights. This more complicated event processing is generally referred to as complex event processing (CEP) [82], [83]. While event stream processing or data stream processing typically focuses

on handling one single event stream, which can be considered a simplified form of CEP, in practice, these three concepts are often used interchangeably. This paper will not further distinguish them from each other and will primarily use the term CEP.

CEP features abstractions of event processing logic, which is, in most cases, separated from the application components (event producers and event consumers). The benefit of this separation is reduced cost of development and maintenance. Event processing logic is often expressed in domain-specific languages called event processing languages (EPLs), which define event pattern types to identify and instruct the operations on events. A more detailed discussion of CEP systems is presented in Section VI.

### B. REQUIREMENTS OF REAL-TIME ANALYTICS SOLUTIONS

In IT systems, the term real-time emphasizes the requirement to handle events as they arrive, typically within a very short time interval. Some researchers focus on ensuring fast, deterministic response times in real-time systems, especially for safety-critical systems. However, the requirement for deterministic timing can often be relaxed in big data analytics, and this is referred to as ''near real-time''. In this paper, we adopt the more relaxed requirement. In either case, real-time systems often involve generating notifications to users about identified occurrences of pre-defined significant events or event patterns and/or invoking internal and external functions of the system for some further action.

Three key requirements for RTA are low latency, high availability, and horizontal scalability [84]:

1) Low latency is the core requirement of an RTA solution to respond to events within very short time limits, usually within milli-, micro-, or even nano-seconds. Two measures of latency are of interest in real-time systems, namely data latency, meaning the delay to access fresh data, and processing latency, meaning the delay of its processing after an event arrives or occurs. The latter includes network latency and computing latency. To minimize data latency, flash technology, and data fetching with anticipation from multiple data streams can be adopted. To minimize processing latency, in-memory processing, incremental evaluation, and parallel processing in the high-performance computing world can be applied. It is worth noting that a recent survey has revealed the importance of time predictability, i.e., the accuracy of predicting the overall performance of a real-time system in addition to the ''low latency'' requirement. To achieve it, multiple methods have been applied, such as watchdog timers, run-time monitors, static schedules, and time partitions [16].
2) High availability means to what extent a system performs its function as required. This requirement's goal is to eliminate single points of failure, so part of this requirement is sometimes referred to as high fault tolerance. A system for RTA without high availability will leave arriving events unprocessed for a time interval that is unacceptable for the analytics needs. Various strategies are used to ensure high availability for effective RTA, including distributing the tasks to multiple nodes (one node taking over in case of failure of another), data backup (storing data on multiple servers in case one machine fails), and redundant processing (multiple nodes processing the same data).
3) Horizontal scalability is, in general, the capability of adding resources to the existing pool for the purpose of capacity increase and performance improvement. For RTA, it is of great importance as this allows systems to dynamically add additional nodes or machines as the volume of data or processing workload increases over time, especially when the rate of data ingress is unpredictable.

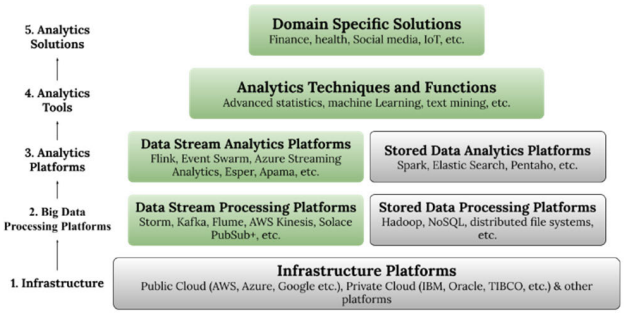


**FIGURE 4.** The real-time analytics stack.

## IV. REAL-TIME ANALYTICS STACK

In this section, a layered classification of real-time analytics (RTA) solutions, informally referred to as the analytics stack [7], is introduced. It allows the positioning of RTA with the big data analytics context and facilitates discussion on various analytics infrastructures, platforms, techniques, and technologies to be described in the subsequent sections. Figure 4 shows the details of the analytics stack. Note that the topics in the stack related to RTA are highlighted in green. Other topics, such as stored/static data analytics, have been included in the figure for completeness only and will not be discussed in detail beyond this section.

The first layer of the stack identifies infrastructure platforms, including public and private cloud PaaS offerings by enterprise vendors such as Amazon, Microsoft, IBM, and Oracle. They increasingly support container orchestration technology such as Kubernetes [85], which is particularly useful in supporting high availability and horizontal scaling. There are other dedicated infrastructure platforms for constructing industrial real-time systems in various application areas such as utility, transportation, and mining. This layer is not specific to RTA and has mature but evolving technologies across academia and industry.

Big data processing platforms, highlighted in the second layer, are software platforms operating on the resources offered by the underlying infrastructure layer and used to process big data, including both stored data and data streams. We distinguish the processing of 'data at rest' (i.e., stored data) from 'data in motion' (i.e., fast data or data streams), as displayed separately in the figure. The stored data processing platforms, including Hadoop, Spark, and NoSQL databases, provide processing tools and capabilities for static data, whereas the data stream processing platforms provide tools and abstractions for processing data as it arrives or becomes available and are the foundation for RTA.

The third layer, termed analytics platforms, denotes a special type of big data processing software platform that takes advantage of the functionalities offered by Layer 2. The objective of Layer 3 is to support analytics applications, and a typical contemporary analytics platform includes components to support data collection, data storage, data preprocessing (for handling incorrect data and normalizing them in a required standard format), database management, and statistical analysis. Akin to the second layer, we distinguish stored data analytics platforms, like ElasticSearch [86] and Pentaho [87], from data stream analytics platforms.

The fourth layer, analytics techniques, and tools, contains high-level methods and solutions for specific analytics problems. Examples include algorithms or libraries from the areas of advanced statistics, machine learning, and text mining. It should be noted that the implementation of these methods should comply with the specific type of the layer below, i.e., whether it is for real-time data or stored data.

The fifth layer denotes the highest-level domain-specific solutions in various application areas, such as finance, health, social media, and IoT. The solutions are designed to take domain knowledge and expertise into consideration on top of general analytics techniques in the fourth layer.

Note that many analytics solutions can hardly be positioned at one single layer, with some of them, such as Azure and AWS architectures, offering a full-stack solution to satisfy the requirements of multiple layers.

For instance, Figure 5 [88] displays an end-to-end architecture example involving a selection of services and products on the Azure infrastructure, referred to as Azure Stream Analytics. We use this as a reference architecture to illustrate the layers discussed above. This architecture handles a number of data types, including streaming data and static data stored in various databases, which can also be used as input into various ML models. It involves five principal phases for both streaming data and stored data, namely Ingest, Store, Process, Enrich, and Serve. For RTA, the Ingest phase uses Azure Event Hubs and Azure IoT Hubs as "Data Stream Processing Platforms" (Layer 2), and Azure Stream Analytics serves as the "Data Stream Analytics Platform" (Layer 3) in the Process phase. The Enrich phase has AI services, including Azure Cognitive Services and Azure Machine Learning as "Analytics Techniques and Functions" (Layer 4). Services

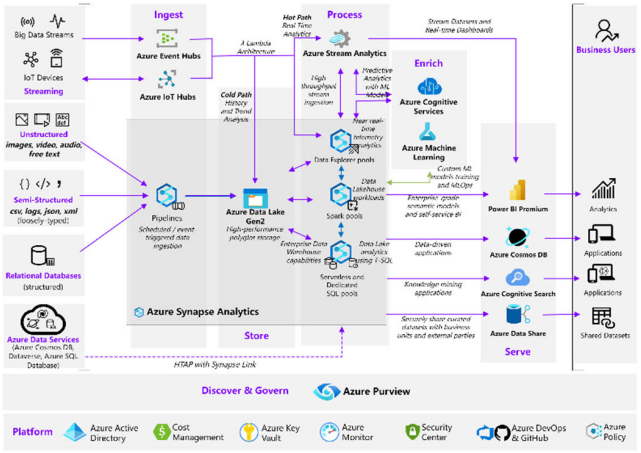


**FIGURE 5.** An example of end-to-end architecture involving Azure Stream Analytics.

in the Serve phase can be mapped into Layer 5, the Analytics Solutions.

More explanation and discussion of the key layers (Layers 2-4) will be provided in the following three sections.

## V. DATA STREAM PROCESSING PLATFORMS (LAYER 2)

In this section, we discuss current platforms used for data stream processing (Layer 2 of the analytics stack, as shown in Figure 4). Many of these platforms are open-source and enable the construction of real-time applications, either message-oriented or event-driven. They read messages or events with minimal latency, feed them for processing and promptly generate alerts. They are primarily based on nodes in distributed or cloud environments, mostly focusing on processing event-driven data streams. They also serve as the foundation for the construction of Layer 3 on top.

It should be noted that there are two categories of data stream processing, namely micro-batching and native streaming. The former takes incoming events every few seconds and processes them in mini-batches with consequent batch latency, whereas the latter adopts an event-driven approach where each incoming event is processed as soon as it arrives with minimal latency.

### A. APACHE HADOOP

Hadoop [89] offers a framework including a group of Apache projects that handle big data, in general, in a distributed manner. It was originally developed and used as a batch processing system, so it does not intrinsically meet real-time performance requirements. However, it can be extended by integrating a dedicated real-time component such as Apache Flume, described in Section V-D, to satisfy the real-time demand. In recent years, the Hadoop ecosystem (i.e., a set of tooling in support of building Hadoop-based applications) has become increasingly comprehensive and powerful with the enhancement and development of advanced modules. The key modules within the Hadoop ecosystem are shown in Figure 6.

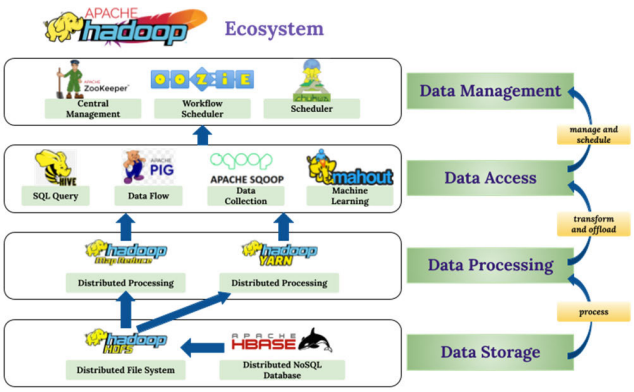


**FIGURE 6.** Apache Hadoop ecosystem.

Babar and Arif [71] is an example of using the Hadoop ecosystem for a real-time analytics (RTA) solution. Some key modules that are of interest to this paper are described as follows:

- HDFS (Hadoop Distributed File System) is a distributed file system that manages vast volumes of data on large clusters of computers;
- MapReduce, the distributed data processing module for the ecosystem, facilitates parallel computation, dividing bigger jobs into sub-tasks and processing them in a distributed manner. It is known for its linear scalability, meaning that it may take no more time to handle increased amounts of data with additional nodes or servers;
- YARN (Yet Another Resource Negotiator), split from the old version of MapReduce, separating the resource management function, has resolved some scalability drawbacks of the earlier MapReduce;
- Hive, a distributed data warehouse that offers an SQL-like interface for data queries and analysis;
- Pig is a tool for dealing with large datasets represented as dataflows. Pig features a high-level dedicated language named Pig Latin. It runs upon HDFS and MapReduce;
- ZooKeeper is a centralized operational management service with high availability, featuring a distributed configuration service, a synchronization service, and a naming registry for distributed systems.

### B. APACHE KAFKA

Kafka [90] is a unified event streaming platform for processing real-time data streams that meets all the requirements for RTA, as mentioned in Section III-B. Kafka is commonly known as a message broker that can be used as a Message Queuing (MQ) system, allowing for message addition to the queue without waiting for a response. Another example of an MQ system is RabbitMQ. However, Kafka can also be used for stream processing as a distributed streaming platform.

Kafka features the following major component concepts:

- Topic: a data stream to be processed;
- Broker: a server on a Kafka cluster that holds multiple topics with different partitions to ensure availability and horizontal scalability;
- Consumer: the component that reads topics (data streams) from the cluster;
- Producer: the component that writes into topics;
- Zookeeper: borrowed from Hadoop for processing node coordination.

An example found in recent publications is [75], which deploys a deep-learning weather prediction model based on the LSTM neural network that continuously learns using weather data streams through Kafka components.

### C. APACHE STORM

Storm [91] is a real-time distributed processing engine for large data streams. It has two components that adopt native streaming and micro-batching, respectively. The following Storm modules enhance its reliability in processing data streams with highly abstracted simple processing models:

- Spout: serves as a data stream source in a computation, either from a message broker like Kafka, by generating the data streams itself or from external sources like the Twitter API.
- Bolt: an event-driven computation module that is responsible for processing input streams and generating output streams, akin to window transformation operators in Storm. Each bolt encapsulates certain computations, including functions, stream aggregations, database connections, etc.
- Topology: a directed acyclic graph (DAG) comprised of nodes (spouts or bolts) and edges (how data flows in the process between spouts and bolts). A topology is a multi-stage stream computation that supports parallelism. Once deployed, it operates non-stop.
- Trident: a high-level abstraction for real-time computing using a micro-batching approach on top of Storm. It supports processing computation and allows for high throughput and stateful processing against any database with low latency.

As an example, Cheng et al. [76] has proposed a high-performance computing (HPC) platform that enables streaming services for RTA via Storm.

### D. APACHE FLUME

Apache Flume [92] is a distributed framework with high availability for efficient, large-scale log data processing. Its simple and flexible architecture supports data stream processing, and its extensive data model facilitates online analytic applications. It has mechanisms for tunable reliability, failover, and recovery to ensure robustness and fault tolerance.

Although Flume and Kafka have similarities in terms of real-time event stream processing, they have their unique features. Kafka is better suited for high throughput messaging

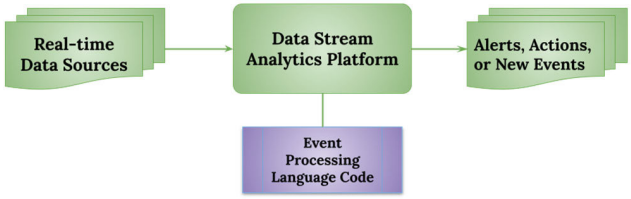


**FIGURE 7.** An abstraction of the data stream analytics process.

platforms on a publish and subscribe basis in which higher availability and scalability are essential, whereas Flume is more suited for data ingestion from various sources into a centralized location with limited event processing capabilities (especially for log data) but is not suitable for CEP. To leverage the capabilities of both Flume and Kafka, some real-time applications have attempted to combine them to get benefits from both, e.g., [93].

### E. COMMERCIAL PLATFORMS

This section briefly describes some commercial platforms for data stream processing. One of them is Amazon Kinesis [94], a cloud-based service for real-time data processing over large, distributed data streams. It has been successfully used for continuous stream data capture and storage on a large scale and at a high rate (terabytes per hour), and it supports a wide variety of data sources like website clicks, social media feeds, financial transactions, IT logs, etc. Kinesis can also be integrated with Storm by providing a Kinesis Storm spout. AWS also offers a stream analytics layer for Kinesis and Kafka based on Flink (more to be discussed in Section VI-D). This kind of incorporation can facilitate a reliable and scalable stream processing service.

Another platform is Confluent [95], a full-scale streaming platform based on Kafka, featuring a streaming SQL engine called KSQL that facilitates real-time data processing.

Last but not least, Microsoft Azure Stream Analytics [88] is a serverless scalable complex event processing engine that supports RTA through an SQL-based language with temporal logic embedded. It supports three input sources, namely Event Hubs (for event data streams in general), IoT Hubs (for data from IoT devices and applications), and Blob Storage (for static data). Another feature is that it enables the integration of PowerBI for real-time dashboarding.

## VI. DATA STREAM ANALYTICS PLATFORMS (LAYER 3)

### A. OVERVIEW

The Layer 2 platforms described in Section V are the foundation for incorporating additional real-time analytics (RTA) functionality, which commonly involves some complex event processing (CEP) capabilities. In this paper, we refer to this software layer as data stream analytics platforms (Layer 3 of the analytics stack, as shown in Figure 4). Generally, when an event stream is fed into an analytics platform, the underlying event processing language code is triggered or executed, which in turn generates a list of alerts, actions, or new events (see Figure 7).

The components of a typical data stream analytics platform may vary and can include the following:

- Event Metadata: contains event data types and event processing rules. Currently, there is no common standard existing for defining and representing event data;
- Event Processing Language (EPL): the language used to express event pattern types or detect event pattern occurrences. Event pattern types must be written in EPL before they can be detected by the Event Processing Engine in the right way;
- EPL Compiler: translates an event pattern description from EPL into machine code that the Event Processing Engine can execute;
- Event Processing Engine: is at the core of a data stream analytics platform. It matches event pattern types specified in the EPL code against the incoming event data and triggers appropriate alerts, actions, and new events. The matched event pattern occurrences are typically made available to these downstream actions once they are detected;
- Event Development and Management Tools: event development tools allow a user to define event data and processing rules, and event management tools are used for managing event data, event generation, event processing infrastructure, etc.;
- Enterprise Integration Components: provides an interface/API for the system to connect to required external services. Examples of common Enterprise Integration Components include event pre-processing, publishing and subscribing, and business process invocation;
- Sources and Targets: specify the sources of incoming event data and the targets on which event-driven actions are performed;
- Event Database: data storage used for storing the event data that are processed by the Event Processing Engine. In more complex RTA systems, a federated database [96] might be used to provide a seamless interface to heterogeneous data sources with a standardized data model or data schemas.

Key operations on events by a typical data stream analytics platform include event expression that allows for defining matching criteria for one event, filtration that subsets events to fit processing needs, transformation that alters events from one form to another by operations like splitting, aggregation, etc., and event pattern detection that identifies occurrences of high-level event pattern types. In particular, event pattern detection can be broken down into three sub-steps, i.e., pre-detection that validates event pattern types and compiles them into executable code by the EPL Compiler component; detection that matches selected event pattern types in incoming data streams and generates event pattern occurrences as output; and post-detection that handles the storage of detected

event pattern occurrences and the execution of follow-on alerts or actions.

As an integral concept of CEP, operations on sliding windows have been introduced to deal with continuously arriving input streams. Sliding windows are used to group events from the original unbounded stream – according to a specific interval – into segments, which can then be manipulated and analyzed with reasonable wait time and memory usage. When handling sliding windows, a formation model that defines how a sliding window is formed, and a processing model that defines when the operation or processing on a sliding window is triggered, need to be defined. The formation model can be one of the following:

- Tuple-based: the sliding window is full when a certain number of tuples/events has entered it;
- Time-based: any event is evicted upon a defined amount of time after it enters into the sliding window;
- Attribute-based: any event that satisfies a condition related to the value of specific attributes of the event is evicted.

Similarly, there are primarily the following four processing models of sliding windows:

- Event-driven: expressions associated with the window are updated incrementally and in real-time as individual events are added/evicted. EventSwarm [97] favors this model;
- Tuple-driven: The content of the window is processed every time a set number of tuples/events arrive. StreamSQL [98] implements this model;
- Time-driven: The content of the window is processed at the end of each time interval;
- Attribute-driven: The content of the window is processed as soon as a condition in terms of attribute values of the constituent events in the window is met.

There are three categories of data stream analytics platforms, namely query-based, rule-based, and programmatic platforms (see Figure 8). Table 3 lists all these categories and example products. In the following subsections, we will provide further details about each category listed.

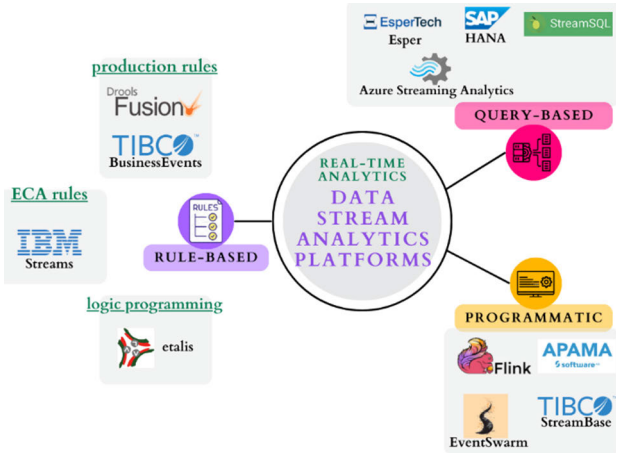


**FIGURE 8.** Taxonomy of data stream analytics platforms for RTA.

**TABLE 3.** Classification of data stream analytics platforms.

| CATEGORY | Products (Open-source ones are marked with *) |
|---|---|
| Query-based | *Esper [99]<br>*Flink SQL [100]<br>SAP HANA Streaming Analytics [101]<br>StreamSQL [98]<br>Azure Streaming Analytics [102] |
| Rule-based: Production rules | *Drools Fusion [103]<br>BusinessEvents [104] |
| Rule-based: ECA rules | IBM Streams [105] |
| Rule-based: Other (Logic programming) | *etalis [106] |
| Programmatic | *Flink [100]<br>*EventSwarm [97]<br>Apama [107]<br>StreamBase [108] |

### B. QUERY-BASED DATA STREAM ANALYTICS

Query-based data stream analytics platforms often have an underlying SQL-based EPL to query data streams. The queries written in such EPLs are applied to incoming event streams and are also known as continuous queries [109], which are stored within the database, as opposed to traditional SQL queries that are non-persistent. The standard steps of executing these queries typically include query definition, query processing, and saving results selectively in the database.

Note that the expressiveness of query-based EPLs varies due to the different processing models adopted, meaning that queries expressed in one EPL may be inexpressible for another language. For example, Azure Stream Analytics [102] implements a time-driven model for sliding windows, whereas StreamSQL adopts a tuple-driven model. In addition, for a particular event pattern type, the results of detected event pattern occurrences by these EPLs may differ, which is hardly controllable by the end user. Therefore, it is critical to stick to consistent semantics when using one particular query-based EPL to ensure expected consistent results.

The advantages of query-based EPLs are that they have good expressiveness in terms of low-level aggregation on top of defined event types, and their basis in SQL makes them easy to learn for most application developers. However, each of these languages has drawbacks for event pattern type abstraction. For instance, CQL lacks the capability of expressing dynamic sliding windows [110], and all have difficulty with the nested data often found in JSON or XML payloads due to their tuple-based event type models.

### C. RULE-BASED DATA STREAM ANALYTICS

In contrast to query-based data stream analytics platforms that have the advantage of expressing low-level abstraction of events, rule-based ones are better suited for expressing conditional event pattern types. Two principal types of rule-based techniques have been used in rule-based data stream analytics platforms, namely production rules and ECA rules.

#### 1) PRODUCTION RULES AND RIPPLE-DOWN RULES

Production rule data stream analytics platforms consist of a set of IF-THEN rules about behavior that responds to the environment, i.e., "if the condition is met, then assert the action". As a common technique within the area of expert systems since the 1980s, production rules have been providing an alternative to artificial intelligence to support knowledge representation and business decision-making in a broad range of domains, including pathology and computer system and network troubleshooting.

While production rules have no intrinsic elements of event stream processing, attempts have been made to introduce additional features on top of conventional production rule systems, including object models and fact base, to enable CEP capabilities, e.g. [111]. Specifically, event types are declared within the production rules. The ingress of events is initialized as instances of the defined event types, and the fact base is used to dynamically add these instances. Functions, including filtering and event pattern detection, are expressed within the condition of the rule and are applied to these event type instances. Tools that adopt this method include Drools [103], Business Events [104], and some dedicated platforms for IoT [112].

Forward-chaining inference and procedural processing have been adopted in most implementations of production rule systems [113], which has led to a consequence that the action has to be modified whenever the condition is updated. In this process, rules may conflict with each other, and the results of rule executions are not typically preserved for future improvements. An error-driven, case-based, and incremental rule acquisition framework called ripple-down rules (RDR) [114] has emerged to address this issue by maintaining a knowledge base and facilitating incremental learning with the domain expert adding rules when a false conclusion occurs without corrupting the existing knowledge base. Specifically, domain experts observe the triggering rule that leads to a wrong conclusion because of a new case (an observed exception) and then add a new rule accordingly to rectify the issue. When the domain expert indicates the conclusion is incorrect, RDR compares the new case with the cornerstone case(s) that have caused the creation of the parent rule resulting in the incorrect conclusion. The interface may ask if each difference is relevant to distinguishing the cases and then build conditions of the new rule accordingly and attach it to the existing rule using the "except" logic. With this approach, essentially, however big the rule base is, no rule is "incorrect" as the newly added rules revise the knowledge without collapsing the original knowledge base.

#### 2) ECA RULES

ECA rules [115] were proposed to facilitate the reaction to events occurring in event-driven architecture and active databases [116], so they are sometimes referred to as active rules. There are three components in an ECA rule, including Event, Condition, and Action, noting the origin of the term ECA:

- Event: the trigger of rule execution. The event can be a composite event comprised of various event types.
- Condition: the criterion that needs to be satisfied via a logic test before the specified action is taken, which is not checked until the specific relevant event occurs.
- Action: the operation to be executed when the condition is met.

To automate rule execution, attempts have been made to integrate some expert systems. Data stream analytics platforms with ECA rules normally support event pattern type description using complex expressions with event algebra operators like And, Or, Sequence, etc. Use cases of ECA rules in data stream analytics platforms include detecting event pattern occurrences, reacting to them, and executing business logic on incoming event streams, e.g. [117], [118]. IBM Streams [105] is an example product of a data stream analytics platform with ECA rules in the in-built EPL.

### D. PROGRAMMATIC DATA STREAM ANALYTICS

Programmatic data stream analytics platforms provide comprehensive functionality for processing complex events. Examples include Apama [107], StreamBase [108], EventSwarm [97], and the Flink DataStream API. These data stream analytics platforms generally follow a high-level architecture.

#### 1) APACHE SPARK

Spark [119] is a newer generation framework than Hadoop for large-scale data analytics that has an execution core engine for distributing programs across clusters and modules for writing programs on top using various languages, including Python, SQL, Scala, Java, and R. It features an interactive Read-Evaluate-Print Loop (REPL) shell environment and utilizes a micro-batching processing approach. As shown in Figure 9, the Spark ecosystem includes a set of tools, including the Spark Core Engine (execution), SQL and Dataframes (interactive SQL), Spark Streaming (the stream processing library), MLlib (the machine learning library), and GraphX (graph processing library). These tools facilitate the implementation of Spark-based applications. For instance, [77] presents an example of utilizing Spark for building an RTA forecasting framework in IoT networks.

Spark Streaming is well suited for RTA, with its scalability, high throughput, and fault-tolerant live stream processing capabilities. Internally, a continuous data stream is

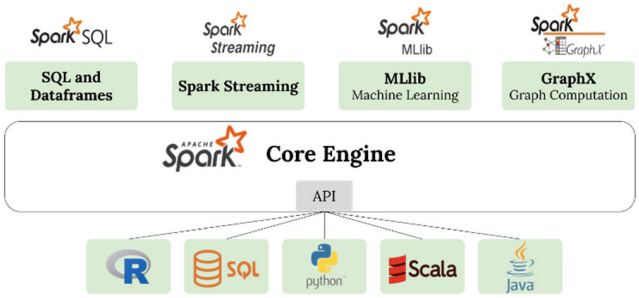


**FIGURE 9.** Apache spark ecosystem.

represented using a high-level abstraction of an immutable, distributed dataset called DStream (discretized stream), which is periodically transformed into an RDD (Resilient Distributed Dataset). RDDs can be processed in a distributed manner by sliding time window computations. The common process of Spark is to first source DStreams directly from data stream sources like Kafka or Flume (discussed in Sections V-B and V-D respectively), HDFS, and Databases or from other DStreams output, divide data into micro-batches, process the data, and finally generate result streams in batches.

In comparison with Hadoop, Spark makes improvements in the following three ways while maintaining the linear scalability and fault tolerance characteristics of MapReduce:

- Support of directed acyclic graph (DAG) operators: with DAG operators supported by the Spark core engine instead of a strict map-then-reduce format, Spark can directly forward results onto the following steps in the workflow while MapReduce must immediately write results to the distributed filesystem.
- Rich set of transformations: this complements the capability of MapReduce, facilitating more natural computation expression by users, including window transformation operators that utilize a micro-batching approach, i.e., dissecting data streams in tiny processable batches within small time intervals;
- In-memory processing across servers: unlike MapReduce, Spark does not rely on disks for reading and writing intermediate data but uses random access memory (RAM). This results in a better performance for large-scale data processing (at least three times faster, according to Apache) than Hadoop.

Despite the differences, various modules in the Hadoop ecosystem can be integrated into the Spark framework. For instance, Spark can read and write all the data formats accepted by MapReduce and can link NoSQL databases like HBase.

### 2) APACHE FLINK

Flink is a recent framework for stream processing as well as batch processing with a core, high-throughput, low-latency, and fault-tolerant streaming engine designed for distributed native streaming [100]. Akin to Apache Spark (discussed in Section VI-D), Flink supports multiple programming languages for the development of applications, including Java, Python, SQL, and Scala. Even though Flink is only a computational engine without internal data storage options, external data storage or platforms can be integrated. Flink supports various input/output sources, including Kafka, RabbitMQ, and Kinesis messaging (discussed in Sections V-B and V-E). Flink can scale horizontally, subject to appropriate partitioning in the underlying data stream platform. It supports both API and SQL queries for pattern matching. In comparison, Confluent KSQL for Kafka has limitations on scale and only supports Kafka for input and output.

Compared with Spark, Flink differentiates itself in the following aspects that make it more suitable for RTA:

- Event-level processing: unlike Spark, which processes data using micro-batching with second-level latency, Flink processes data streams in true real-time (sub-second latency), treating batch processing as bounded data streaming;
- Flexible windowing: Flink provides operations and transformation as part of its DataStream API, including flexible window computations, such as sliding (overlapping time-driven windows of fixed length), tumbling (time-driven windows of specified size without overlapping), and session windows (demarcation of windows using the period of inactivity), whereas the support of windowing in Spark is limited by nature;
- Event pattern detection: its FlinkCEP library offers complex event pattern detection capabilities;
- Support of controlled cyclic dependency graph: instead of DAGs, this approach makes the iterations in the native platform more efficient, resulting in better scalability and performance, which is especially beneficial for the use of ML algorithms.

While Spark has been dominant in big data analytics for years with a huge community, Flink has gradually gained popularity within the RTA world. The Spark community has recognized the ideas brought by Flink and commenced attempts to apply them in Spark. As an example, [120] has proposed a real-time video partitioning tool using Flink deployed on the cloud.

### 3) EVENTSWARM

EventSwarm [121] is an open, Java-based programming framework with a variety of pre-defined event abstractions and event pattern types implemented in Java. Figure 10 demonstrates its conceptual model. EventSwarm supports the processing of all types of event streams, including financial market events like stock trading, Twitter feeds, and HL7 V2 messages in the eHealth domain. Specifically, EventSwarm continuously detects event pattern types and generates an alert or invokes a business action upon the occurrence of a specific event pattern type. Additionally, it has capabilities for filtering computed abstractions such as statistical analysis, supporting causal precedence in

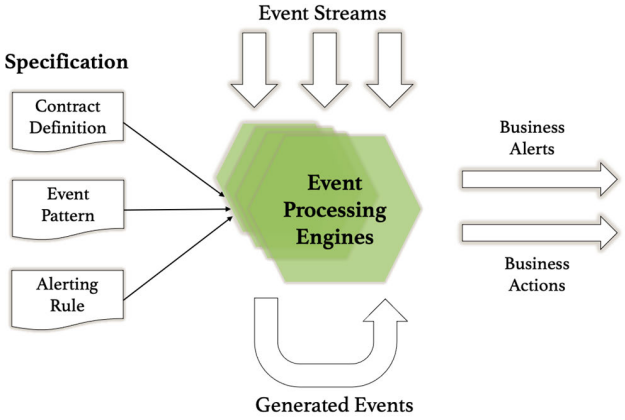


**FIGURE 10.** EventSwarm architecture.

sequence patterns, and handling time and ordering issues using a range of mechanisms, including buffering, causal ordering, flexible relationships, and time skew allowance. Two types of applications implemented using this framework include pattern-oriented ones that are designed for particular pre-defined event patterns and domain user-oriented ones that allow new event patterns to be defined by domain users. The latter typically requires a user-friendly graphical interface.

Note that Drools Fusion [103], a production rule-based data stream analytics platform, has also adopted similar semantics.

### E. SUMMARY

Overall, event processing technology is broadly applied in many application domains and is still being actively researched. Among all the data stream analytics platforms available, many commonalities exist, but the underlying EPL and product functionality varies in accordance with the relevant domain and targeted community and markets. Each of the languages/products listed in Table 3 and Figure 8 has its advantages and disadvantages, reflecting the common trade-off between expressiveness and efficiency. Consequently, the selected EPL can affect the performance, scalability, availability, and maintainability of data stream analytics, and there is currently no one-size-fits-all solution. Each platform has limitations, and it may be costly and troublesome to switch from one platform to another. Software engineers and domain experts should consider the key features discussed in the preceding sections and select the most appropriate platform for their needs.

## VII. ANALYTICS TOOLS (LAYER 4)

The Analytics Tools layer (Layer 4 of the analytics stack, as shown in Figure 4) captures many analytics techniques, augmented with machine learning and AI algorithms, that can provide support for better data insights and prediction capabilities.

### A. AI/ML FOR DATA ANALYTICS

First of all, the following list shows examples of different types of data analytics tools for general purposes, matching various key stages in the usual data analytics lifecycle:

- Data collection tools: survey/data entry tools, e.g., Qualtrics [122] and REDCap [123];
- Data pre-processing tools: Data Wrangler [124], OpenRefine [125];
- Data storage management systems: SQL databases, e.g., MySQL [126], PostgreSQL [127]); NoSQL databases, e.g., Cassandra [128], MongoDB [129];
- Statistical analysis tools: R, SAS [130], Stata [131], and SPSS [132]. In particular, these tools often have visualization capabilities in-built for a decent presentation of analytics results. For predictive analysis, many machine learning tools are available, which will be separately described as follows.

Further, the proliferation and increasing sophistication of artificial intelligence (AI) provide an avenue to perform data analytics tasks with minimal human intervention. There are two major branches of AI for data analytics, namely rule-based AI and data-based AI, also known as machine learning (ML), including artificial neural network-based deep learning models such as the trending large language models (LLMs) [133]. Rule-based AI normally involves a set of human-defined ''if-then'' rules with conditions and actions that facilitate inference and decision-making, e.g. [78]. Data-based AI has gained prevalence in recent years and provides data-driven inference, learning, and creation of models through training and validation and prediction of activities in a particular domain, e.g. [70], [72], [73], [74]. The distinctions between rule-based and data-based AI include the following:

- Rule-based AI features deterministic models, so the results are fixed as long as the rules are set. This is more suitable for cases where the knowledge is well-established. Data-based AI (ML) models largely rely on probability and can evolve continuously through ongoing training. This is particularly important for continuous data streams;
- The inference of rule-based AI is based on the known knowledge, whereas ML training can discover new knowledge or trends from large volumes of data which is uninterpretable directly, and thus ML training creates models to capture this discovered knowledge for future prediction, classification, and other activities;
- ML requires larger amounts of training and testing data for meaningful results as opposed to rule-based AI, which requires much less data.

In many cases, ML is well suited for RTA due to its evolving nature. Much research in ML has focused on accessing, curating, and pre-processing data sets for training purposes. Note that some consider ML as a method of data analytics that automates analytical model building. Specifically, conventional data analytics uses existing data to generate

insights. By contrast, ML focuses on building and training models/algorithms using some existing data and conducting predictive or classification analyses on other data. There are many types of ML algorithms, including supervised learning (classification and regression), unsupervised learning (clustering), self-supervised learning (for processing unlabeled data to obtain useful representations that can help with downstream learning tasks), semi-supervised learning (combining supervised and unsupervised), and reinforcement learning (learning from mistakes for reacting to the environment). Machine learning models range from traditional methods like regression algorithms, decision trees, random forest (RF), and support vector machine (SVM) to more advanced neural network-based deep learning (DL) models such as multi-layer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), and autoencoders (AE). ML and DL models have been extensively employed in almost all fronts of academia and industry. The most recent large language models are mostly based on LSTM with attention mechanism, and reinforcement learning has been utilized.

Figure 11 describes the types of AI and ML algorithms that can be integrated into RTA solutions.

In addition, libraries and tools that facilitate ML tasks include the following:

- Open libraries: TensorFlow, Keras, PyTorch, Scikit-learn, etc. These libraries are mostly Python-based and are broadly used in both academia and industry;
- Open tools and frameworks: Weka, KNIME, Apache Mahout, etc. These tools/frameworks are free to use and provide integrated platforms for ML-based analytics;
- Commercial tools: RapidMiner, Google Colab, Google ML, Amazon ML, etc. These tools are provided by giants like Google and Amazon, either as a standalone platform or as an optional service within their cloud computing offerings.

### B. DATA ANALYTICS TECHNIQUES FOR STREAM PROCESSING

Due to the unique requirements of RTA, special techniques for data stream analytics may be required in addition to the generic analytics techniques for static data, including continuous computation updates, data stream sampling, and out-of-order event handling.

Firstly, mathematical and statistical computation over data streams must take into account that the data is dynamic and changing over time. In most cases, the data arrival can be managed by a sliding window abstraction, enabled in most data stream analytics platforms, as discussed in Section VI, meaning that the arrival of new events may result in the removal of old events based on the type of sliding window. Therefore, the computation must be continuously

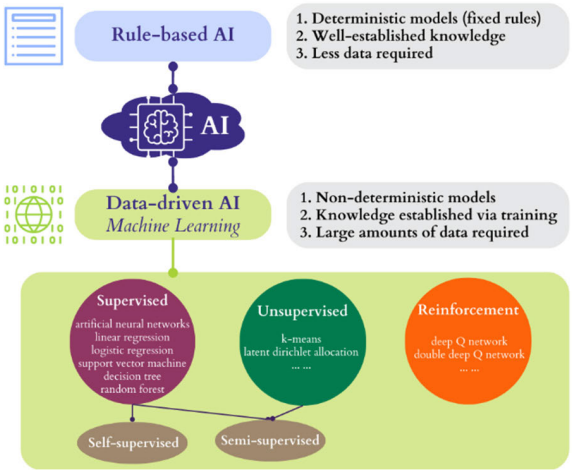


**FIGURE 11.** Taxonomy of AI used in real-time analytics.

updated over time, including simple calculations like min, max, mean, average, and standard deviation and more complex algorithms like multivariable regressions. The trigger of computation updates can be simply time ticks, event arrival, or event removal.

Secondly, when dealing with an infinite stream of data or data streams with huge volumes, it is crucial to balance the cost, performance, and timeliness. Data stream sampling techniques come into play under such circumstances, meaning extracting a representative sample of events from a particular data stream. Types of data stream sampling include sliding window reservoir sampling and biased reservoir sampling [134]. For instance, when adopting prediction algorithms like regression modeling on huge amounts of incoming data, data stream sampling algorithms need to be applied.

Lastly, the ordering of events from multiple sources is always non-deterministic. In particular, events can arrive out of order, and clock skew across event sources can result in the source timestamps being inconsistent with causality. Handling out-of-order events and clock skew is essential for accurate pattern matching on streaming data [121], [135]. A variety of techniques are available, such as buffering for a certain amount of time before processing, recalculating upon receipt of an out-of-order event, adding checkpoints for incremental aggregation, fuzzy timestamp comparison (e.g., events considered concurrent if timestamp difference is $< N$), vector clocks in bounded contexts or designing calculation algorithms that ensure correctness in case of out-of-order events.

Due to the commonality between real-time data streams and time series, many techniques for time series analysis can be used for data stream analytics. However, adjustments to these techniques may be necessary to satisfy specific data stream analytics requirements, as discussed in the three points above.

### C. REAL-TIME ANALYTICS COUPLED WITH MACHINE LEARNING – A NEW TREND

The recent maturity of AI and ML has pushed data analytics to a new level with data-driven inference, learning, and forecasting capabilities. In light of this, applying ML and AI technologies to real-time predictive analytics has gradually become a new trend recently. Processing of continuously changing data streams for ML purposes, however, brings many additional challenges [136]. One of them is continuous learning capabilities, meaning the model's ability to learn continuously from a data stream. This requires flexible architectures to support the integration of pre-processing, training, and prediction tasks.

There are primarily two paths to achieving this, namely, extending existing ML tools by adding event processing capabilities and extending existing streaming platforms by integrating ML capabilities. A case in point for the first path is that the newest version of TensorFlow 2, a mainstream Python machine learning library, has supported interfacing with the Kafka platform to enable RTA [137]. Other examples include a recent extension of the scikit-learn library called stream-learn proposed in [138], the River library proposed in [139], a software design that combines Automated Machine Learning (AutoML) with complex event processing [83], and extending Kafka with artificial neural network for RTA prediction tasks [75]. Examples of the second path include the Kafka-ML framework that integrates ML capabilities into Kafka to facilitate the management and deployment of ML/AI pipelines through data streams [140] and Flink ML API/library [141] to support ML pipelines built on top of Flink. The architecture proposed in [142] has further combined event pattern detection capabilities from EventSwarm with Kafka and TensorFlow. In summary, the key components of such a combined approach include:

- Fetch: a data collection component that delivers data (events) from external sources for processing.
- Stream Processor: a component for simple event processing, e.g., filtering, placing events of specific types on different event channels, possibly to be persisted via specific messaging topics, keeping a persistent store of event occurrences that can subsequently be processed for analytics purposes. An example of implementing this component is Kafka.
- Catch: an event pattern detection component that processes the events from the Stream Processor and matches patterns as per CEP rules. This component applies sophisticated CEP functionality over the events.
- Trainer: an ML training component intended to train models using datasets created incrementally on the Stream Processor. These datasets consist of either the data from the raw events received by the Stream Processor from Fetch or the event pattern instances matched by Catch. The dataset supplied to the machine learning model is determined by the user.
- Predictor: a component that uses the trained model produced by the Trainer, applying it to incoming raw events or detected event patterns to generate predictions. Being separate from Trainer, the Predictor continues its tasks without interruption while the ML model is updated in the background. An example of implementing the Predictor and Trainer components is TensorFlow.

These components will be described in more detail in the case study presented in Section VIII-B.

**TABLE 4.** Applications of combining ML/AI and RTA in recent literature.

| Ref | Year | Application | RTA technique | ML/AI technique |
|---|---|---|---|---|
| [83] | 2023 | Financial market prediction | Self-built CEP | AutoML |
| [72] | 2022 | Video analysis | Self-built stream processing, deployed on Cloud | Rule-based & deep neural network |
| [73] | 2022 | Disease diagnostic | Apache Spark | RF, logistic regression, and SVM |
| [74] | 2021 | Twitter analysis | Apache Spark | Latent Dirichlet allocation, K-means |
| [75] | 2020 | Weather prediction | Apache Kafka | LSTM |
| [143] | 2018 | HDD failure detection | Hadoop + Spark | RF |
| [112] | 2017 | IoT event detection | Query-based data stream analytics + ECA rules | Rule-based |

Table 4 lists representative example applications of combining ML/AI and RTA techniques in the selected literature. Noticeably, the application fields are broad, but the data types involved were mostly time series or event-based data, including financial time series, video streams, Tweets, weather time series, and IoT events. The RTA solutions vary, but all fall into the RTA stack we have defined in Section IV. The ML/AI techniques applied range from traditional ML models to more advanced deep learning models.

## VIII. CASE STUDIES: FINANCE AND HEALTH DOMAINS

To demonstrate the real-life application of the concepts, methods, and technologies discussed in the sections above, this section will consider some scenarios in the real-time financial market and health data analytics and then illustrate how the implementation of real-time analytics (RTA) solutions can address user requirements in these scenarios. These case studies will provide relevant researchers and practitioners with more concrete implementation examples, which further support the answering to RQ2 – "How can practitioners design and implement their own RTA solutions", and RQ3 – "How can practitioners design and implement their own RTA solutions", as identified in Section I-B.

**TABLE 5. Data quality control rules for duplicate dividend announcements.**

| Rule ID | Condition (with event pattern types) | Action |
|---|---|---|
| 1 | Two dividend announcement events have identical timestamps, dividend amounts, and ex-dividend dates. | Remove the former dividend announcement event. |
| 2 | No daily stock price data exists on the ex-dividend date. | Report: Missing price data |
| 3 | A dividend announcement event has no value in the dividend amount and ex-dividend date fields. | Discard this dividend announcement event and report missing value. |
| 4 | A pair of seemingly duplicate dividend announcements (with the same timestamp, etc.) have different "dividend market level IDs". | Retrieve the action in Rule 1, as this case is not a true duplicate. |
| 5 | The payment status of a dividend announcement event is not "Approved". | Exclude this dividend announcement event in any "duplicate dividends" case, as it is outdated, so cannot be considered. |
| 6 | A dividend announcement event is marked for deletion. | Exclude this dividend announcement event in any "duplicate dividends" case, as this event is virtually deleted by the data provider, i.e., it is an invalid entry. |

### A. FINANCE SCENARIO 1 – RULE-BASED DATA QUALITY CONTROL

It is commonly seen that finance experts need to spend a vast amount of time on data collection and pre-processing, in many cases, manually. Complexity arises when dealing with data quality issues, including non-temporal ones like missing data, duplication, and inconsistencies, as well as temporal ones like poor timeliness, out-of-order events, overlapped events, timestamp conflicts, etc., so data quality control (also known as data cleansing) is an indispensable phase of the data pre-processing, as they potentially render the final analysis results error-prone or unreliable.

In this scenario, a real-time data stream is simulated using historical data downloaded from Refinitiv tick data that is part of the Datascope product [144], and we define some complex event processing rules to detect particular event pattern occurrences related to the data quality issue called ''duplicate dividend announcements''. This scenario involves six rules, as shown in Table 5.

The CEP architecture shown in Figure 12 deals with the scenario described above. It combines ripple-down rules (as discussed in Section VI-C) with data stream processing. It is comprised of four key components: a stream processor handling incoming real-time event streams (Layer 2 of the analytics stack), a CEP component responsible for event pattern detection (Layer 3 of the stack), an RDR Builder allowing finance experts to define ripple down rules with domain-specific patterns (Layer 5 of the stack), and an RDR

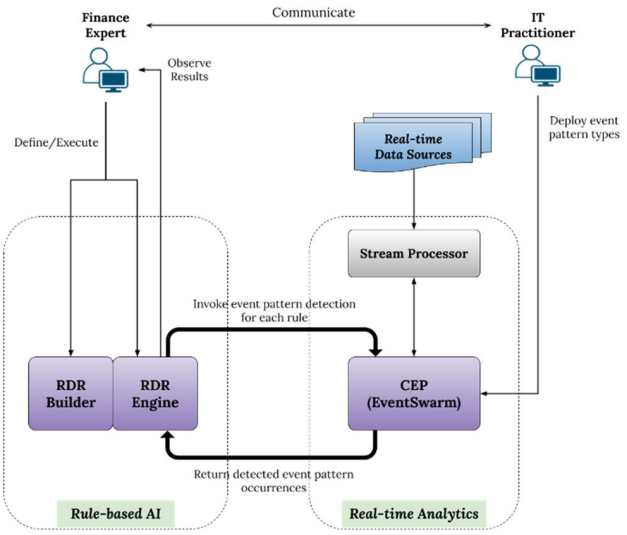


**FIGURE 12. Example CEP application for real-time financial data quality control.**

Engine that executes pre-defined rules, invokes the event pattern detection function of the CEP component, and conducts further analytics of the results (Layer 4 of the stack). In summary, the RDR component manages rules to express the high-level processing logic, and the stream processor and CEP components facilitate real-time data stream processing and analytics, whose role is to detect event pattern occurrences as defined in each rule in Table 5. The combination of these technologies allows for flexibility in event pattern rule definition and management.

A prototype was implemented using EventSwarm, introduced in Section VI-D, as the selected CEP engine and using Kafka as the stream processor. To facilitate fast and convenient development, event pattern definition constructs are provided as Ruby wrappers of the core Java library of EventSwarm. Event pattern types are defined in Ruby as directed acyclic graphs with nodes denoting events and edges denoting relationships and functions such as sliding windows, filters, and abstractions. Matched event pattern occurrences are then sent back to the RDR Engine component to generate subsequent calculations, actions, or alerts.

Experiments involved a finance expert defining the set of six event pattern rules as listed in Table 5. The event pattern types were specified in a natural language (English) and communicated to the IT expert, who then coded the event pattern types in the CEP component. The finance expert then was able to define the six rules using the RDR Builder and selected the rules to be executed using the RDR Engine, which then invokes the CEP component, i.e., EventSwarm, using an HTTP GET request to a configurable URL. Detected event pattern occurrences are returned in JSON format, which are then sent back to the RDR Engine for further processing, giving the option for the finance expert to download and inspect the results at any time.

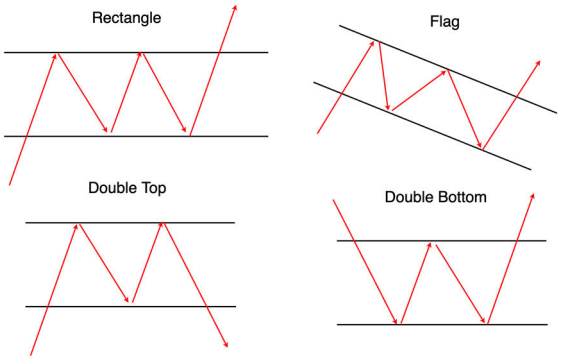


**FIGURE 13.** Common stock price movement patterns.

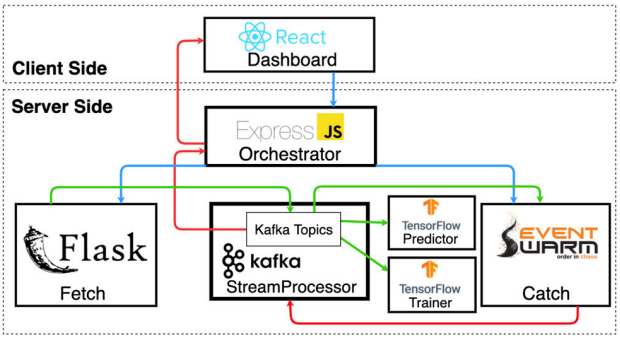


**FIGURE 14.** An instance application for real-time analytics with machine learning capabilities.

The experimental results have shown the advantages of this application, including a simple and easy-to-use API that enables easy integration of EventSwarm into various applications; speedy and efficient complex event processing with an average of over 10,000 events processed per second on tick history daily stock market data, which is in line with the performance of a locally deployed bespoke program dedicated to a fixed event processing computation; rapid definitions of new event patterns (approximately one day for all the six patterns in Table 5 ); and well-structured output in the JSON format that is convenient for further analysis.

A couple of limitations of this RTA application need to be highlighted. First, the natural language (English) was used by finance experts to define event patterns, and communication between the finance experts and the IT practitioners is required. To fully automate the event pattern definition process, natural language processing techniques can be integrated. Second, only one type of CEP was available in the application, which can be extended by wrapping multiple CEP services using a service-oriented architecture.

This scenario highlights the capability of employing rule-based inferencing (rule-based AI) for RTA and how this architecture can be implemented in real life. This approach works best when the rules, part of the domain knowledge, are already established or easy to identify and explain (e.g., the rules for data quality control in this scenario are well-identified and established). In such cases, domain experts and IT practitioners can define the rules based on existing domain knowledge without training, as opposed to machine learning methods (data-driven AI), where the knowledge and patterns are determined based on large amounts of data. More cases like this, including this scenario, can be found in our previous work [145], [146].

### *B. FINANCE SCENARIO 2 – ML-BASED PAIRS TRADING DETECTION*

Pairs trading detection analysis has gained popularity as a profitable investment strategy over the last decades. The goal is to identify correlated securities whose trade prices move in the same direction, i.e., following similar patterns. Common patterns of stock price movement are described in Figure 13 [142], including Rectangle, Flag, Double Top, and Double Bottom. Key elements of these patterns are gradients of movement which can be up, down, or flat. Each gradient line segment denotes the stock prices carried by quote events in the financial market data. Pairs trading detection is beneficial for investors to monitor specific trading opportunities in real-time or select assets during portfolio construction.

In this scenario, machine learning was used to facilitate the detection of pairs trading event patterns. The data was sourced from Yahoo Finance. An event pattern to be detected in this scenario is two or more events in the same direction. We used the method of combining event pattern detection capabilities from EventSwarm with Kafka and TensorFlow, as described in Section VII-C.

The key components used to implement this method are shown in Figure 14 [142]. In addition to the components described in Section VII-C, the Dashboard is the user interface (implemented by React.js) that allows users to specify event patterns, which serves as the domain-specific solution (Layer 5 of the analytics stack). The Orchestrator (implemented by Express.js) is the component that coordinates the activities of other components and manages the internal state of the application. Continuous learning is supported by collecting the event pattern instances detected by the Catch component (implemented using the EventSwarm framework), which serves as Layer 3 of the analytics stack. These event pattern instances are then persisted in the messaging layer of the Stream Processor (implemented using Kafka, corresponding to Layer 2 of the analytics stack) and can be used as training data sets for the Trainer component and testing data sets for the Predictor component. The Trainer and Predictor correspond to Layer 4 of the analytics stack and are both implemented using TensorFlow. In addition, Flask has been used to facilitate communication between components through APIs.

Training data is either sourced from Yahoo Finance by the Fetch component every minute, encoded in a JSON message, and published on the corresponding Kafka topic, or retrieved from event patterns captured by the Catch component.

In this experiment, a prototype has been implemented, including two graphical user interfaces (GUIs), namely the Define Tile and the Display Tile. The Define Tile allows a user to choose the ''Analytics Type'', including the ML algorithm type and the CEP type. In particular, for ML algorithm types, a finance expert could select one of the ML Algorithms, including the LSTM neural network and Dynamic Time Warping (DTW), in this implementation, which can be extended with other algorithms easily. The Display Tile enables the user to specify the stock code of the stocks in question and the data source that the Training component can train on, initiate the training process, and download the real-time dataset in CSV format; this interface shows a counter for the number of detected event patterns, which ticks when a match is found by the Catch component, together with the result of predictions.

From the technical perspective, the Trainer pulls data from a Kafka topic based on the Data Source specified, then performs the training, after which the trained model is transferred to the Predictor for predictions on the new incoming data from the same Data Source.

In our experiment, the finance expert was able to create RTA tasks on pair trading for two different companies, namely Microsoft (stock code ''msft'') and Apple (stock code ''aapl''). In this scenario, DTW was selected as the ML algorithm type, and the GUI showed that the calculated DTW value was approximately 3.84, meaning that these two stocks tended to peak together during the experiment period. It should be noted that the smaller the distance, the more correlated these event patterns are, i.e., these two stocks tend to move in a similar way during this scenario. If the DTW equals 0, it indicates identical event patterns. This experiment has proven the functionality of the method, and finance experts or investors can test with various stock pairs and identify their correlations, i.e., to what extent their prices move in a related manner in real-time.

This scenario reveals the feasibility of integrating data-driven AI (machine learning) into an RTA solution, enabling continuous real-time training and forecasting, and how this architecture can be implemented in practice. More detail of this scenario can be found in our previous work [142].

### C. HEALTH SCENARIOS AND APPLICATIONS

Typically, healthcare information is held in different repositories, and in many situations, real-time access and analytics are required to improve the safety, quality, and efficiency of healthcare. This access is enabled through improved interoperability owing to the emergence of the HL7 FHIR standard [147], both in terms of the common data model and API specification of the standard. So, the analysis of large amounts of health data from different sources, including the detection of clinical patterns of interest, can provide actionable insights to improve operational quality and efficiency for providers, ultimately leading to consumer benefits [148]. This ability is also important in support of the shift from volume-based to value-based healthcare [149].

For example, in our previous work [121], we used EventSwarm CEP for monitoring a stream of health data carried by healthcare messages arriving from pathology labs and generating notifications when certain conditions were detected to assist parties involved in healthcare delivery. The purpose of this work was to detect unusual or inappropriate lab ordering, critical clinical conditions related to patients' health conditions, and data quality issues with laboratory equipment.

For unusual or inappropriate lab ordering detection, we used the sliding time window pattern to immediately identify orders for a test that has recently been ordered by another clinician (i.e., duplicate orders). Detecting such conditions can lead to reduced costs associated with unnecessary requests, which do not add any further clinical value. For example, there is no clinical value in repeating a request for an INR test (which measures the time for the blood to clot) within some threshold time interval (e.g., three days), and thus we use a time-window mechanism to implement this rule, where a clinician changes the threshold value as required. This is essentially Layer 3 of the RTA stack, as we relied on the available EventSwarm constructs, but through the use of statistical analysis, it can be extended to detect where a clinician is ordering higher-than-usual volumes of tests, which can indicate either disease outbreak or over-servicing, which is an example of Layer 4 of the RTA stack.

For critical clinical condition detection, the application provides the basis for a real-time clinical decision support system for situations that require rapid reaction (also Layer 4). This is achieved through applying rule-based AI to combine pathology results arriving in a stream of data messages and patient data stored in a clinical repository to identify patient clinical risks. Examples of detecting such risks include monitoring Haemoglobin A1c results and combining them with patient age, diagnoses, and medications to identify high-risk diabetes patients; and monitoring blood creatinine results and, again, combining them with patient age, diagnoses, and medications to identify patients at risk of kidney failure. The detection of such clinical conditions has been achieved by combining rule-based AI with EventSwarm, akin to the architecture shown in the first finance scenario (Figure 12). The processing capability of EventSwarm (e.g., the order of 1000 messages/sec) is much higher than what is required in a typical laboratory environment at present, but we expect that the increasing number of patients and test results will require such functionality in the near future.

For data quality issue detection, we implemented statistical analysis of pathology messages and rule-based inference into RTA to detect ''outliers''. Syndromic surveillance in real-time was implemented by calculating continuous and incremental statistics for pathology results in messages, then evaluating configurable syndromic surveillance rules. Examples of rules are: when a single data point is great than three standard deviations (SD) from the mean; or when 2 of 3 successive

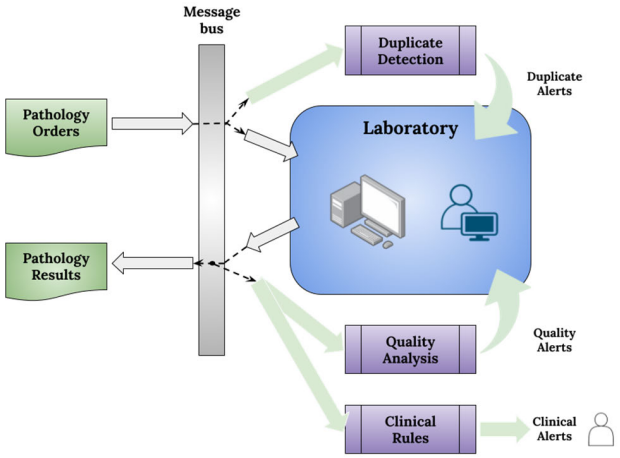


**FIGURE 15.** Real-time architecture for pathology analytics in a typical laboratory setup.

points are greater than 2 SD from the mean. These conditions can identify data quality issues in source systems and allow healthcare providers to improve patient safety through early detection. Figure 15 depicts the overall RTA architecture for this implementation, deployed as an add-in to a typical laboratory setup. From a technical perspective, Pathology orders or results are exchanged via a message bus between a requesting system and a laboratory information system. In this case, the RTA solution treats each of the messages as an event and applies the required rules as the messages flow to and from the laboratory system. EventSwarm (corresponding to Layer 3 of the RTA stack) was used to intercept messages sent between the requestor and the laboratory system and applies rules for ordering (e.g., detecting duplicate orders) or for results (e.g., for the CDS and data quality issues). Here, the system intercepts and creates a copy of each message transmitted to and from a laboratory to EventSwarm processing components that implement rules for data quality and duplicate orders. In case of an event pattern match, the EventSwarm then sends an alert to a clinician in the case of the CDS or duplicate order or to the laboratory operator when a new message matches a data quality alerting rule.

The health scenarios discussed above highlight the use of CEP and rule-based AI to improve healthcare messaging, safety, and efficiency. By monitoring quality metrics and detecting duplicates in pathology orders, alerts are generated to notify relevant parties. The implemented solution demonstrates efficient processing of HL7 v2 messages and suggests potential applications in monitoring medical terms and detecting various issues in healthcare settings. Despite the successful application in real life, improvements can be achieved by implementing data-driven AI (ML) techniques when the rules are unknown to the domain, especially when a large amount of clinical data is available.

In addition, two other applications in the industry are worth noting. The first one is Beamtree [150], an Australian company that provides real-time clinical and health data management with ripple-down rules (RDR, a special rule management technique discussed in Section VI-C) integrated into the RTA implementation. Taking advantage of RDR, the rule base stores clinical knowledge, and expertise, supporting decision-making at scale. The second application is Pathling [151], used for the development of health data analytics applications and workflows used in digital health. It leverages the increasingly-used HL7 FHIR standard and makes use of the RTA features of Apache Spark (Layer 3 of the RTA stack).

## IX. LESSONS LEARNT, CHALLENGES AND FUTURE DIRECTIONS

### A. LESSONS LEARNT AND REMAINING CHALLENGES

There are several lessons learned from this review. First of all, integrating ML and AI into real-time analytics (RTA) solutions has become increasingly prominent in recent literature and practice. As illustrated in Section II and as can be seen in Table 4 in Section VII-C, there are not many applications with integration of ML/AI techniques into RTA, although industrial systems such as digital twins are increasingly adopting ML/AI and RTA in various manufacturing and industrial applications [152]. In addition, despite the proliferation of deep neural networks in recent years, more than half of these existing attempts merely used traditional ML algorithms or rule-based AI rather than advanced deep learning models. However, in many cases, traditional ML and rule-based AI would satisfy the real-life requirements, as described in our case studies in Section VIII.

Several research challenges remain in the area of real-time analytics (RTA). First of all, machine learning and complex event processing are still two distinct areas that have not been integrated to their full potential. Mechanisms for incremental learning in real-time using existing technologies are yet to be investigated. Automating user choices related to ML technique selection and fine-tuning by integrating additional external components for data profiling, transformation, and mining may enable more complex ML techniques, e.g., deep learning and large language models (LLM), to be deployed with CEP-established techniques. This requires applying complex data transformations as required by these techniques. In particular, despite the potential to integrate LLM into RTA systems, this area is still largely unexplored.

Secondly, in many cases, domain users and IT practitioners still have to communicate with each other either verbally or in writing to implement components of RTA solutions. For instance, in the first finance scenario in Section VIII-A, the finance expert had to write in a natural language (English in this case), and then the IT practitioner could then implement the event patterns to detect data quality issues in EventSwarm, which was non-trivial work for both finance experts and IT practitioners. Improvements should be made to simplify and automate such processes.

Thirdly, the data quality of the event streams used in RTA is a key determinant of the quality of analytics results, especially the performance of ML models. The need for ensuring

data quality while executing machine learning algorithms is worth noting, especially when event streams consist of a large number of low-level features. Real-time dimensionality reduction to represent domain knowledge more precisely may be necessary, and this remains challenging.

Another challenge is that there is a gap between the expertise level needed to use RTA techniques and the capabilities of domain experts. Domain experts are not able to complete such analytics tasks by themselves without the intervention of IT experts. Therefore, it is critical to develop user-friendly and domain-specific solutions that have domain-specific models integrated into existing methods.

Last but not least, standards are lacking for supporting interoperation between systems across the various layers or components of the analytics stack discussed in Section IV or even various RTA platforms. Specifically, no standard exists for the representation of event-based abstractions, including event types and event pattern types. It is essential to propose generic models and standards for accurate abstraction representation, domain knowledge representation, and facilitating interoperability between relevant platforms.

### B. RESEARCH AGENDA

Based on the remaining challenges discussed, we highlight the following future research directions, which require exploration from the broader research community.

#### 1) SYNERGY WITH LLM

This is inspired by the benefits of increasingly popular large language models (LLMs). For example, event pattern descriptions in CEP could be considered knowledge graphs that capture real-time semantics like temporal and causal relationships. Incorporating such knowledge graphs into LLM may add more formal semantics to the LLM in a way suitable for the real-time analytics domain, leveraging CEP experience and solutions. On the other hand, LLM can be leveraged to feed the natural language processing results into RTA solutions, such as converting domain experts' statements into the required syntax within the RTA solution. One example is the use of CEP constructs as a target language for conversion from natural language expressions of domain experts. Again, in the first finance scenario in Section VIII-A, this would be a conversion from the natural language expressions of event patterns provided by finance experts into EventSwarm code constructs that describe the event patterns. This would minimize the time-consuming and inefficient communications between finance experts and IT practitioners.

#### 2) EXPLORATION OF REAL-TIME FEATURE EXTRACTION

To ensure data quality of continuously arriving data streams, real-time feature extraction is a topic worthy of extensive research. This could be achieved, for example, by deploying and applying feature extraction techniques in a distributed manner upon low-latency and high-performance infrastructure. Sufficient experiments and appropriate evaluation metrics are required to assess any proposed method in this regard.

**TABLE 6. Summary of core sections in this paper.**

| Section | Title | Content |
|---|---|---|
| III | Concepts and Requirements for real-time analytics | This section sets the scene for the rest of the paper by explaining the key concepts and requirements for real-time analytics. |
| IV | Real-time analytics stack | This section develops a real-time analytics stack to facilitate the discussion on various elements and components for designing and implementing an RTA solution from a software architecture perspective. The following Sections V, VI, and VII, elaborate on critical layers within this analytics stack. |
| V | Data stream processing platforms (Layer 2) | This section discusses data stream processing platforms in detail, which corresponds to Layer 2 of the analytics stack. |
| VI | Data stream analytics platforms (Layer 3) | This section discusses data stream analytics platforms built on top of data stream processing platforms (Layer 2) in detail, which corresponds to Layer 3 of the analytics stack. |
| VII | Analytics tools (Layer 4) | This section discusses analytics tools that capture many analytics techniques augmented with machine learning and AI algorithms. This layer corresponds to Layer 4 of the analytics stack, which is built on top of data stream analytics platforms (Layer 3). |
| VIII | Case studies: finance and health domains | This section illustrates some real-life scenarios and implemented applications in finance and health domains to give some sound grounding to the review and to support the discussed architecture and techniques in previous sections. These case studies also serve as application examples for practical implementations by practitioners. |

#### 3) ENHANCING USER INTERFACE AND EXPERIENCE.

RTA and ML/AI are both highly technical fields that require deep understanding, programming expertise, and competent technical skills. Thus, in many scenarios, domain users may find RTA systems difficult to use. More research on improving the user interface to enhance the user experience is needed. One possible solution could be encapsulating all technical details into fully automated systems. For instance, in our recent work [83], we attempted to combine CEP with AutoML instead of traditional ML or DL, so domain users do not need to be concerned about setting up and fine-tuning ML models, such as data preparation, feature selection, feature extraction, hyperparameter tuning, etc.

#### 4) FORMULATION OF INTEROPERATION STANDARDS

To facilitate smooth communication between components of one RTA system or between multiple various RTA systems, standards of data modeling abstracting domain knowledge, events, and event patterns should be formulated. This could be achieved by discussions in conferences gathering top experts in the RTA field, setting the protocols to agree on worldwide, augmented by promoting these solutions in an open community manner, and possibly leveraging other related repositories and communities such as Hugging Face [153].

It is our intent to continue exploring new academic and industry trends related to AI developments, formal knowledge ontologies, and techniques, such as knowledge graphs, as well as new architecture and software language developments, such as Elixir [152] and applying them in finance, health, and other domains, as we have been doing in the past.

## X. CONCLUSION

This paper has provided an overview of real-time analytics (RTA) solutions, including the relevant infrastructure, processing, and analytics platforms, as well as analytics techniques and software tools. They were described in the context of a logical analytics stack we developed to support the description of key functionality and relationships between components. Several application scenarios in the finance and health domains have been illustrated as case studies to demonstrate the utilization of the techniques and technologies discussed in this paper. Table 6 summarizes the content of the core sections. Key research questions for designing RTA solutions have been answered. Specifically, RQ 1 - ''In accordance with recent research and industrial practice, what are the key concepts, prominent software architectures, techniques, and tools available for RTA solutions'' is addressed by reviewing, illustrating, and summarizing the RTA-related technologies seen in the most recent publications and industrial applications. RQ 2 – ''How can software engineering practitioners design and implement their own RTA solutions?'' is addressed by thoroughly discussing different RTA architectures and their components. Furthermore, the real-life case studies enhance the focus on practicality. RQ3 – ''What are the methods to integrate machine learning or artificial intelligence into RTA solutions'' has been addressed by reviewing, classifying, and discussing different ways of combining RTA and ML/AI, which is again supported by real-life applications demonstrated in the case studies.

This review paper focuses on a unique software perspective of the RTA topic. The novelty of this paper is three-fold. First, our review highlights the methods to integrate ML/AI techniques into RTA solutions. Second, practicality is kept in mind throughout the paper, so we have reviewed both research publications and industrial applications. Last, the demonstrated real-life case studies in finance and health enhance the paper by giving more concrete guidelines for practitioners. Remaining challenges and future research directions, including integration of machine learning and complex event processing to their full potential (e.g., the potential ways of incorporating LLM), data quality, standards for interoperation between components, and domain user-friendly solutions, have been discussed. The content of this paper is significant in that it answers all the identified research questions, and it provides not only a scholarly review of the related state-of-the-art research work but also industrial practices, including real-life implementations and potential future directions, which would benefit both academia and industry in the broader RTA-related community.

## ACKNOWLEDGMENT

None of the authors have a conflict of interest to disclose.

## REFERENCES

[1] A. Woodie. *The Upcoming Year in Big Data: A 2022 Preview*. Datanami. Accessed: 2023. [Online]. Available: https://www.datanami.com/2022/01/10/the-upcoming-year-in-big-data-a-2022-preview/

[2] R. Bean. *Your Data Should Be Faster, Not Just Bigger*. Harvard Business Review. Accessed: 2023. [Online]. Available: https://hbr.org/2015/02/your-data-should-be-faster-not-just-bigger

[3] C. Neff, M. Mendieta, S. Mohan, M. Baharani, S. Rogers, and H. Tabkhi, ''REVAMP$^2$T: Real-time edge video analytics for multicamera privacy-aware pedestrian tracking,'' *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2591–2602, Apr. 2020, doi: 10.1109/JIOT.2019.2954804.

[4] W. Chen, F. Rabhi, W. Liao, and I. Al-Qudah, ''Leveraging state-of-the-art topic modeling for news impact analysis on financial markets: A comparative study,'' *Electronics*, vol. 12, no. 12, p. 2605, Jun. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/12/2605

[5] K. Muthukumaran and K. Hariharanath, ''Deep learning enabled financial crisis prediction model for small-medium sized industries,'' *Intell. Automat. Soft Comput.*, vol. 35, no. 1, pp. 522–536, 2023.

[6] T. Dhar, N. Dey, S. Borra, and R. S. Sherratt, ''Challenges of deep learning in medical image analysis—Improving explainability and trust,'' *IEEE Trans. Technol. Soc.*, vol. 4, no. 1, pp. 68–75, Mar. 2023, doi: 10.1109/TTS.2023.3234203.

[7] Z. Milosevic, W. Chen, A. Berry, and F. A. Rabhi, ''Real-time analytics,'' in *Big Data*, R. Buyya, R. N. Calheiros, and A. V. Dastjerdi, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2016, ch. 2, pp. 39–61.

[8] S. J. Morshed, J. Rana, and M. Milrad, ''Real-time data analytics: An algorithmic perspective,'' in *Data Mining and Big Data*, Y. Tan and Y. Shi, Eds. Cham, Switzerland: Springer, 2016, pp. 311–320.

[9] S. Verma, Y. Kawamoto, Z. Md. Fadlullah, H. Nishiyama, and N. Kato, ''A survey on network methodologies for real-time analytics of massive IoT data and open research issues,'' *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017, doi: 10.1109/COMST.2017.2694469.

[10] T. Saheb, ''Big data analytics in the context of Internet of Things and the emergence of real-time systems: A systematic literature review,'' *Int. J. High Perform. Syst. Archit.*, vol. 8, nos. 1–2, pp. 34–50, Aug. 2018, doi: 10.1504/IJHPSA.2018.094143.

[11] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, ''Real-time big data processing for anomaly detection: A survey,'' *Int. J. Inf. Manage.*, vol. 45, pp. 289–307, Apr. 2019, doi: 10.1016/j.ijinfomgt.2018.08.006.

[12] T. Dubuc, F. Stahl, and E. B. Roesch, ''Mapping the big data landscape: Technologies, platforms and paradigms for real-time analytics of data streams,'' *IEEE Access*, vol. 9, pp. 15351–15374, 2021, doi: 10.1109/ACCESS.2020.3046132.

[13] R. Ghaffari, D. S. Yang, J. Kim, A. Mansour, J. A. Wright, J. B. Model, D. E. Wright, J. A. Rogers, and T. R. Ray, ''State of sweat: Emerging wearable systems for real-time, noninvasive sweat sensing and analytics,'' *ACS Sensors*, vol. 6, no. 8, pp. 2787–2801, Aug. 2021, doi: 10.1021/acssensors.1c01133.

[14] M. Boumrah, S. Garbaya, and A. Radgui, ''Real-time visual analytics for in-home medical rehabilitation of stroke patient—Systematic review,'' *Med. Biol. Eng. Comput.*, vol. 60, no. 4, pp. 889–906, Apr. 2022, doi: 10.1007/s11517-021-02493-w.

[15] S. D. Kuznetsov, P. E. Velikhov, and Q. Fu, "Real-time analytics: Benefits, limitations, and tradeoffs," *Program. Comput. Softw.*, vol. 49, no. 1, pp. 1–25, Feb. 2023, doi: 10.1134/S036176882301005X.

[16] Z. Zhang, S. Yang, C. Zhao, X. Ren, H. Yu, Q. Han, and S. Guo, "RTCoInfer: Real-time collaborative CNN inference for stream analytics on ubiquitous images," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1212–1226, Apr. 2023, doi: 10.1109/JSAC.2023.3242730.

[17] Z. Wang, X. He, Z. Zhang, Y. Zhang, Z. Cao, W. Cheng, W. Wang, and Y. Cui, "Edge-assisted real-time video analytics with spatial–temporal redundancy suppression," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 6324–6335, Apr. 2023, doi: 10.1109/JIOT.2022.3224750.

[18] H. Wang, Q. Li, H. Sun, Z. Chen, Y. Hao, J. Peng, Z. Yuan, J. Fu, and Y. Jiang, "VaBUS: Edge-cloud real-time video analytics via background understanding and subtraction," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 90–106, Jan. 2023, doi: 10.1109/JSAC.2022.3221995.

[19] B. Qian, Z. Wen, J. Tang, Y. Yuan, A. Y. Zomaya, and R. Ranjan, "OsmoticGate: Adaptive edge-based real-time video analytics for the Internet of Things," *IEEE Trans. Comput.*, vol. 72, no. 4, pp. 1178–1193, Apr. 2023, doi: 10.1109/TC.2022.3193630.

[20] M. N. Hasnine, H. T. Nguyen, T. T. T. Tran, H. T. T. Bui, G. Akçapınar, and H. Ueda, "A real-time learning analytics dashboard for automatic detection of online learners' affective states," *Sensors*, vol. 23, no. 9, p. 4243, Apr. 2023, doi: 10.3390/s23094243.

[21] J. Chenyu, Y. Jun, X. Ke, H. Zhanyu, and Y. Ming, "Coupling of adjoint-based Markov/CCMT predictive analytics with data assimilation for real-time risk scenario forecasting of industrial digital process control systems," *Process Saf. Environ. Protection*, vol. 171, pp. 951–974, Mar. 2023, doi: 10.1016/j.psep.2023.01.077.

[22] M. Abdel-Aty, O. Zheng, Y. Wu, A. Abdelraouf, H. Rim, and P. Li, "Real-time big data analytics and proactive traffic safety management visualization system," *J. Transp. Eng. A, Syst.*, vol. 149, no. 8, Aug. 2023, Art. no. 04023064, doi: 10.1061/JTEPBS.TEENG-7530.

[23] L. Zheng, J. Niu, and L. Zhong, "Effects of a learning analytics-based real-time feedback approach on knowledge elaboration, knowledge convergence, interactive relationships and group performance in CSCL," *Brit. J. Educ. Technol.*, vol. 53, no. 1, pp. 130–149, Jan. 2022, doi: 10.1111/bjet.13156.

[24] Y. Yang, S. Lee, and J. Lee, "Video analytics-based real-time intelligent crossing detection system (RICDS): Killer app for edge computing," *Future Gener. Comput. Syst.*, vol. 133, pp. 84–94, Aug. 2022, doi: 10.1016/j.future.2022.03.013.

[25] B. T. Thodi, B. R. Chilukuri, and L. Vanajakshi, "An analytical approach to real-time bus signal priority system for isolated intersections," *J. Intell. Transp. Syst.*, vol. 26, no. 2, pp. 145–167, Mar. 2022, doi: 10.1080/15472450.2020.1797504.

[26] T. H. Tan, C. Y. Ooi, and M. N. Marsono, "RtFog: A real-time FPGA-based fog node with remote dynamically reconfigurable application plane for fog analytics redeployment," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 341–351, Mar. 2022, doi: 10.1109/TGCN.2021.3122545.

[27] E. Shin, S. Yoo, Y. Ju, and D. Shin, "Knowledge graph embedding and reasoning for real-time analytics support of chemical diagnosis from exposure symptoms," *Process Saf. Environ. Protection*, vol. 157, pp. 92–105, Jan. 2022, doi: 10.1016/j.psep.2021.11.002.

[28] P. Sagmeister, F. F. Ort, C. E. Jusner, D. Hebrault, T. Tampone, F. G. Buono, J. D. Williams, and C. O. Kappe, "Autonomous multi-step and multi-objective optimization facilitated by real-time process analytics," *Adv. Sci.*, vol. 9, no. 10, Apr. 2022, Art. no. 2105547, doi: 10.1002/advs.202105547.

[29] H. Maosa, K. Ouazzane, and V. Sowinski-Mydlarz, "Real-time cyber analytics data collection framework," *Int. J. Inf. Secur. Privacy*, vol. 16, no. 1, pp. 1–10, Oct. 2022, doi: 10.4018/IJISP.311465.

[30] S. Irmak, D. Brar, M. S. Kukal, L. Odhiambo, and K. Djaman, "Automated real-time irrigation analytics inform diversity in regional irrigator behavior and water withdrawal and use characteristics," *Agricult. Water Manage.*, vol. 272, Oct. 2022, Art. no. 107837, doi: 10.1016/j.agwat.2022.107837.

[31] F. Zhao, S. Li, B. B. Zhou, H. Jin, and L. T. Yang, "HCache: A hash-based hybrid caching model for real-time streaming data analytics," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1384–1396, Sep. 2021, doi: 10.1109/TSC.2018.2874966.

[32] G. Ulm, S. Smith, A. Nilsson, E. Gustavsson, and M. Jirstrand, "OODIDA: On-board/off-board distributed real-time data analytics for connected vehicles," *Data Sci. Eng.*, vol. 6, no. 1, pp. 102–117, Mar. 2021, doi: 10.1007/s41019-021-00152-6.

[33] S. J. Raychaudhuri, S. Manjunath, C. P. Srinivasan, N. Swathi, S. Sushma, K. N. N. Bhushan, and C. N. Babu, "Prescriptive analytics for impulsive behaviour prevention using real-time biometrics," *Prog. Artif. Intell.*, vol. 10, no. 2, pp. 99–112, Jun. 2021, doi: 10.1007/s13748-020-00229-9.

[34] K. Padmaja and R. Seshadri, "Analytics on real time security attacks in healthcare, retail and banking applications in the cloud," *Evol. Intell.*, vol. 14, no. 2, pp. 595–605, Jun. 2021, doi: 10.1007/s12065-019-00337-z.

[35] A. Naseer, H. Naseer, A. Ahmad, S. B. Maynard, and A. M. Siddiqui, "Real-time analytics, incident response process agility and enterprise cybersecurity performance: A contingent resource-based analysis," *Int. J. Inf. Manage.*, vol. 59, Aug. 2021, Art. no. 102334, doi: 10.1016/j.ijinfomgt.2021.102334.

[36] J. T. Klamo, T. M. Turner, C. Y. Cool, K. I. Yeager, and Y. W. Kwon, "On the accuracy of an analytical solution to model wave-induced loads on an underwater vehicle in real-time," *J. Offshore Mech. Arctic Eng.*, vol. 143, no. 3, Jun. 2021, doi: 10.1115/1.4049119.

[37] M. S. A. Khan, K. M. Kadir, M. K. Alam, S. Mahmud, S. R. M. F. U. Hossain, M. P. Sikder, F. Jefreen, and A. Kamal, "An analytical approach to real-time cloud services on IoT-based applications for smart city planning," *Int. J. Grid Utility Comput.*, vol. 12, nos. 5–6, pp. 507–523, 2021, doi: 10.1504/IJGUC.2021.120120.

[38] C. Kattadige, K. N. Choi, A. Wijesinghe, A. Nama, K. Thilakarathna, S. Seneviratne, and G. Jourjon, "SETA++: Real-time scalable encrypted traffic analytics in multi-Gbps networks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3244–3259, Sep. 2021, doi: 10.1109/TNSM.2021.3085097.

[39] R. Hasan, W. Shehzad, E. Ahmed, H. A. Khattak, A. S. AlGhamdi, and S. S. Alshamrani, "Location-dependent query processing: Semantic cache for real-time smart city analytics," *Appl. Bionics Biomech.*, vol. 2021, Dec. 2021, Art. no. 9958647, doi: 10.1155/2021/9958647.

[40] Y.-T. Chen, E. W. Sun, M.-F. Chang, and Y.-B. Lin, "Pragmatic real-time logistics management with traffic IoT infrastructure: Big data predictive analytics of freight travel time for Logistics 4.0," *Int. J. Prod. Econ.*, vol. 238, Aug. 2021, Art. no. 108157, doi: 10.1016/j.ijpe.2021.108157.

[41] F. Yao and Y. Wang, "Towards resilient and smart cities: A real-time urban analytical and geo-visual system for social media streaming data," *Sustain. Cities Soc.*, vol. 63, Dec. 2020, Art. no. 102448, doi: 10.1016/j.scs.2020.102448.

[42] L. Wu, G. Cai, S. Zhao, and R. Ramamoorthi, "Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–14, Aug. 2020, doi: 10.1145/3386569.3392373.

[43] E. Skordilis and R. Moghaddass, "A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics," *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106600, doi: 10.1016/j.cie.2020.106600.

[44] P. P. Ray, D. Dash, and D. De, "Real-time event-driven sensor data analytics at the edge-Internet of Things for smart personal healthcare," *J. Supercomput.*, vol. 76, no. 9, pp. 6648–6668, Sep. 2020, doi: 10.1007/s11227-019-03089-w.

[45] R. Peng, T. Zheng, X. Lu, X. Xu, and F. Xu, "Simulation of a synchronous planar magnetically levitated motion system based on a real-time analytical force model," *Energies*, vol. 13, no. 23, p. 6367, Dec. 2020, doi: 10.3390/en13236367.

[46] E. Pagliano, M. Onor, Z. Mester, and A. D'Ulivo, "Application of direct analysis in real time to study chemical vapor generation mechanisms: Reduction of dimethylarsinic(V) acid with aqueous $NaBH_4$ under non-analytical conditions," *Anal. Bioanal. Chem.*, vol. 412, no. 27, pp. 7603–7613, Nov. 2020, doi: 10.1007/s00216-020-02896-y.

[47] G. Manogaran, S. Baskar, P. M. Shakeel, N. Chilamkurti, and R. Kumar, "Analytics in real time surveillance video using two-bit transform accelerative regressive frame check," *Multimedia Tools Appl.*, vol. 79, nos. 23–24, pp. 16155–16172, Jun. 2020, doi: 10.1007/s11042-019-7526-3.

[48] S. Manjunatha and B. Annappa, "Real-time big data analytics framework with data blending approach for multiple data sources in smart city applications," *Scalable Comput.*, vol. 21, no. 4, pp. 611–623, 2020, doi: 10.12694/scpe.v21i4.1759.

[49] P. Kumar and H. H. Huang, "GRAPHONE: A data store for real-time analytics on evolving graphs," *ACM Trans. Storage*, vol. 15, no. 4, pp. 1–40, Nov. 2019, doi: 10.1145/3364180.

[50] Z. Khan and J. J. Lehtomäki, "FPGA-assisted real-time RF wireless data analytics system: Design, implementation, and statistical analyses," *IEEE Access*, vol. 8, pp. 4383–4396, 2020, doi: 10.1109/ACCESS.2019.2962200.

[51] H. Hu and L. Tang, "Edge intelligence for real-time data analytics in an IoT-based smart metering system," *IEEE Netw.*, vol. 34, no. 5, pp. 68–74, Sep. 2020, doi: 10.1109/MNET.011.2000039.

[52] A. Haghighati and K. Sedig, "VARTTA: A visual analytics system for making sense of real-time Twitter data," *Data*, vol. 5, no. 1, p. 20, Feb. 2020, doi: 10.3390/data5010020.

[53] V. Gupta and R. Hewett, "Real-time tweet analytics using hybrid hashtags on Twitter big data streams," *Information*, vol. 11, no. 7, p. 341, Jun. 2020, doi: 10.3390/INFO11070341.

[54] K. Gkiotsalitis and E. C. van Berkum, "An analytic solution for real-time bus holding subject to vehicle capacity limits," *Transp. Res. C, Emerg. Technol.*, vol. 121, Dec. 2020, Art. no. 102815, doi: 10.1016/j.trc.2020.102815.

[55] G. A. de Oliveira, R. D. O. Albuquerque, C. A. B. de Andrade, R. T. de Sousa, A. L. S. Orozco, and L. J. G. Villalba, "Anonymous real-time analytics monitoring solution for decision making supported by sentiment analysis," *Sensors*, vol. 20, no. 16, pp. 1–29, 2020, doi: 10.3390/s20164557.

[56] R. Bolla, R. Bruschi, F. Davoli, and J. F. Pajo, "A model-based approach towards real-time analytics in NFV infrastructures," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 529–541, Jun. 2020, doi: 10.1109/TGCN.2019.2961192.

[57] B. A. Hammou, A. A. Lahcen, and S. Mouline, "Towards a real-time processing framework based on improved distributed recurrent neural network variants with fastText for social big data analytics," *Inf. Process. Manage.*, vol. 57, no. 1, Jan. 2020, Art. no. 102122, doi: 10.1016/j.ipm.2019.102122.

[58] Y. Zhang, Y. Xu, Z. Y. Dong, and R. Zhang, "A hierarchical self-adaptive data-analytics method for real-time power system short-term voltage stability assessment," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 74–84, Jan. 2019, doi: 10.1109/TII.2018.2829818.

[59] M. Valero, F. Li, S. Wang, F. Lin, and W. Song, "Real-time cooperative analytics for ambient noise tomography in sensor networks," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 5, no. 2, pp. 375–389, Jun. 2019, doi: 10.1109/TSIPN.2018.2876751.

[60] C. Soares and K. Gray, "Real-time predictive capabilities of analytical and machine learning rate of penetration (ROP) models," *J. Petroleum Sci. Eng.*, vol. 172, pp. 934–959, Jan. 2019, doi: 10.1016/j.petrol.2018.08.083.

[61] H. Shao and W. Li, "A comprehensive optimization strategy for real-time spatial feature sharing and visual analytics in cyberinfrastructure," *Int. J. Digit. Earth*, vol. 12, no. 3, pp. 250–269, Mar. 2019, doi: 10.1080/17538947.2017.1421719.

[62] G. Ortiz, J. A. Caravaca, A. García-de-Prado, F. de la O. Chávez, and J. Boubeta-Puig, "Real-time context-aware microservice architecture for predictive analytics and smart decision-making," *IEEE Access*, vol. 7, pp. 183177–183194, 2019, doi: 10.1109/ACCESS.2019.2960516.

[63] M. P. V. D. Oliveira and R. Handfield, "Analytical foundations for development of real-time supply chain capabilities," *Int. J. Prod. Res.*, vol. 57, no. 5, pp. 1571–1589, Mar. 2019, doi: 10.1080/00207543.2018.1493240.

[64] M. Kuentz, "Drug supersaturation during formulation digestion, including real-time analytical approaches," *Adv. Drug Del. Rev.*, vol. 142, pp. 50–61, Mar. 2019, doi: 10.1016/j.addr.2018.11.003.

[65] J. Feder, "Real-time analytics improves process safety in a drilling-contractor operations center," *J. Petroleum Technol.*, vol. 71, no. 5, pp. 58–61, May 2019, doi: 10.2118/0519-0058-JPT.

[66] F. Fan, K. Bell, and D. Infield, "Transient-state real-time thermal rating forecasting for overhead lines by an enhanced analytical method," *Electr. Power Syst. Res.*, vol. 167, pp. 213–221, Feb. 2019, doi: 10.1016/j.epsr.2018.11.003.

[67] A. Ellis, W. Hunt, and J. Hart, "Real-time analytic antialiased text for 3-D environments," *Comput. Graph. Forum*, vol. 38, no. 8, pp. 23–32, Nov. 2019, doi: 10.1111/cgf.13757.

[68] V. Chauhan, R. Gaur, A. Tiwari, and A. Shukla, "Real-time BigData and predictive analytical architecture for healthcare application," *Sādhanā-Acad. Proc. Eng. Sci.*, vol. 44, no. 12, p. 237, Dec. 2019, doi: 10.1007/s12046-019-1220-z.

[69] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, May 2019, doi: 10.3390/s19092048.

[70] A. C. Y. Yi, T. K. Ying, S. J. Yee, W. M. Chin, and T. T. Tin, "InPath forum: A real-time learning analytics and performance ranking forum system," *IEEE Access*, vol. 10, pp. 128536–128542, 2022, doi: 10.1109/ACCESS.2022.3227430.

[71] M. Babar and F. Arif, "Real-time data processing scheme using big data analytics in Internet of Things based smart transportation environment," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 10, pp. 4167–4177, Oct. 2019, doi: 10.1007/s12652-018-0820-5.

[72] M. Ali, A. Anjum, O. Rana, A. R. Zamani, D. Balouek-Thomert, and M. Parashar, "RES: Real-time video stream analytics using edge enhanced clouds," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 792–804, Jun. 2022, doi: 10.1109/TCC.2020.2991748.

[73] E. Yıldırım, A. Çalhan, and M. Cicioğlu, "Performance analysis of disease diagnostic system using IoMT and real-time data analytics," *Concurrency Comput. Pract. Exper.*, vol. 34, no. 13, p. e6916, Jun. 2022, doi: 10.1002/cpe.6916.

[74] A. P. Rodrigues, R. Fernandes, A. Bhandary, A. C. Shenoy, A. Shetty, and M. Anisha, "Real-time Twitter trend analysis using big data analytics and machine learning techniques," *Wireless Commun. Mobile Comput.*, vol. 2021, Oct. 2021, Art. no. 3920325, doi: 10.1155/2021/3920325.

[75] K. Lavanya, S. Venkatanarayanan, and A. A. Bhoraskar, "Real-time weather analytics: An end-to-end big data analytics service over Apach Spark with Kafka and long short-term memory networks," *Int. J. Web Services Res.*, vol. 17, no. 4, pp. 15–31, Oct. 2020, doi: 10.4018/IJWSR.2020100102.

[76] Y. Cheng, Z. Hao, and R. Cai, "Auto-scaling for real-time stream analytics on HPC cloud," *Service Oriented Comput. Appl.*, vol. 13, no. 2, pp. 169–183, Jun. 2019, doi: 10.1007/s11761-019-00262-0.

[77] A. M. Fernández-Gómez, D. Gutiérrez-Avilés, A. Troncoso, and F. Martínez-Álvarez, "A new Apache Spark-based framework for big data streaming forecasting in IoT networks," *J. Supercomput.*, vol. 79, no. 10, pp. 11078–11100, Jul. 2023, doi: 10.1007/s11227-023-05100-x.

[78] L. Lim, M. Bannert, J. van der Graaf, S. Singh, Y. Fan, S. Surendrannair, M. Rakovic, I. Molenaar, J. Moore, and D. Gašević, "Effects of real-time analytics-based personalized scaffolds on students' self-regulated learning," *Comput. Hum. Behav.*, vol. 139, Feb. 2023, Art. no. 107547, doi: 10.1016/j.chb.2022.107547.

[79] U. I. Minhas, M. Russell, S. Kaloutsakis, P. Barber, R. Woods, G. Georgakoudis, C. Gillan, D. S. Nikolopoulos, and A. Bilas, "NanoStreams: A microserver architecture for real-time analytics on fast data streams," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 3, pp. 396–409, Jul. 2018, doi: 10.1109/TMSCS.2017.2764087.

[80] O. Etzion and P. Niblett, *Event Processing in Action*. Shelter Island, NY, USA: Manning, 2011.

[81] J. C. Nwokeji and R. Matovu, "A systematic literature review on big data extraction, transformation and loading (ETL)," in *Intelligent Computing*. Cham, Switzerland: Springer, 2021, pp. 308–324.

[82] D. Luckham, *Event Processing for Business: Organizing the Real-Time Enterprise*. Hoboken, NJ, USA: Wiley, 2012.

[83] W. Chen, A. El Majzoub, I. Al-Qudah, and F. A. Rabhi, "A CEP-driven framework for real-time news impact prediction on financial markets," *Service Oriented Comput. Appl.*, vol. 17, no. 2, pp. 129–144, Jun. 2023, doi: 10.1007/s11761-023-00358-8.

[84] B. Ellis, *Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data*. Hoboken, NJ, USA: Wiley, 2014.

[85] Kubernetes. *Kubernetes*. Accessed: 2023. [Online]. Available: https://kubernetes.io/

[86] *Elasticsearch*. Accessed: 2023. [Online]. Available: https://en.wikipedia.org/wiki/Elasticsearch

[87] Hitachi. *Hitachi Vantara*. Accessed: 2023. [Online]. Available: https://www.hitachivantara.com/en-us/products/data-management-analytics.html

[88] Microsoft. *Analytics End-to-End with Azure Synapse*. Accessed: 2023. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/example-scenario/dataplate2e/data-platform-end-to-end?tabs=portal

[89] *Apache Hadoop*. Accessed: 2023. [Online]. Available: http://hadoop.apache.org/

[90] *Apache Kafka*. Accessed: 2023. [Online]. Available: https://kafka.apache.org

[91] *Apache Storm*. Accessed: 2023. [Online]. Available: http://storm.apache.org
[92] *Apache Flume*. Accessed: 2023. [Online]. Available: https://flume.apache.org
[93] A. L. Karn, S. Pandya, A. Mehbodniya, F. Arslan, D. K. Sharma, K. Phasinam, M. N. Aftab, R. Rajan, R. K. Bommisetti, and S. Sengan, "An integrated approach for sustainable development of wastewater treatment and management system using IoT in smart cities," *Soft Comput.*, vol. 27, pp. 5159–5175, Sep. 2021, doi: 10.1007/s00500-021-06244-9.
[94] *Amazon Kinesis*. Accessed: 2023. [Online]. Available: https://aws.amazon.com/kinesis/
[95] Confluent. Accessed: 2023. [Online]. Available: https://www.confluent.io/
[96] L. Azevedo, E. Soares, R. Souza, and M. Moreno, "Modern federated database systems: An overview," in *Proc. 22nd Int. Conf. Enterprise Inf. Syst. (ICEIS)*, 2020, pp. 276–283.
[97] Deontik. *EventSwarm*. Accessed: 2023. [Online]. Available: http://deontik.com/Products/EventSwarm.html
[98] StreamSQL. *StreamSQL*. Accessed: 2023. [Online]. Available: https://streamsql.io/
[99] EsperTech. *Esper*. Accessed: 2023. [Online]. Available: https://www.espertech.com/esper/
[100] *Apache Flink*. Accessed: 2023. [Online]. Available: https://flink.apache.org/
[101] SAP. *HANA Streaming Analytics*. Accessed: 2023. [Online]. Available: https://blog.sap-press.com/what-is-sap-hana-streaming-analytics
[102] Microsoft. *Azure Streaming Analytics*. Accessed: 2023. [Online]. Available: https://azure.microsoft.com/en-us/services/stream-analytics/
[103] Drools. *Drools Fusion*. Accessed: 2023. [Online]. Available: https://drools.org/
[104] TIBCO. *TIBCO BusinessEvents*. Accessed: 2023. [Online]. Available: https://www.tibco.com/products/tibco-businessevents
[105] *IBM Streams*. Accessed: 2023. [Online]. Available: https://www.ibm.com/cloud/streaming-analytics
[106] Google. *ETALIS*. Accessed: 2023. [Online]. Available: https://code.google.com/archive/p/etalis/
[107] SoftwareAG. *Apama*. Accessed: 2023. [Online]. Available: https://www.softwareag.com/en_corporate/platform/iot/apama.html
[108] TIBCO. *TIBCO StreamBase*. Accessed: 2023.[Online]. Available: https://www.tibco.com/resources/datasheet/tibco-streambase
[109] A. M. Rahmani, Z. Babaei, and A. Souri, "Event-driven IoT architecture for data analysis of reliable healthcare application using complex event processing," *Cluster Comput.*, vol. 24, no. 2, pp. 1347–1360, Jun. 2021, doi: 10.1007/s10586-020-03189-w.
[110] K. Robert, *Evaluation of the Stream Query Language CQL*. Sweden, U.K.: The European Library, 2010.
[111] A. Vankipuram, M. Vankipuram, V. Ghaemmaghami, and V. L. Patel, "A mobile application to support collection and analytics of real-time critical care data," *Comput. Methods Programs Biomed.*, vol. 151, pp. 45–55, Nov. 2017, doi: 10.1016/j.cmpb.2017.08.014.
[112] M. I. Ali, N. Ono, M. Kaysar, Z. U. Shamszaman, T.-L. Pham, F. Gao, K. Griffin, and A. Mileo, "Real-time data analytics and event detection for IoT-enabled communication systems," *J. Web Semantics*, vol. 42, pp. 19–37, Jan. 2017, doi: 10.1016/j.websem.2016.07.001.
[113] A. Paschke and A. Kozlenkov, "Rule-based event processing and reaction rules," in *Rule Interchange and Applications* (Lecture Notes in Computer Science), vol. 5858, G. Governatori, J. Hall, and A. Paschke, Eds. Berlin, Germany: Springer, 2009, ch. 8, pp. 53–66.
[114] P. Compton and B. H. Kang, *Ripple-Down Rules: The Alternative to Machine Learning*. London, U.K.: Chapman & Hall, 2021.
[115] K. Dittrich, S. Gatziu, and A. Geppert, "The active database management system manifesto: A rulebase of ADBMS features," in *Rules in Database Systems* (Lecture Notes in Computer Science), vol. 985, T. Sellis, Ed. Berlin, Germany: Springer, 1995, ch. 1, pp. 1–17.
[116] N. W. Paton and O. Díaz, "Active database systems," *ACM Comput. Surv.*, vol. 31, no. 1, pp. 63–103, Mar. 1999, doi: 10.1145/311531.311623.
[117] A. Ramírez, N. Moreno, and A. Vallecillo, "Rule-based preprocessing for data stream mining using complex event processing," *Expert Syst.*, vol. 38, no. 8, Dec. 2021, Art. no. e12762, doi: 10.1111/exsy.12762.
[118] A. Mattioli and F. Paternò, "Recommendations for creating trigger-action rules in a block-based environment," *Behav. Inf. Technol.*, vol. 40, no. 10, pp. 1024–1034, Jul. 2021, doi: 10.1080/0144929X.2021.1900396.
[119] Apache Spark. *Spark*. Accessed: 2023. [Online]. Available: https://spark.apache.org
[120] D. Kastrinakis and E. G. M. Petrakis, "Video2Flink: Real-time video partitioning in Apache Flink and the cloud," *Mach. Vis. Appl.*, vol. 34, no. 3, p. 42, May 2023, doi: 10.1007/s00138-023-01391-5.
[121] A. Berry and Z. Milosevic, "Real-time analytics for legacy data streams in health: Monitoring health data quality," in *Proc. 17th IEEE Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, Vancouver, BC, Canada, Sep. 2013, pp. 91–100.
[122] Qualtrics. *Qualtrics*. Accessed: 2023. [Online]. Available: https://www.qualtrics.com/
[123] Vanderbilt. *REDCap*. Accessed: 2023. [Online]. Available: https://www.project-redcap.org/
[124] DataWrangler. *Stanford Visualization Group*. Accessed: 2023. [Online]. Available: http://vis.stanford.edu/wrangler/
[125] OpenRefine. *OpenRefine User Manual*. Accessed: 2023. [Online]. Available: https://docs.openrefine.org/
[126] Oracle. *MySQL*. Accessed: 2023. [Online]. Available: http://www.mysql.com/
[127] *PostgreSQL*. Accessed: 2023. [Online]. Available: http://www.postgresql.org/
[128] Apache Cassandra. *Open Source NoSQL Database*. Accessed: 2023. [Online]. Available: http://cassandra.apache.org/
[129] *MongoDB*. Accessed: 2023. [Online]. Available: http://www.mongodb.org/
[130] SAS. *SAS Analytics Software*. Accessed: 2023. [Online]. Available: https://www.sas.com/
[131] StataCorporation. *Stata*. Accessed: 2023. [Online]. Available: https://www.stata.com/
[132] *IBM SPSS Statistics*. Accessed: 2023. [Online]. Available: https://www.ibm.com/products/spss-statistics
[133] B. D. Lund and T. Wang, "Chatting about ChatGPT: How may AI and GPT impact academia and libraries?" *Library Hi Tech News*, vol. 40, no. 3, Feb. 2023, doi: 10.1108/LHTN-01-2023-0009.
[134] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining Anal.*, vol. 3, no. 2, pp. 85–101, Jun. 2020, doi: 10.26599/BDMA.2019.9020015.
[135] S. Bou, H. Kitagawa, and T. Amagasa, "CPiX: Real-time analytics over out-of-order data streams by incremental sliding-window aggregation," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5239–5250, Nov. 2022, doi: 10.1109/TKDE.2021.3054898.
[136] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine Learning for Data Streams: With Practical Examples in MOA*. Cambridge, MA, USA: MIT Press, 2018.
[137] Tensorflow. *Robust Machine Learning on Streaming Data Using Kafka and Tensorflow-IO*. Accessed: 2023. [Online]. Available: https://www.tensorflow.org/io/tutorials/kafka
[138] P. Ksieniewicz and P. Zyblewski, "Stream-learn—Open-source Python library for difficult data stream batch analysis," *Neurocomputing*, vol. 478, pp. 11–21, Mar. 2022, doi: 10.1016/j.neucom.2021.10.120.
[139] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: Machine learning for streaming data in Python," *J. Mach. Learn. Res.*, vol. 22, no. 110, pp. 1–8, 2021. [Online]. Available: https://hdl.handle.net/10289/14402
[140] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, "Kafka-ML: Connecting the data stream with ML/AI frameworks," *Future Gener. Comput. Syst.*, vol. 126, pp. 15–33, Jan. 2022, doi: 10.1016/j.future.2021.07.037.
[141] Apache. *Flink ML*. Accessed: 2023. [Online]. Available: https://nightlies.apache.org/flink/flink-ml-docs-stable/
[142] N. N. T. Luong, Z. Milosevic, A. Berry, and F. Rabhi, "An open architecture for complex event processing with machine learning," in *Proc. IEEE 24th Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, Oct. 2020, pp. 51–56, doi: 10.1109/EDOC49727.2020.00016.
[143] C.-J. Su and S.-F. Huang, "Real-time big data analytics for hard disk drive predictive maintenance," *Comput. Electr. Eng.*, vol. 71, pp. 93–101, Oct. 2018.
[144] Refinitiv. Accessed: 2023. [Online]. Available: https://www.refinitiv.com/
[145] Z. Milosevic, W. Chen, A. Berry, and F. A. Rabhi, "An open architecture for event-based analytics," *Int. J. Data Sci. Anal.*, vol. 2, nos. 1–2, pp. 13–27, Dec. 2016, doi: 10.1007/s41060-016-0029-7.

[146] W. Chen and F. A. Rabhi, "Enabling user-driven rule management in event data analysis," *Inf. Syst. Frontiers*, vol. 18, no. 3, pp. 511–528, Jun. 2016, doi: 10.1007/s10796-016-9633-2.
[147] FHIR. *HL7 FHIR*. Accessed: 2023. [Online]. Available: https://build.fhir.org
[148] S. Dash, S. K. Shakyawar, M. Sharma, and S. Kaushik, "Big data in healthcare: Management, analysis and future prospects," *J. Big Data*, vol. 6, no. 1, p. 54, Dec. 2019, doi: 10.1186/s40537-019-0217-0.
[149] Selerity. *What's the Potential for Real-Time Analytics in Healthcare?*. Accessed: 2023. [Online]. Available: https://seleritysas.com/blog/2020/09/02/whats-the-potential-for-real-time-analytics-in-healthcare/
[150] Beamtree. *Beamtree*. Accessed: 2023. [Online]. Available: https://beamtree.com.au/
[151] J. Grimes, P. Szul, A. Metke-Jimenez, M. Lawley, and K. Loi, "Pathling: Analytics on FHIR," *J. Biomed. Semantics*, vol. 13, no. 1, p. 23, Sep. 2022, doi: 10.1186/s13326-022-00277-1.
[152] S. V. Nath, P. van Schalkwyk, and D. Isaacs, *Building Industrial Digital Twins: Design, Develop, and Deploy Digital Twin Solutions for Real-World Industries Using Azure Digital Twins*. Birmingham, U.K.: Packt, 2021.
[153] S. M. Jain, "Hugging face," in *Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems*. Berkeley, CA, USA: Apress, 2022, pp. 51–67.

**WEISI CHEN** (Member, IEEE) was born in Harbin, China, in 1988. He received the B.E. degree in computer science and technology from Zhejiang University, China, in 2011, and the Ph.D. degree in computer science and engineering from the University of New South Wales, Australia, in 2015.

He is currently a Lecturer with the School of Software Engineering, Xiamen University of Technology, China. He has extensive experience in both academia and industry for more than ten years. His research interests include machine learning-enabled big data analytics, such as natural language processing, predictive analysis, and complex event processing. He has been awarded the High-Level Talent of Xiamen, China.

**ZORAN MILOSEVIC** (Senior Member, IEEE) is currently an Interoperability and AI Consultant with Best Practice Software, Australia. He has been involved in many digital health interoperability projects and standards in Australia, the USA, and Europe. He endeavors to bring the latest research results and established architecture practices from enterprise distributed systems, such as ISO/ITU-T RM-ODP standards, into the evolving digital health ecosystems. He was a Co-Founder of Deontik, a boutique Australian IT company (https://deontik.com), excelling in the delivery of high-quality consulting services and innovative software. He is a Founder of the International Enterprise Design, Operations, and Computing (EDOC) Conference and is an Adjunct Professor at the Institute for Integrated and Intelligent Systems, Griffith University.

**FETHI A. RABHI** is a Professor with the School of Computer Science and Engineering, University of New South Wales (UNSW), Australia. He has a 35-year experience in conducting research projects as well as teaching subjects in computer science, software engineering, and information systems. Besides his academic role, he is actively engaged in facilitating university–industry collaborations, particularly in the context of multidisciplinary applied research. His initial involvement was in physics and engineering applications, and over the last years has moved to the application of IT technologies to the design of financial software systems and big data analysis.

**ANDREW BERRY** is a hands-on software architect with skills in software development, scalable software architecture, enterprise architecture, and integration. He has a unique understanding of horizontal scalability through both hands-on experience and Ph.D. research in distributed systems. He was a Co-Founder of Deontik, a boutique Australian IT company (https://deontik.com), excelling in the delivery of high-quality consulting services and innovative software. He has provided strategic technical leadership in all recent roles, applying his knowledge, experience, and vision to client solutions.