IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 July 2023, accepted 5 August 2023, date of publication 9 August 2023, date of current version 16 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303810

SURVEY

A Survey on Networked Data Streaming With Apache Kafka

THEOFANIS P. RAPTIS AND ANDREA PASSARELLA
Institute of Informatics and Telematics, National Research Council, 56124 Pisa, Italy
Corresponding author: Theofanis P. Raptis (theofanis.raptis@iit.cnr.it)

This work was supported by the European Union's Horizon 2020 Research and Innovation Program MARVEL under Grant 957337.

ABSTRACT Apache Kafka has become a popular solution for managing networked data streaming in a variety of applications, from industrial to general purpose. This paper systematically surveys the research literature in this field by carefully classifying it into key macro areas, namely algorithms, networks, data, cyber-physical systems, and security. Through this meticulous classification, the paper aims to identify and analyze the optimization aspects relevant to each area, drawing upon practical applications as the basis for analysis. In this respect, the paper synthesizes and consolidates existing knowledge, saving researchers valuable time and effort in searching for relevant information across multiple sources. The tangible benefits of this survey paper include providing a consolidated knowledge base about research-intensive Apache Kafka topics, highlighting practical insights and novel approaches, pointing up cross-domain applications, identifying related research challenges, and serving as a trusted reference for the Apache Kafka community.

INDEX TERMS Algorithms, cyber-physical, data, Internet of Things, networks, pub-sub, security, stream processing.

I. INTRODUCTION

Networked data streaming is an essential methodological paradigm that has become increasingly important in today's fast-paced digital landscape. As shown in Fig. 1, its workflow involves several steps to transmit and process real-time data: Real-time data are generated by various networked sources such as sensors, devices, or software applications, and transmitted over a computer network such as the Internet or a private network. The real-time data is then published to a message broker, which acts as a central hub that receives and distributes data to multiple subscribers in real-time. This process is known as publish/subscribe (pub/sub) and enables the data to be shared efficiently across multiple applications or processing systems. The data is then processed in real-time using stream processing technology. Stream processing involves applying algorithms or rules to the data stream as it flows through the system, allowing for immediate analysis, aggregation, filtering, or transformation of the data. The processed data is then outputted as a stream of information in a structured format, often in real-time. This output stream can be consumed for further analysis or decision-making by multiple applications, dashboards, or visualization tools that use the processed data for different purposes, such as generating alerts, updating databases, or triggering automated actions.

With the massive amounts of data generated every day, it has become crucial for organisations to process and analyse this data in real-time. Networked data streaming allows organisations to provide an efficient way to rapidly transmit data from various sources to a central location for processing and analysis. The applications of networked data streaming are varied and extensive. For example, it can be used for real-time monitoring of smart city network traffic and financial transactions [1], as well as analysing sensor data in manufacturing to predict equipment failure [2] and sensing and actuating on large smart agricultural fields [3]. By providing real-time insights into business operations, networked data streaming enables organisations to make informed decisions faster, improving overall efficiency and competitiveness. One of the most significant benefits of networked data streaming is its ability to identify potential

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

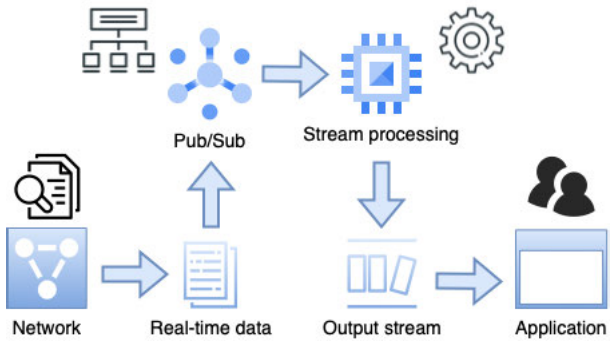

FIGURE 1. Networked data streaming.

issues before they escalate, reducing downtime and improving productivity.

Designing a networked data streaming system using publish/subscribe (pub/sub) technologies is a popular approach to handling and analysing large volumes of data in real-time (figure 1). Pub/sub is a messaging paradigm that allows publishers to send messages to subscribers who have expressed interest in specific topics. Using pub/sub technologies in networked data streaming offers several benefits [4]. First, it provides a flexible and scalable solution that can accommodate changing data volumes and sources. Second, it enables efficient data processing by distributing it across multiple nodes in the network. Finally, it allows for seamless integration with other systems and services, making it easier to leverage the data for various use cases. When designing a pub/sub-based networked data streaming system, it is essential to consider factors such as scalability, fault tolerance, and data security. It is also important to select appropriate pub/sub technologies and configure them correctly to ensure optimal performance and reliability.

Apache Kafka [5] is a popular open-source distributed streaming platform that is widely used for building networked data streaming applications, such as LinkedIn [6]. It provides a pub/sub messaging system that allows data to be processed in real-time and distributed across multiple nodes in the network. Using Apache Kafka as the pub/sub system for networked data streaming provides a highly scalable and fault-tolerant solution that has the potential to handle massive amounts of data with ease. However, when using Apache Kafka as the pub/sub system for networked data streaming, it is important to consider optimisation factors such as data serialisation, message size, and partitioning and therefore to configure Apache Kafka correctly to ensure the desired performance and reliability. As a result, there has been a significant amount of research conducted on various aspects of Apache Kafka to better understand its capabilities, limitations, and potential applications. As an open-source platform, it is constantly evolving, with new features and improvements being added regularly. This presents researchers with opportunities to explore and experiment with the platform and develop new use cases and applications. The vibrant research on Apache Kafka reflects its growing importance as a data streaming platform and its ability to constantly revolutionise how organisations handle and analyse data in real-time.

In this paper, we survey the research literature of Apache Kafka for networked data streaming so as to provide a comprehensive overview of the current state of knowledge on the platform. There has been a significant amount of research conducted on various aspects of Apache Kafka, including its architecture, performance, scalability, and security. However, as far as we know, no study literature has addressed the knowledge organisation on the topic in a synthetic manner. To fill this gap, we synthesise and summarise the existing literature on Apache Kafka, in order to provide insights into the current state of the art and identify gaps and opportunities for future research. The paper can serve as a valuable resource for practitioners who are looking to leverage Apache Kafka for their data streaming needs, providing a comprehensive overview of the platform's capabilities, limitations, and potential use cases. Overall, our paper provides the following contributions:

1) A research literature classification in representative macro areas (algorithms, networks, data, cyber-physical, and security) and identification of the corresponding optimisation aspects, based on practical applications
2) Exploration of the algorithmic foundations of Apache Kafka, including its combinatorial and reliability aspects
3) Coverage of the area of networked infrastructure optimisation, examining how Kafka can improve the performance and scalability of distributed systems
4) Discussion on data handling and processing, highlighting how Apache Kafka can be used for real-time data streaming, message queuing, and ML-based computing
5) Investigation of the emerging trend of cyber-physical convergence and the role of Apache Kafka in integrating physical systems with data-driven applications, in the contexts of Internet of Things, vehicles, mobility and environmental use cases
6) Coverage of security considerations related to Apache Kafka and of how the platform can be used securely in enterprise environments
7) Identification of selected open research challenges for networked streaming with Apache Kafka are identified.

After evaluating the state of the art in section II, and, as displayed in Fig. 2, presenting the Apache Kafka basics in section III, we present and in-depth survey on Apache Kafka, analysing the latest research and developments in several critical areas related to Kafka. The paper is structured into several sections, each focusing on specific Apache Kafka-related topics. Section IV delves into the algorithmic foundations of Apache Kafka, exploring its architecture and design principles. Section V covers networked infrastructure optimisation and examines how Kafka can improve the performance and scalability of distributed systems. Section VI explores data handling and processing, discusses how Apache Kafka can be used for real-time data streaming, message queuing, and ML-based computing. Section VII

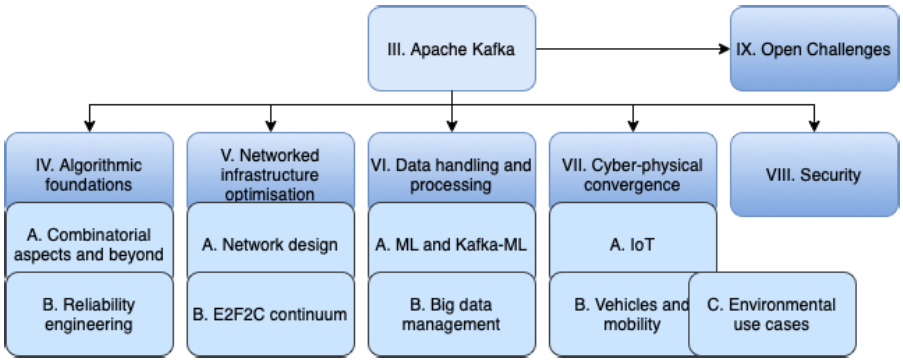

FIGURE 2. Roadmap of the paper.

discusses the emerging trend of cyber-physical convergence, highlighting the role of Apache Kafka in the integration of physical systems with data-driven applications. Section VIII focuses on security considerations related to Apache Kafka, exploring how the platform can be used securely in enterprise environments. Finally, in Section IX, we identify some open research challenges.

II. METHODOLOGY AND LITERATURE OVERVIEW

This survey paper follows a systematic literature review methodology to provide a comprehensive analysis of the existing research on Apache Kafka-related topics. The systematic review process involved several key steps:

1) Research question formulation: We defined clear research questions to guide our literature search and analysis. These questions address the key aspects of Apache Kafka, including its architecture, performance, scalability, reliability, security, and integration with other systems.
2) Search strategy: We developed a comprehensive search strategy to identify relevant studies. The search was conducted in various academic databases, including IEEE Xplore, ACM Digital Library, and Google Scholar. We also searched conference proceedings, industry reports, and technical documentation to ensure a broad coverage of the literature. We combined and utilized a series of keywords in order to achieve the maximum coverage of the area, such as Kafka, streaming, data, service, network.
3) Study selection: We applied predefined inclusion and exclusion criteria to select studies that met our research objectives. The criteria considered the relevance of the study to Apache Kafka, the publication date, and the quality of the research.The main search criteria for related publications is the presence of notable use or advancement of Apache Kafka and its applicability in the context of networked streaming. The number of found papers was more than 90. However, based on our own judgement and using as exclusion criterion the lack of a solid Apache Kafka contribution in the core of a given work, the final number of papers considered was in the end up to 70.
4) Data extraction and analysis: We extracted relevant data from the selected studies, including information on the study's objectives, methodology, findings, and contributions. We analyzed the extracted data to identify common themes, emerging trends, and research gaps in the field of Apache Kafka.
5) Quality assessment: We assessed the quality of the selected studies using established criteria such as the relevance of the research question, the rigor of the methodology, and the validity of the findings.
6) Synthesis and reporting: We synthesized the findings from the selected studies and organized them thematically. The results are presented in a structured and coherent manner in the survey paper, providing insights into the current state of research on Apache Kafka-related topics.

Although Apache Kafka is already a commercially popular platform, to the best of our knowledge there is a very limited set of past papers which partially report some systematic research advancements on the field in a comprehensive manner. For this reason, our browsing was extended to papers which explore the pros and cons of Apache Kafka at large. We list those papers in Table 1 and we compare them to our contribution. It is noteworthy that none of the identified papers is a pure survey paper; they are rather technical contributions which, however, offer some detailed outline of their research field of reference. Specifically, we conducted the comparison according to the fundamental thematic parts of the current paper: (i) algorithmic foundations, (ii) networked infrastructure, (iii) data handling, (iv) cyber-physical systems, and, (v) security. As we can see in Table 1, there is no paper that covers all the thematic parts. Also, due to the fact that the past papers were published between 2015 and 2021, the current paper naturally covers a more up-to-date perspective on the topic. Last but not least, to the best of our knowledge, the current paper systematically outlines for the first time in the state of the art the cyber-physical and the security aspects of the literature.

III. APACHE KAFKA

Apache Kafka is a distributed streaming platform that is designed to handle massive amounts of data in real-time. At its core, an Apache Kafka cluster provides a

TABLE 1. Comparison with past papers surveying Apache Kafka elements (2015-2023).

Reference	Year	Algorithmic foundations	Networked infrastructure	Data handling	Cyber-physical	Security
current paper	2023	✓	✓	✓	✓	✓
[7]	2021	✗	✓	✓	✗	✗
[8]	2021	✗	✓	✗	✗	✗
[9]	2019	✓	✓	✓	✗	✗
[10]	2018	✗	✗	✓	✗	✗
[11]	2015	✗	✗	✓	✗	✗

publish-subscribe messaging service (Fig. 3), and a pub/sub messaging system that allows producers to publish data to Kafka topics and consumers to subscribe to those topics and receive data as it arrives. Producers in Kafka are responsible for publishing data to Kafka topics. They can publish data in any format, including text, binary, or JSON. When a producer publishes data to a Kafka topic, it sends a message that includes a key and a value. The key is used to identify the message and can be used for partitioning and indexing. The value contains the actual data payload.

Apache Kafka topics are logical categories or streams of data. They are created by administrators and can have multiple producers and consumers. Topics can be partitioned, which allows for parallel processing of messages and increased scalability. When a message is published to a topic, it is appended to the end of the topic's log. Partitions in Kafka are the basic unit of parallelism. Each partition is a sequence of messages that is stored on a single broker node. When a message is published to a partition, it is assigned a sequential offset that represents its position within the partition. Consumers can read messages from a partition in parallel, which allows for high throughput and scalability. Replicas in Kafka are copies of partitions that are stored on multiple broker nodes. Replication provides fault tolerance and ensures that data is not lost in the event of a broker failure. Kafka supports configurable replication factors, which specify the number of replicas that should be created for each partition.

Consumers in Apache Kafka are responsible for subscribing to Kafka topics and reading messages from them. They can read messages from one or more partitions and can process messages in parallel to achieve high throughput. Consumers can also group together to form consumer groups, which allows for load balancing and failover. Kafka supports both push and pull-based consumption models. In the push model, Kafka sends messages to consumers as soon as they are available. In the pull model, consumers request messages from Kafka and receive them in batches. Kafka also provides support for stream processing. Stream processing involves processing data in real-time as it arrives in Kafka, rather than storing it in a database for batch processing later. Kafka Streams is a Java library that provides a high-level application programming interface (API) for building stream processing applications on top of Kafka.

In Fig. 3, an illustrative example of an Apache Kafka cluster with four brokers, b_1, b_2, b_3, and b_4, and two topics, τ_1 and τ_2, each with multiple partitions is displayed. Additionally, there are two data producers, p_1 and p_2, and four data consumers, c_1, c_2, c_3, and c_4, which can subscribe

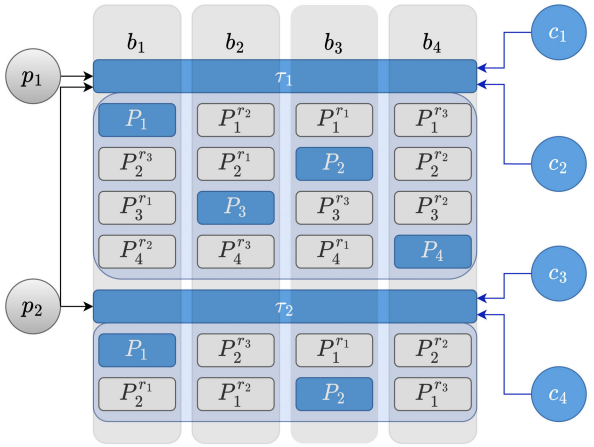

FIGURE 3. An example of an Apache Kafka cluster of four brokers b_1, b_2, b_3, b_4 for two given topics τ_1, τ_2, two data producers p_1, p_2, four data consumers c_1, c_2, c_3, c_4, and replication factor $r = 3$. Leader partitions in blue, replicas in gray. Partitions are typically allocated to brokers via a topic partitioning allocation algorithm.

to and consume data from the topics. Partitions are used to break down a topic into smaller, more manageable chunks of data that can be distributed across multiple brokers. Each partition is replicated multiple times, with a replication factor of three in this case, to ensure fault tolerance and data redundancy. The blue-colored partition is the leader partition, which is responsible for handling all read and write operations for a given partition. The gray-colored replicas are backups of the leader partition, which take over if the leader fails. When a data producer, such as p_1 or p_2, sends a message to a Kafka topic, the message is first received by the Kafka broker that is the leader partition for the partition to which the message is being sent. The leader partition then writes the message to its local disk and sends copies of the message to the other replicas of that partition. Once the replicas have acknowledged receipt of the message, the leader partition sends an acknowledgment back to the producer. Data consumers, such as c_1, c_2, c_3, and c_4, can subscribe to one or more Kafka topics and consume messages from the partitions assigned to them. When a consumer joins a topic, it is assigned one or more partitions to consume from, and each consumer group is guaranteed to receive all messages from each partition assigned to them. In summary, Apache Kafka works by breaking down topics into partitions and replicating them across multiple brokers, allowing for fault tolerance and data redundancy. Data producers send messages to the leader partition of a partition, which then distributes the messages to the replicas. Data consumers can subscribe to one or more Kafka topics and consume messages

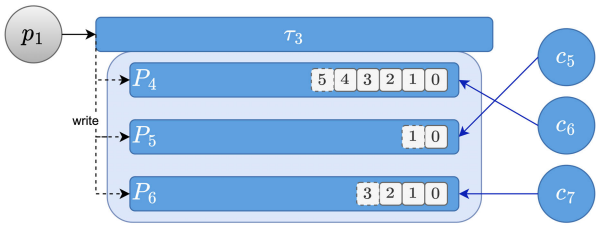

FIGURE 4. An example of an Apache Kafka write/read process of one given topics τ_3, one data producer p_3, and three data consumers c_5, c_6, c_7.

from the partitions assigned to them. This architecture allows for efficient and scalable real-time data transfer across multiple applications and services.

An illustrative example of what happens when a producer generates a message is displayed in Fig. 4. In the image, we have one Kafka topic τ_3, one data producer p_1, and three data consumers c_5, c_6, c_7. The topic is divided into three partitions P_4, P_5, P_6 for scalability and performance reasons. First, let us look at the write process. The data producer sends a message to the Apache Kafka broker by specifying the name of the topic and the partition. Except for its value, the message contains a numbered key, and the broker appends the message to the end of the partition's log. Each message is assigned a unique offset within the partition that represents its position within the partition. This offset is used by the consumers to track their progress and ensure they do not miss any messages.

Now let us look at the read process. Each data consumer subscribes to the topic and one of the partitions they are interested in. When a consumer first subscribes, it specifies the offset from which it wants to start reading messages. The consumer can choose to start reading from the earliest available offset, which means it will read all messages from the beginning of the partition, or from a specific offset, which means it will start reading from that position in the partition's log. As messages are written to the partitions, they become available for consumption. In the example, the consumers use a pull-based model to read messages from the partitions. The consumer sends a request to the broker asking for a batch of messages starting from its current offset position. The broker responds with a batch of messages, and the consumer processes them one by one. Once a message is processed, the consumer updates its offset position to the next message in the partition.

In our example, we have three consumers reading from the same topic, each reading from a different partition. Each consumer maintains its own offset position and reads messages independently. This means that messages can be processed in parallel, which provides high throughput and scalability. It is important to note that Apache Kafka also supports consumer groups, which allow multiple consumers to work together to read from a partition. In a consumer group, each consumer is assigned a subset of the partition's offsets to read from. This ensures that each message is only processed once, even if there are multiple consumers in the group. In this example, we also assumed that there are no replicas of the partition. However, in a production environment, it is common to have multiple replicas of each partition for fault tolerance and high availability, as displayed in Fig. 3. Replicas are simply copies of the partition's log that are stored on different broker nodes. When a broker fails, one of the replicas can take over and continue serving requests.

IV. ALGORITHMIC FOUNDATIONS

The algorithmic foundations of Apache Kafka (Table 2) are crucial to achieving combinatorial optimisation of the topic partitioning process and ensuring reliability engineering in large-scale data processing and streaming applications. These foundations are based on core principles that underlie the design and functionality of the Kafka platform. In this subsection, we will delve into the key components of the Kafka architecture, including its distributed messaging system, its partitioning and topic organisation of data, and its replication and fault-tolerance mechanisms. We will also explore the core design principles that have guided the development of Kafka, such as its focus on simplicity, scalability, and performance. Understanding of the algorithmic foundations of Kafka, researchers and practitioners can optimise the use of its capabilities to meet the complex data management needs of modern applications while ensuring reliability engineering.

A. COMBINATORIAL ASPECTS AND BEYOND

The combinatorial aspects of Apache Kafka form a significant part of its algorithmic foundations, providing powerful approaches for optimising large-scale data processing and streaming. Combinatorial optimisation is concerned with finding the best solutions from a finite set of possibilities. In the context of Kafka, this involves maximising the throughput of data streams, reducing latency, and minimising resource usage. The literature offers a considerable amount of combinatorial optimisation techniques used in Apache Kafka, such as batch processing, compression, and batching. By understanding the combinatorial aspects and beyond of Apache Kafka, researchers and practitioners can explore new ways of optimising data processing and streaming to meet the ever-increasing demands of modern applications.

A first category of optimisations focuses on data transfer modelling and design. Despite the large and growing user community, there remains a significant gap in formal modeling approaches that can be used to reason about the behavior of producers and consumers in Kafka-based systems. One key challenge in formalizing Kafka's data transfer model is that it involves multiple layers of abstraction, from low-level network protocols to high-level message processing semantics. Additionally, there are many different ways that producers and consumers can interact with Kafka, depending on factors such as partitioning strategies, replication settings, and client library configurations. All of these factors make it difficult to develop general models that accurately capture the behavior of real-world Kafka systems.

TABLE 2. Algorithmic foundations in Apache Kafka.

Article	Modelling consideration	Methodology
Combinatorial aspects		
[12]	Data transfer	Formal methods
[13]		Queueing theory
[14]	Partitioning	Integer programming, heuristics
[15]		Bin packing, R-score
[16]	Consensus	Control theory
[17]	Data starvation	Load shedding
[18]	Service overhead	Simulation modelling
Reliability engineering		
[19]	Fault tolerance	Checkpoint interval values optimisation
[20]		Cooperative clustering
[21]		Latency engineering
[22]		Architectural configurations
[23]	Reliability estimation	
[24]	Reliability prediction	Neural networks
[25]	Reliability assesment	Testing tool

Following this reasoning, in [12], the communication in Apache Kafka between producers and consumers is modelled using formal methods. Selected system characteristics are verified using the model testing tools. The verification findings demonstrate that the Apache Kafka data transfer model adheres to its specifications, which leads to the conclusion that the system is trustworthy. In [13], in order to forecast performance measures of Apache Kafka cloud services, the authors analyse the structural characteristics of Apache Kafka and suggest a data transfer model inspired by queuing methods. The number of brokers in the Apache Kafka cluster, the number of partitions in a topic, and the size of the data batch are the initial configuration inputs for this approach. The effect of specific setup factors on performance measures, such as producer output, relative payload and overhead, and changes in disk storage utilisation over time, can be determined by users using this model. In order to assess the end-to-end delay of messages, queuing theory is used.

A second important category of works is on how to solve issues related to topic partitioning or to partition-consumer assignments. Although Apache Kafka includes some out-of-the-box optimisations, the authors of [14] point out that it does not explicitly specify how every topic should be partitioned in order to be distributed effectively. In this regard, they first simulate how Apache Kafka partitions topics for a specific subject. They then pose the optimisation problem of determining the required number of partitions and demonstrate that it is computationally hard (it can be formulated as an integer program). The authors suggest two straightforward but effective methods to fix the issue: the first aims to maximise the number of brokers used in the cluster, while the second minimises it. The authors of [15] use bin packing problem variation to abstract the challenge of finding the necessary number of consumers and the partition-consumer assignments. They suggest indicative metrics that take the rebalancing expenses into consideration, and introduce and evaluate a variety of methods in comparison to known strategies for the bin packing problem in this context.

The remaining works on combinatorial aspects with Apache Kafka focus on a variety of different problems. We group them in this paragraph and we provide a brief description; they mainly are on (i) Kafka-based consensus algorithms, (ii) data starvation modelling, and, (iii) mobility simulation modelling. In [16], the authors look into an adaptive tuning method to calibrate the parameters related to an Apache Kafka-based consensus algorithm for blockchain-specific use case applications. Specifically, their method is based on feedback control theory, and targets to adjust the parameters connected to its consensus algorithm in order to address the sudden abundant inflow of data and quickly adapt to the current system workloads. According to the authors of [17], data starvation may occur if Kafka's data production rate outpaces its consumption rate. A load shedding method is introduced to restrict the incoming data and keep system efficiency when the system is under stress in order to address the starvation issue. In [18], a simulation platform that allows assessments of potential future mobility use cases is described by the authors. To support all of these needs and the coupling of various simulation tools into a co-simulation, Apache Kafka's is used as a communication building block.

B. RELIABILITY ENGINEERING

The reliable and efficient transmission of data streams is a critical aspect of networked data streaming. As data streams grow in size and complexity, the need for sophisticated algorithms and systems that can handle them increases. The algorithmic foundations of Apache Kafka provide a set of tools and techniques that are specifically designed to address these challenges. Algorithmic design can be used to optimise the reliability and availability of data streams, particularly in the face of potential failures or disruptions. We present the various approaches that have been designed to achieve these goals, including techniques for fault tolerance, partitioning, replication, and load balancing. Partitioning and replication are key techniques used to optimize the reliability and availability of data streams in Kafka. Partitioning allows Kafka to break down a topic into smaller, more manageable chunks of data that can be distributed across multiple brokers. This helps to ensure that each partition can be processed independently, which improves the overall performance and scalability of the system. Additionally, replication is used to provide fault tolerance and data redundancy by ensuring that each partition is replicated multiple times, with replicas distributed across multiple brokers.

Fault tolerance for improving reliability is a major design goal in a set of papers in the literature. The authors of [19], suggest that, in parallel to replication, message recovery checkpointing can serve an an alternative fault tolerance approach. By defining ideal checkpoint interval values that have an effect on the data resilience of the Apache Kafka workflow, they encourage the enhancement of fault tolerance in the design. The authors estimate the overall total overhead cost after defining the optimal checkpoint interval, and they fine-tune it with respect to maximizing the lost message

recovery rate. It has been demonstrated that the use of the ideal checkpoint interval time, significantly reduces the amount of lost data. According to the comparative study, the changed system enhances Apache Kafka's ability to recover data. In [20], the authors investigate the challenges of disaster recovery fault tolerance for Apache Kafka and introduce an approach of spatially-cooperative and redundant Kafka clusters to boost resiliency. In [21], the authors tackle the problem of Apache Kafka's message delivery delay in settings where network faults can happen, and they conclude that the batch size has a direct impact on the data loss rate, especially when the network connection is not stable. Last but not least, in [22], the authors investigate the impact on the reliability performance of different configuration parameters of Apache Kafka, including partition replication for fault tolerance.

Architectural benchmarking can also help measure various properties of an Apache Kafka cluster. According to [23], projections of the performance effect of various Apache Kafka configurations are not always accurate. The authors emphasise on unexpected behaviours that were found in Kafka data operations. Specifically, (i) two independently executed data producers do not double the data input rate compared to a single producer, as expected, and (ii) the observed memory usage was never close to its limits for non of the presented scenaria. However, the research demonstrates that not all observed instances support the hypothesis. In [24], the authors demonstrate that the alteration of the configurations under normal circumstances (such as the utilization of network bandwidth and the mean service rate of Kafka producers) is able to impact the data delivery properties. They employ reliability prediction of Kafka given various configurations and network conditions, and define two reliability metrics to be predicted, the probability of data loss and the probability of data duplication. Artificial neural networks are applied in the prediction model and some key parameters are selected, as well as network metrics as the features. Finally, in [25], the authors present the design of a test framework for assessing the reliability aspects of Apache Kafka in order to investigate diverse data delivery approaches under sub-optimal network quality. Two metrics, data loss rate and duplicate rate, are used in the experiments in order to evaluate the reliability of data delivery in Kafka. The experimental results show that under high network delay the size of data matters.

V. NETWORKED INFRASTRUCTURE OPTIMIZATION

As more and more applications move to the cloud and edge computing environments, the need for efficient and scalable network infrastructures becomes increasingly critical. Apache Kafka offers a robust platform for networked data streaming, enabling seamless data exchange and processing across distributed systems. This section focuses on the networked infrastructure optimisation capabilities of Apache Kafka, with an emphasis on its network design and Edge-to-Fog-to-Cloud (E2F2C) architecture. First, we explore how Apache Kafka can be used to optimise network performance and scalability and we examine the various network design considerations that must be taken into account when implementing with Apache Kafka, such as pipelining, downstream/upstream alteration, as well as smart queuing and filtering. Then, we focus on how Kafka can be used to optimise data streaming and processing across distributed systems in the E2F2C continuum. We discuss how the various architectural options can be used to improve data processing efficiency and reduce latency, by pushing data processing to the edge and leveraging fog and cloud resources. Additionally, we examine the various technologies and techniques used to implement the E2F2C architecture, such as third-party resource consumption, over-the-air resource allocation, as well as load shredding.

A. NETWORK DESIGN

Network design is a crucial task for ensuring the efficient operation of networked systems. In the context of networked data streaming with Apache Kafka, network design refers to the process of designing the network topology and protocols to ensure reliable data transmission and low latency. It involves decisions such as the number and location of brokers, the configuration of Kafka producers and consumers, and the choice of communication protocols. Efficient network design is critical for achieving high throughput and low latency in streaming applications. In this section, we will review the key works on network design for Apache Kafka and discuss the different approaches and techniques proposed in the literature.

In [26], the authors separate the streaming process in two different parts and evaluate the delays for two diverse deployments to determine if a typical streaming application is network intensive enough to benefit from a faster interconnect. Moreover, they explore whether the volume of input data stream has any effect on the latency characteristics of the streaming pipeline, and if so how does it compare for different stages in the streaming pipeline and different network interconnects. In [27], the authors discover that, rather than being pushed to and replicated in downstream locations, filtering on big datasets is best done in a shared upstream location. They evolve Apache Kafka to conduct restricted data operations, taking overs some operational processes form the downstream applications, to illustrate the benefits of such a strategy. In comparison to standard Kafka, their method scores higher than four popular analytics pipeline designs with minimal overhead. In [28], the authors present a streaming mechanism for optical networks based on the Kafka architecture and protocols, to efficiently distribute state and network updates following the upcoming Open Networking Foundation Transport API streaming implementation agreement. The proposed mechanism is validated and experimentally evaluated. In [29], the authors design a distributed message system of Apache Kafka to support large-scale distributed messages between SDN controllers. The proposed system measured the message processing time of Kafka and the existing Message Queue and evaluated its performance. In order to accomplish effective

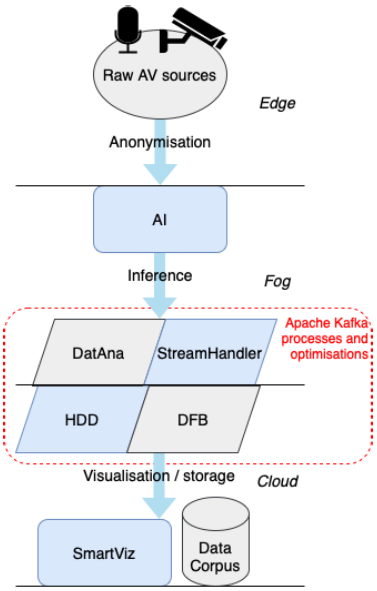

FIGURE 5. Example [31] Apache Kafka implementation in the E2F2C continuum.

rate correction, the authors of [30] design a consumer model that improves Apache Kafka with reliable, scalable, and intelligent filtering and queuing methods. Even in cases of extreme congestion, the introduced consumer model is able to ensure no lost data while limiting the number of retransmissions. The experimental findings show that, in terms of reliability, consumption rate, and output, the given approach outperforms the traditional Kafka consumer.

B. E2F2C CONTINUUM

The E2F2C continuum is a distributed computing architecture that integrates different layers of processing and analysis of data generated by edge devices. The architecture consists of three layers: Edge, Fog, and Cloud. The Edge layer represents devices that are physically located near the data source, while the Fog layer is a distributed layer that provides intermediate processing capabilities between the Edge and Cloud layers. The Cloud layer is the final layer that is responsible for storing and processing the collected data at scale. This architecture provides a framework for distributed processing, which is becoming increasingly important as organizations collect more data and require faster and more efficient processing. By leveraging the benefits of each layer, the architecture can enable real-time data processing, analysis, and decision-making. In this section, we will survey the works related to the Edge-to-Fog-to-Cloud Continuum in the context of Apache Kafka. An indicative Apache Kafka implementation within the E2F2C continuum is provided in [31] and its respective architecture is displayed in Fig. 5.

In [32], the authors present a methodology for augmenting the current Kafka-based reference architecture of an E2F2C use case by generalising it and permitting to extend the resources of a given Apache Kafka cluster with additional resources situated on third-party industrial cloud owners, leveraging on the functionalities of off-the-shelf products beyond the edge deployments. A framework dubbed Kafka On Hadoop (KOHA), created and implemented by the authors of [33], offers users a quick, easy, and effective approach to create a large-scale distributed Kafka-based application that runs on top of a Hadoop cluster. The architecture allots resources to launch producers and consumers and automatically builds and runs Kafka brokers on demand. Users do not need to comprehend the YARN programming model or make any attempts to set up a Kafka cluster in order to use the framework to embrace Apache Kafka. The ability of Apache Kafka to tolerate network faults is examined in [34]. The authors note that Apache Kafka exhibits some fault tolerance for network problems across various setups, and they also note some of its drawbacks. Additionally, they establish a benchmark for network failure tolerance that can be used to compare other distributed streaming systems. In [35], the authors evaluate various Apache Kafka configurations and performance measures to help users avoid bottlenecks and ultimately take advantage of best practices for effective stream processing. In [36], the authors introduce an Apache Kafka-based load shedding engine that works when the delay exceeds the benchmark and quickly manages overflow by discarding data in the Apache Kafka producer. Load sharing proved effective for both single and numerous sources in tests using Apache Storm.

VI. DATA HANDLING AND PROCESSING

With the proliferation of data in modern-day systems, the need for efficient and scalable big data management systems has become increasingly important. Kafka, with its ability to handle large volumes of data in real-time, provides a potential solution to this problem, as it provides a scalable and fault-tolerant solution for real-time data processing. In this section, we explore the various ways in which Kafka can be used for data handling and processing. We begin by examining how Kafka can be integrated with machine learning (ML) frameworks to enable efficient and scalable ML-based computations. Specifically, we delve into Kafka-ML, which is an open-source library that allows for easy integration of Kafka with popular ML frameworks such as TensorFlow, Keras, and Scikit-learn. We also discuss the benefits of using Kafka-ML for large-scale data processing and present some real-world use cases. In the second subsection of this section, we focus on big data management using Apache Kafka. We explore the various features of Kafka that make it suitable for big data management, including its distributed architecture, fault-tolerance, and scalability. We also discuss some of the challenges associated with using Kafka for big data management, such as data serialization and integration with other big data technologies.

A. ML AND KAFKA-ML

In recent years, ML has emerged as a powerful tool for analyzing and processing large amounts of data. However, the traditional approach of batch processing is not always suitable for real-time applications that require immediate responses to incoming data. To address this, a new paradigm of real-time

ML has emerged, which involves processing data streams using ML algorithms. Apache Kafka has become a popular choice for handling real-time data streams and integrating with ML frameworks. Kafka-ML is a powerful open-source library that enables the integration of ML algorithms into Kafka streams. This integration enables the development of real-time ML applications, which can provide immediate responses to incoming data.

In [37], Kafka-ML is introduced as a cutting-edge, open-source framework that enables the management of ML pipelines through data streams. Users may quickly construct ML models, train, test, and deploy them for inferences using Kafka-ML's accessible and user-friendly Web user interface. Through the use of containerization technologies, Kafka-ML and the components it uses are fully managed, guaranteeing their portability, ease of distribution, and other features like fault tolerance and high availability. The last step is the introduction of a unique method for managing and reusing data streams, which may do away with the necessity for file systems or data storage.

In [38], the authors expand the Kafka-ML framework to support the administration and implementation of distributed deep neural networks throughout the E2F2C Continuum as they claim that Kafka-ML does not take the distribution of deep neural network models into account. In order to provide quick forecasts, they have also thought about the potential of including early exits in the E2F2C layers. By modifying and deploying their model in three distinct scenarios, they assess its potential. In comparison to a cloud-only implementation, experiments show that Kafka-ML can greatly increase reaction time and throughput by distributing DNN models across the Cloud-to-Things Continuum.

The authors of [39] provide a technological framework that combines the benefits of BranchyNet (a neural network architecture where side branches are added to the main branch, the original baseline neural network, to allow certain test samples to exit early) with the Edge-Cloud architectural idea to allow fault-tolerant and low-latency AI predictions. Implementing and evaluating this methodology enables evaluating the advantages of using Distributed DNN (DDNN) along the continuum from Cloud to Things. The statistics collected demonstrate a response time improvement of 45.34% as compared to a Cloud-only deployment. Additionally, this proposal offers a Kafka-ML extension that lessens rigidity while managing and deploying DDNN across the Cloud-to-Things continuum.

In [40], the authors propose a method for automated analysis of heterogeneous news through complex event processing and ML algorithms. Initially, news content streamed using Apache Kafka, stored in Apache Druid, and further processed by a blend of natural language processing and unsupervised ML techniques.

In [41], the authors suggest creating KafkaFed, an information-centric networking-based scalable communication architecture, to enable the Federated Learning (FL) method. The Information Centric Networking (ICN)-based infrastructure enables for rapid data retrieval for mobile nodes while overcoming the drawbacks of conventional client-server designs for FL, which use content-based or name-based routing. To ensure effective and dependable data delivery, data are stored at intermediate nodes in the ICN network. In a simulated setting, a proof of concept for the KafkaFed communication architecture is created and tested. With just 32 clients, the suggested framework outperformed the client server-based FL architecture, or FLOWER, in terms of performance. It also had various additional benefits in terms of scalability, stability, and security.

As social media platforms continue to grapple with the challenge of fake accounts and automated activity, there is a growing interest in leveraging ML techniques to identify and track social bots in real-time. Recently, the authors of [42] have explored the use of Apache Kafka as a powerful streaming platform for processing social media data and applying ML algorithms to identify suspicious activity. To provide real-time identification of social bots on Twitter using ML, they use Apache Kafka to stream data from the Twitter API. They employ details from profiles as features. To forecast the kind of inbound data, an ML method is used.

B. BIG DATA MANAGEMENT

The increasing growth of data has led to the development of various techniques and tools for managing data efficiently. Apache Kafka, as a distributed streaming platform, provides a robust solution for managing large volumes of real-time data in a fault-tolerant and scalable manner. Kafka enables high-throughput, low-latency data ingestion, processing, and delivery, which is crucial for managing big data. Additionally, Kafka integrates with various big data processing frameworks such as Apache Spark, Apache Storm, and Apache Flink to support large-scale data processing and analytics.

In [43], the authors expand Apache Kafka by delivering a distributed complex event recognition system designed on top of Apache Kafka streams. The system's main goal is to reason about the semantics of Kafka stream operators. In order to do so, the authors design it with the abstraction operations of event construction, transformation and and composition.

The authors of [44] introduce KSQL, a streaming SQL engine for Apache Kafka. For stream processing on Apache Kafka, KSQL offers a straightforward and fully interactive SQL interface, eliminating the need to write code in a computer language like Java or Python. Open-source, distributed, scalable, trustworthy, and real-time, KSQL is also. Aggregations, joins, windowing, sessionization, and a broad range of other advanced stream processing functions are supported. Using User Defined Functions (UDFs) and User Defined Aggregate Functions (UDAFs), it is expandable. Since KSQL is built using the Kafka Streams API, it offers the exact-once delivery guarantee, linear scalability, fault tolerance, and the ability to operate as a library without a separate cluster.

Reference [45] focuses on object-knowledge-model grammar and how it can be used to analyse and recognize the metadata in streams based on Kafka. The suggested grammar is more flexible and proves to be better when

combined with other known NLP techniques. The authors assert that the suggested model is a better match for NLP comprehension and projecting the benefits of using them in knowledge management models because grammar tests the veracity of framing sentences (in different languages) and because each language has its own origins and connotations.

In [46], the architecture of an RSP engine that is based on cutting-edge Big Data frameworks, especially Apache Kafka and Apache Spark, is described by the authors. Together, they enable the development of a production-ready RSP engine that ensures high availability, scalability, fault tolerance, low latency, and high throughput. They also stress how much easier it is to develop complicated applications needing libraries for machine learning, graph processing, query processing, and stream processing thanks to the Spark framework.

In [47], the authors compare Apache Kafka and RabbitMQ using the fundamental properties of pub/sub systems, and they also engage in a qualitative and empirical evaluation of the qualities that are shared by the two systems. They also emphasize the unique characteristics that each of these systems possesses. They strive to lead the reader through a determination table to select the appropriate architecture for his or her specific set of needs after listing a selection of use cases that are best suited for RabbitMQ or Kafka.

The three most network-intensive datapaths (record output, record duplication, and record consumption) are accelerated using remote direct memory access by the Apache Kafka extension KafkaDirect, which the authors of [48] present. They investigate design options, such as which remote direct memory access procedures to employ to fully utilise offloaded communication. They use one-sided remote direct memory access requests in their suggested architecture to achieve real zero-copy communication without using intermediary buffers in Kafka servers, resulting in low delay and high throughput communication.

In order to decrease the latency of content-based data dissemination, the authors of [49], suggest a brand-new form of topic in Kafka dubbed the fat topic. The basic idea behind improving forwarding performance with fat topics is that after matching an event with a set of subscriptions, the event can be stored together with the match list in a topic, instead of forwarding the event from one topic to many topics. Additionally, they alter the Kafka code to introduce consumer and producer APIs for fat topic access. To assess the effectiveness of fat topics, they ran comprehensive tests. The experiment's findings indicate that when compared to the initial Kafka topic, the fat topic can reduce the latency of content-based event dissemination by about 3.7 times.

The authors of [50] suggest using stream computing to streamline the genome resequencing workflow, enhancing its efficiency and fault-tolerance. In order to provide simple composability and inclusion into the pre-existing YARN-based pipelines, they divide the first stages of the genomic processing into two discrete and specialized modules (preprocessing and alignment). We then loosely compose these modules via communication over Kafka streams. The suggested solution is then empirically verified using actual data, and it is demonstrated that it scales approximately linearly.

TABLE 3. Cyber-physical convergence.

Article	How	Why
Internet of Things		
[51]	Robot Operating System	Robotic integration
[52]	Integrated streaming	Roboticised microscopy
[53]		Medical data analysis
[54]	Distributed messaging	Image recognition
[55]	Data analysis throughput	Video analysis
[56]	Micro-workflows	Digital twin
Vehicles and mobility		
[57]	MQTT integration	C-ITS messages
[58]	Video streams	Vehicle tracking
[59]	Architectural design	Autonomous vehicles
[60]		Mobility simulation
Environmental use cases		
[61]	Heterogeneous streaming	Environmental data analytics
[62]	Homogeneous streaming	Aquaculture monitoring
[63]		CO2 monitoring
[64]	LSTM neurons	Weather forecast
[65]	Stream storage	Meteo sensor data storage
[66]	Distributed cluster processing	Seismic waveform data

VII. CYBER-PHYSICAL CONVERGENCE

The convergence of physical systems with data-driven applications (Table 3) is an emerging trend that is rapidly gaining traction in diverse fields such as healthcare, energy, transportation, and environmental monitoring. This section of the paper delves into the role of Apache Kafka in the integration of physical systems with data-driven applications, with a particular emphasis on cyber-physical convergence. We examine how Kafka can be used to enable real-time communication and coordination between physical systems and data-driven applications, facilitating the development of intelligent systems that can learn from and respond to real-world events in a timely and efficient manner. The section is divided into three subsections, each of which focuses on a specific application domain. The first subsection explores how Kafka can be used to manage and process data generated by sensors and other IoT devices, such as robots or image/video/VR-enabled devices. The second subsection, examines how Kafka can be used to enable real-time communication and coordination between vehicles and other elements of the transportation infrastructure. The third subsection explores how Kafka can be used to manage and process data generated and collected for environmental applications, in which Apache Kafka has been a mainstream option for data transfer, and examines how Kafka can be used to enable real-time monitoring and analysis of environmental data, facilitating the development of intelligent systems for environmental monitoring and management. Overall, this section highlights the important role that Apache Kafka can play in enabling cyber-physical convergence across a range of application domains.

A. INTERNET OF THINGS

As Internet of Things (IoT) applications become increasingly popular, there are emerging challenges related to data ingestion, processing, storage, and analysis. Traditional data management approaches are not well-suited to handle the scale, velocity, and variety of IoT data, as well as the introduced mobility constraints which emerge from the employment of networked robotic elements. In addition, there are security and privacy concerns associated with IoT devices, as they often collect sensitive image and video data from individuals and organizations. Apache Kafka has emerged as a powerful tool for managing IoT data, enabling reliable and scalable data processing pipelines. However, there are still many challenges related to data transfer performance and interoperability that need to be addressed.

The authors of [51] contend that it is crucial to take into account how robotic platforms, such the Robot Operating System, deal with integrating with streaming systems that control the orders that are sent to smart warehouses. Streaming platform Kafka, which is widely used in e-commerce systems, is presented in this study as a straightforward method of integrating Robot Operating System. To connect these two systems, they create a bridge code, which they verify using three realistic simulation situations. As they demonstrate, a greater degree of dependability may be attained by utilizing the QoS profiles offered by the Robot Operating System data distribution service.

In [52], the authors, motivated by the development of near real-time image processing pipelines for roboticised microscopy, evaluate the suitability of Apache Spark for streams more typical of scientific computing applications, those with large message sizes, and heavy per-message CPU load, under typical stream integrations. For comparison, they benchmark a P2P stream processing framework, HarmonicIO, developed in-house. The data are preloaded into Kafka and they investigate ingress bottlenecks by writing and reading data through Kafka during the benchmarking, to get a full measurement of sustained throughput. The study reveals a complex interplay of performance trade-offs, revealing the boundaries of good performance for each framework and integration over a wide domain of application loads.

In [53], the authors analyse and calculate the medical data of an intelligent telemedicine Internet cloud computing "The Smart Hospital with the Best Doctors" platform in China, by using Spring integration technology, and the monitoring of diseases and deaths is realised on Amazon Managed Streaming with Apache Kafka.

According to the data broadcast characteristics from sensors, the authors of [54] suggest a building plan for a highly effective distributed stream processing infrastructure that enables scalable processing of moving image recognition jobs. They use Ray and Apache Kafka to create a prototype of the proposed distributed stream processing infrastructure and assess its performance. The outcomes of the experiments show how extremely scalable the suggested distributed stream processing infrastructure is.

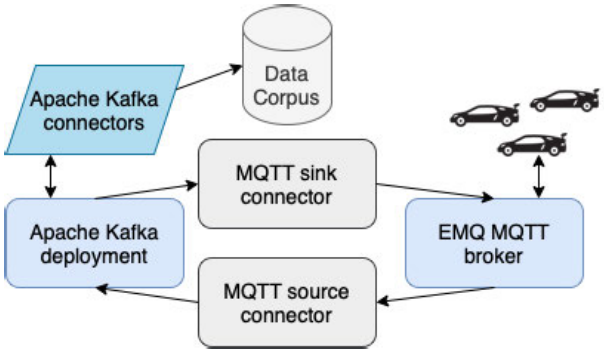

FIGURE 6. Cooperative intelligent transportation systems conceptual architecture based on [57] Apache Kafka implementation in the E2F2C continuum.

The authors of [55] provide a system for video analysis that gathers movies from various cameras and analyzes them with Apache Kafka and Apache Spark Streaming. They start by looking at Apache Kafka's data transmission performance and efficient cluster design and parameter settings. The throughput of the data analysis is then measured after applying this configuration to the suggested framework.

In [56], the authors, to organise a flexible cloud computing support for the digital twin execution, propose a concept of micro-workflows that combines the power of scientific workflows, the flexibility of containers technology, and robustness of the distributed streaming approach.

B. VEHICLES AND MOBILITY

The topic of vehicles and mobility has gained increasing attention in recent years as the transportation industry is undergoing a rapid transformation with the emergence of new technologies. With the growth of connected cars, data is being generated at an unprecedented rate, presenting an opportunity to utilize this data for various applications, such as vehicle tracking, real-time traffic management, and personalized services. Apache Kafka has become a key technology in this domain, enabling the collection and processing of data from multiple sources in real-time. As the industry continues to evolve, it is essential to develop robust and scalable solutions that can manage the vast amounts of data generated by connected vehicles and provide value-added services to customers while ensuring data security and privacy.

The authors of In [57] describe a feasibility study on the methods for receiving, processing, and distributing signals from cooperative intelligent transportation systems. Their strategy involves connecting several message types to Apache Kafka via Message Queuing Telemetry Transport (MQTT), with fault tolerance, horizontal scalability, and minimal latency. By utilizing the Kafka Connector API, they have created both a Kafka-MQTT sink connector and a MQTT-Kafka source connector, as displayed in Fig. 6. These make it simple and configurable to map topics from mobile devices and roadside equipment to a central application as well as the other way around. By successfully sourcing and sinking CAM messages with little latency, the trials with this bridging technique, carried out on a basic desktop computer

with a single instance Kafka configuration, demonstrate its viability. In [58], the authors propose an approach to track a specific vehicle over the video streams published by the collaborating traffic surveillance cameras. They report how the number of partitions, replications, and brokers has an effect on producer-send packet throughput (mb/sec): producer throughput varies depending on the number of brokers, partitions of topics, and replications. In [59], the authors explore the existing and emerging platforms for mobile edge computing and human-centric applications, and propose a suitable Kafka-based architecture that can be used in the context of autonomous vehicle systems. The proposed architecture will support scalable communication among sensing devices and edge/cloud computing platforms, as well as orchestrate services for computing, storage, and learning. In [60], the authors design a simulation platform enabling evaluations of future mobility scenarios, based on an Apache Kafka architecture.

C. ENVIRONMENTAL USE CASES

Apache Kafka is also finding increasing use in environmental monitoring and management applications. Real-time data acquisition from sensors and systems installed in natural resources or environmental systems can be efficiently collected and analyzed through Kafka, allowing stakeholders to gain valuable insights into the health and functioning of environmental systems. For instance, in the context of water resource management, sensors can be used to monitor water quality and levels, and the data can be streamed in real-time to Kafka for analysis and decision-making. Similarly, in the context of weather and climate monitoring, environmental data from remote sensors and stations can be streamed through Kafka to enable real-time analysis and forecasting of weather patterns and climate trends. The use of Kafka in environmental applications can greatly improve our understanding and management of natural resources, leading to more effective and sustainable use of our planet's resources.

The authors of [61] propose an event-stream processing engine for the environmental monitoring domain (ESTemd) as a distributed framework for the use of stream processing on heterogeneous environmental data. Their work in this field exemplifies the value of big data approaches in early warning, forecasting, and environmental decision support systems. The suggested framework uses a publish/subscribe mechanism via a single data pipeline with the deployment of Apache Kafka for real-time analytics to handle the difficulties of data heterogeneity from disparate systems and real-time processing of enormous environmental datasets.

The authors of the [61] propose a distributed framework for the use of stream processing on heterogeneous environmental data that addresses the problems of data heterogeneity from heterogeneous systems and provides real-time processing of large environmental datasets through a publish/subscribe method via a unified data pipeline with the use of Apache Kafka for real-time analytics.

The aquaculture industry needs to monitor the water quality as well as other water and weather parameters for large and diverse farm types. An aquaculture monitoring system targets at continuously online monitoring of water quality sensors. The authors of [62] propose and develop an aquaculture monitoring system using Flink, MongoDB, and Kafka. Among these, Flink offers a platform for processing sensor data with high throughput and low latency. The effectiveness of a few typical operations between HBase and their solution is examined and contrasted using a real aquaculture dataset. The testing findings demonstrate that their solution's efficiency is significantly higher than that of HBase, which offered a workable option for the storing and processing of aquaculture sensor data.

In [63], the authors developed a wireless sensor networking system for CO2 monitoring using Kafka and Impala to distribute a huge amount of data. Sensor nodes gather data and accumulated in temporary storage then streamed via Kafka platform to be stored into Impala database. System tested with data gathered from the custom made sensor nodes and give a good performance.

Weather models are simulations of the future state of the atmosphere out through time. Millions of observations are used as initial conditions in trillions of calculations, producing a three dimensional picture of what the atmosphere might look like at some time in the future. In [64], the authors create a weather forecast model that automatically picks up new information from the daily input of weather data from a third-party API source. The weather feed is streamed into the forecast model using Kafka components and is sourced from openweathermap, an internet service that offers weather data. The forecast model's LSTM neural network is built to continually learn from forecasts and conduct real analysis. The model may be built so that it can be used in very large applications that can process huge amounts of stored or streaming data.

A TimescaleDB and Kafka-based data storage system for meteorological sensor data is suggested in [65]. This system used Kafka to collect and send meteorological sensor data, which was then forwarded to TimescaleDB for archiving and analysis. It compared the solution against existing NoSQL stores including Redis, MongoDB, Cassandra, HBase, and Riak TS using a dataset of simulated weather sensor data. The experimental findings demonstrate that the suggested storage technique is preferable for both storing and processing large amounts of data from meteorological sensors.

In [66], the authors design a distributed cluster processing model based on Apache Kafka data queues, to optimise the inbound efficiency of seismic waveform data.

VIII. SECURITY

In recent years, data breaches and cyber attacks have become increasingly common, highlighting the need for robust security measures to protect sensitive data. With the rise of networked data streaming and the widespread use of Apache Kafka, it is crucial to ensure that these systems are secure and able to withstand potential threats. This has led

TABLE 4. Security services for Apache Kafka.

Article	Security issue addressed	Service
[67]	DDoS classification	Highly scalable ML
[68]	DDoS filtering/detection	Programmable network telemetry
[69]	Sensitive data processing	Encryption-at-rest
[70]	User privacy leakage risk	Secure data transmission
[71]	Encrypting/decrypting compliance	Multi-database coherency
[72]	Data storing and processing	Public-key cryptography

to an increased focus on security for Apache Kafka, with numerous studies and works exploring the various challenges and potential solutions for securing Kafka-based systems (Table 4). In this section, we survey the literature on security for Apache Kafka, including studies on data transmission, encryption, and other security-related topics. We discuss the various security issues associated with Apache Kafka and review the different services that have been proposed to address the corresponding challenges.

In [67], the authors introduce a distributed method based on Apache Kafka which targets at classifying distributed denial of service (DDoS) attacks. To begin with, they gather data from Hadoop and create distributed classification models on the Hadoop network using highly scalable ML methods. Second, they use the Kafka Stream cluster to classify inbound network data into nine categories in real time. Furthermore, using a fresh collection of instances, this distributed classification method saves highly discriminative features with expected results in Hadoop.

In [68], the authors introduce a hardware and software solution that is configurable, extensible, and expressive and that generates and analyses per-packet telemetry data with nanosecond-accurate timing. They emphasise their design, which they believe to be the key performance factor that enables secure handling of telemetry packets at the scale of millions per second, more than enough to manage high loads of traffic. Additionally, they demonstrate real-time stream processing apps for the system that feature sophisticated filtration, aggregation, and windowing features. Their use-cases demonstrate the versatility in supporting a range of complex performance tracking, troubleshooting, and security duties. Specifically, they focus on TCP-based SYN flood attacks, which are hard to be filtered by routers when the source IP address is spoofed.

According to the authors of [69], Kafka must adhere to the same security and compliance standards as traditional data storage systems like relational databases for cloud providers and businesses. One crucial criterion that Kafka does not presently provide is encryption-at-rest. They initially analyze several Kafka encryption implementation strategies before outlining the first full solution for implementing encryption-at-rest at scale at the level of a Kafka topic. They use a functioning implementation to illustrate the difficulties in implementing encryption policy, key distribution, key rotation, and data re-encryption in Kafka.

In [70], the authors point out that, in contrast to typical cloud-based access control designs, Kafka service providers frequently must construct their systems on high-performance cloud platforms owned by other businesses. Since the cloud platform is owned by a third party, it is not always dependable. They demonstrate the paradoxical fact that Kafka's data is kept in the cloud in unencrypted form, and as a result, they showed a significant risk of user privacy being compromised. To prevent the data from being exposed in Kafka, they suggest a secure fine-grained data transmission method which, in addition to being more secure than Kafka's built-in security system, can successfully thwart the theft of unencrypted data by third-party clouds.

The authors of [71] outline their initial effort to solve some of the practical issues of employing Kafka as a single data storage within a business. They specifically discuss basic methods for ensuring consistency and coherence of data delivered via Kafka from various database tables as well as how to manage compliance by encrypting and decrypting data at the Kafka producers and consumers.

The authors of [72] suggest a Hadoop ecosystem to support a number of features in the industrial sector. Public-key cryptography, which uses both public and private keys, is used as a security technique. Furthermore, the private key is kept in the Kafka consumer, while the public key is located in the Kafka producer. The performance and accuracy of data storage, processing, and security in the industrial environment will improve with the integration of the aforementioned technologies.

IX. OPEN CHALLENGES

As Apache Kafka continues to gain popularity as a reliable and scalable data streaming platform, it faces new challenges that should be addressed to continue to meet the needs of modern data-driven applications. In this section, we present some of the open challenges that we identified in the context of our investigation for this paper, related to scaling Kafka for massive data streams, real-time analytics and processing, integrating Kafka with cyber-physical systems, and ensuring security in Kafka deployments (Fig. 7).

A. SCALING APACHE KAFKA FOR MASSIVE DATA STREAMS

Following the discussion of section V on networked infrastructures and section VI on data handling and processing, we can conclude that one of the main challenges in Apache Kafka is scaling it to handle massive data streams. As the volume of data generated in many modern applications continues to grow exponentially, there is a need for scalable and reliable messaging systems that can handle these data streams. Although Apache Kafka was designed with scalability in mind, scaling it to handle very large data streams can still be a challenge. One approach to addressing this challenge is to distribute the Kafka brokers across multiple nodes in a cluster, thus enabling horizontal scaling. However, this approach can also introduce new challenges related to load balancing, data partitioning, and fault tolerance. To address such challenges, research can be conducted on optimising the performance and scalability of Apache

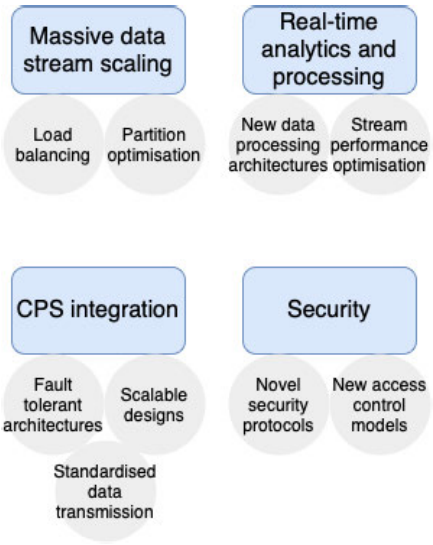

FIGURE 7. Identified Apache Kafka challenges and respective envisioned approaches.

Kafka in distributed environments; for example, with new load-balancing algorithms that can distribute the data traffic across Kafka brokers more evenly, thus reducing the risk of bottlenecks and improving overall system performance [68], or with optimising the partitioning of data across Kafka topics to dynamically adjust the partitioning strategy based on the workload and data characteristics [31].

New research suggestions:

1) *Adaptive Load Balancing Mechanisms*: Investigate and design novel adaptive load-balancing algorithms for Apache Kafka that can intelligently distribute data traffic across brokers in a dynamic manner. These mechanisms should be capable of detecting variations in data stream volumes and adaptively allocate resources to brokers to prevent bottlenecks and ensure efficient utilization of cluster resources.
2) *Enhanced Data Partitioning Strategies*: Explore advanced data partitioning strategies that optimize data distribution across Kafka topics based on workload characteristics. Research could focus on developing partitioning algorithms that can dynamically adjust partition assignments to better align with data flow patterns, ensuring balanced distribution and reducing latency in high-throughput scenarios.
3) *Fault-Tolerant Horizontal Scaling*: Investigate fault-tolerant approaches for horizontal scaling of Apache Kafka clusters. This research could explore mechanisms to handle node failures gracefully, redistribute partitions, and maintain data consistency across nodes. The goal is to enhance the system's ability to recover from failures seamlessly, ensuring high availability and reliability.
4) *Stream Compression Techniques*: Explore the use of innovative stream compression techniques to optimize the storage and transfer of data streams within Apache Kafka. Research could focus on developing efficient compression algorithms that minimize storage overhead and reduce network bandwidth utilization, especially for large-scale data streams.
5) *Scaling for Multi-Tenancy*: Examine how Apache Kafka can efficiently handle multi-tenancy scenarios, where multiple applications or tenants share the same Kafka cluster. Investigate resource isolation, performance isolation, and security mechanisms to ensure smooth coexistence of diverse workloads within a shared infrastructure.

B. REAL-TIME ANALYTICS AND PROCESSING WITH APACHE KAFKA

Another challenge in Apache Kafka is leveraging it for real-time analytics and processing of streaming data. In many applications, the ability to process data in real-time can provide significant advantages, such as enabling proactive decision-making, reducing response times, and improving overall system efficiency. Apache Kafka provides a powerful platform for real-time data processing, but it also introduces new challenges related to the complexity of stream processing, the need for high throughput and low latency, and the requirements for managing stateful processing. To address such challenges, research could focus on developing new approaches and techniques for stream processing with Apache Kafka; for example, with new data processing architectures that use Apache Kafka as a central component for data ingestion and distribution, while also integrating other technologies such as Apache Spark or Flink for stream processing and analytics [73], or with optimising the performance of Kafka streams by reducing processing overhead, improving fault tolerance, and enabling stateful processing in a distributed environment [74].

New research suggestions:

6) *Integrated Stream Processing Architectures*: Investigate and propose innovative integrated stream processing architectures that leverage Apache Kafka as a central component for data ingestion and distribution. This research could explore the seamless integration of other stream processing technologies, such as Apache Spark or Apache Flink, to enhance real-time analytics capabilities. The goal is to design architectures that optimize data flow, enable efficient data processing, and provide seamless interoperability between different processing components.
7) *Performance Optimization for Kafka Streams*: Focus on optimizing the performance of Kafka streams in distributed environments. This research could explore methods to reduce processing overhead, improve fault tolerance mechanisms, and enhance the scalability of Kafka streams for handling large-scale data processing tasks. Investigate techniques that enable stateful processing while efficiently managing the associated complexities in a distributed setting.
8) *Efficient State Management for Stream Processing*: Develop novel approaches for efficient stateful

processing in Apache Kafka-based stream processing. This research could explore advanced state management techniques that optimize memory usage, reduce latency, and ensure fault tolerance in managing stateful processing applications. Investigate methods to efficiently distribute and synchronize state across partitions and nodes, while maintaining data consistency and minimizing overhead.

C. INTEGRATING APACHE KAFKA WITH CYBER-PHYSICAL SYSTEMS

Apache Kafka has the potential to be used as a core data streaming platform for cyber-physical systems, allowing for real-time communication between physical devices and data-driven applications. However, there are several challenges associated with tightly integrating Apache Kafka with cyber-physical systems. One challenge is ensuring the reliability and availability of the system. In a cyber-physical environment, system failures can have serious consequences, such as equipment damage, production delays, and even safety hazards. Therefore, it is essential to design the system architecture in such a way that it is fault-tolerant and can recover quickly from failures [75]. This leads to the challenge of ensuring the scalability of the cyber-physical architecture. As the number of physical devices and data-driven applications in a cyber-physical environment grows, the cyber-physical architecture must be able to scale to accommodate the increased traffic and data volume [76]. Finally, there is a challenge related to the interoperability of different system devices and applications. Different devices and applications may use different protocols or formats for data transmission, which can lead to compatibility issues when integrating them with Apache Kafka. This challenge can be addressed by using standard protocols and formats for data transmission in cyber-physical environments [77]. Overall, the integration of Apache Kafka with cyber-physical systems has the potential to enable new applications and use cases in a wide range of industries. However, addressing these challenges is critical to ensure the reliability, interoperability, and scalability of cyber-physical architectures that rely on Apache Kafka.

New research suggestions:

9) *Fault-Tolerant Architecture Design*: Investigate novel fault-tolerant architecture designs that ensure the reliability and availability of cyber-physical systems built on Apache Kafka. This research could focus on developing redundancy mechanisms, intelligent failover strategies, and effective error handling to minimize downtime and prevent system failures. The goal is to enhance the system's ability to recover swiftly and maintain uninterrupted operations, even in the face of critical failures.
10) *Scalability Solutions for High-Traffic Environments*: Explore innovative scalability solutions tailored for high-traffic cyber-physical environments. This research could include optimizing Kafka's performance for handling large data volumes, implementing efficient data partitioning schemes, and devising load balancing algorithms to distribute the increased traffic evenly across brokers. The aim is to ensure that the system can effectively handle the growing data demands as the number of devices and applications increases.
11) *Standardized Data Transmission Protocols*: Address the challenge of interoperability by proposing standardized data transmission protocols and formats for cyber-physical environments. Investigate the adaptation of widely accepted industry standards to enable seamless communication between different devices and applications. The research should focus on developing protocols that efficiently handle data conversion and enable secure, reliable data exchange, ensuring compatibility and reducing integration complexities.

D. SECURITY CONSIDERATIONS IN APACHE KAFKA DEPLOYMENTS

As with any distributed system, security is a critical concern when deploying Apache Kafka in enterprise environments. As discussed in section VIII, Apache Kafka introduces new security challenges related to data privacy, access control, authentication, and authorisation. Ensuring the security of Apache Kafka requires careful attention to these issues, as well as the configuration of the underlying infrastructure and network. To address these challenges, research will need to focus on developing new security models and protocols for Apache Kafka deployments; for example, with new encryption techniques for securing data transmission in Kafka clusters, such as using TLS or AES algorithms [78], or with developing new access control models for Kafka topics, such as using role-based access control or attribute-based access control to manage user privileges [79], to help ensure the security of Apache Kafka deployments and protect against potential security threats.

New research suggestions:

12) *Enhanced Data Transmission Security*: Investigate and develop advanced encryption techniques to bolster the security of data transmission within Apache Kafka clusters. Research could focus on leveraging robust encryption algorithms, such as TLS or AES, to safeguard data privacy and confidentiality during data transfer. By implementing robust encryption methods, researchers and practicioners can fortify the security posture of Apache Kafka deployments, protecting sensitive information from potential unauthorized access.
13) *Advanced Access Control Models*: Explore novel access control models tailored for Apache Kafka topics to strengthen access management and ensure data integrity. Research could focus on designing role-based access control (RBAC) or attribute-based access control (ABAC) mechanisms to grant user privileges based on specific attributes or roles. By implementing fine-grained access controls, application owners can enforce

strict data access policies, mitigating the risks of unauthorized data access and manipulation.

14) *Security Auditing and Threat Detection*: Develop effective security auditing and threat detection mechanisms for Apache Kafka deployments. This research could explore methods to monitor and log security-related events within Kafka clusters, enabling real-time detection of potential security breaches or suspicious activities. By incorporating robust threat detection mechanisms, we can proactively respond to security incidents and bolster the overall resilience of Apache Kafka environments.

In conclusion, addressing the challenges of scaling Apache Kafka for massive data streams, enabling real-time analytics and processing, integrating with cyber-physical systems, and ensuring robust security in deployments, presents a rich landscape of new research opportunities. By exploring and advancing these research suggestions, the potential of Apache Kafka as a core data streaming platform can be fully realized, offering transformative benefits across diverse industries and enhancing its position as a leading solution for managing networked data streaming applications.

X. CONCLUSION

The growing popularity of Apache Kafka as a solution for managing networked data streaming in various domains has prompted extensive research in the field. In this paper, we have conducted a comprehensive survey of the research literature on networked data streaming with Apache Kafka and identified several key findings and open research challenges that pave the way for future advancements. To provide a structured analysis, we classified the research literature into representative macro areas, namely algorithms, networks, data, cyber-physical systems, and security. Within these areas, we explored different aspects and optimization techniques related to Apache Kafka. In the realm of algorithmic foundations, we delved into the combinatorial aspects and reliability engineering of Apache Kafka. This analysis shed light on the underlying principles and design considerations that enable Kafka to handle high-volume event data efficiently. Furthermore, we examined the networked infrastructure optimization aspects, emphasizing how Apache Kafka enhances the performance and scalability of distributed systems. We discussed the various techniques and architectural designs that leverage Kafka to optimize networked infrastructure. Data handling and processing emerged as another crucial area, with Apache Kafka proving its effectiveness in real-time data streaming, message queuing, and machine learning-based computing. We explored the different applications and use cases where Kafka enables seamless and efficient data handling and processing. The emerging trend of cyber-physical convergence was also explored, highlighting the role of Apache Kafka in integrating physical systems with data-driven applications. We examined specific use cases in the Internet of Things, vehicles, mobility, and environmental domains, illustrating how Kafka facilitates the convergence of physical and digital worlds. Security considerations were not overlooked, as we delved into the measures and best practices for ensuring the secure deployment of Apache Kafka in enterprise environments. Lastly, we identified several open research challenges that require further exploration in the field of networked data streaming with Apache Kafka. These challenges include scaling Kafka for massive data streams, advancing real-time analytics and processing capabilities, integrating Kafka with cyber-physical systems more seamlessly, and enhancing security mechanisms for Kafka deployments.

REFERENCES

[1] D. Bajovic et al., "MARVEL: Multimodal extreme scale data analytics for smart cities environments," in *Proc. Int. Balkan Conf. Commun. Netw. (BalkanCom)*, Sep. 2021, pp. 143–147.

[2] T. P. Raptis, A. Passarella, and M. Conti, "Distributed data access in industrial edge networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 915–927, May 2020.

[3] C. M. Angelopoulos, G. Filios, S. Nikoletseas, and T. P. Raptis, "Keeping data at the edge of smart irrigation networks: A case study in strawberry greenhouses," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 107039.

[4] T. Raptis, A. Passarella, and M. Conti, "Performance analysis of latency-aware data management in industrial IoT networks," *Sensors*, vol. 18, no. 8, p. 2611, Aug. 2018.

[5] M. J. Sax, *Apache Kafka*. Cham, Switzerland: Springer, 2018, pp. 1–8, doi: 10.1007/978-3-319-63962-8_196-1.

[6] G. Wang, J. Koshy, S. Subramanian, K. Paramasivam, M. Zadeh, N. Narkhede, J. Rao, J. Kreps, and J. Stein, "Building a replicated logging system with Apache Kafka," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1654–1655, Aug. 2015.

[7] S. Vyas, R. K. Tyagi, C. Jain, and S. Sahu, "Literature review: A comparative study of real time streaming technologies and Apache Kafka," in *Proc. 4th Int. Conf. Comput. Intell. Commun. Technol. (CCICT)*, Jul. 2021, pp. 146–153.

[8] S. Kul and A. Sayar, "A survey of publish/subscribe middleware systems for microservice communication," in *Proc. 5th Int. Symp. Multidisciplinary Stud. Innov. Technol. (ISMSIT)*, Oct. 2021, pp. 781–785.

[9] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A survey of distributed data stream processing frameworks," *IEEE Access*, vol. 7, pp. 154300–154316, 2019.

[10] A. H. Ali and M. Z. Abdullah, "Recent trends in distributed online stream processing platform for big data: Survey," in *Proc. 1st Annu. Int. Conf. Inf. Sci. (AiCIS)*, Nov. 2018, pp. 140–145.

[11] G. Hesse and M. Lorenz, "Conceptual survey on data stream processing systems," in *Proc. IEEE 21st Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2015, pp. 797–802.

[12] J. Xu, J. Yin, H. Zhu, and L. Xiao, "Modeling and verifying producer-consumer communication in Kafka using CSP," in *Proc. 7th Conf. Eng. Comput. Based Syst. (ECBS)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–10.

[13] H. Wu, Z. Shang, and K. Wolter, "Performance prediction for the Apache Kafka messaging system," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 154–161.

[14] T. P. Raptis and A. Passarella, "On efficiently partitioning a topic in Apache Kafka," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Jul. 2022, pp. 1–8.

[15] D. Landau, X. Andrade, and J. G. Barbosa, "Kafka consumer group autoscaler," 2022, *arXiv:2206.11170*.

[16] L. Xu, X. Ma, and L. Xu, "A novel adaptive tuning mechanism for Kafka-based ordering service," in *Web Information Systems and Applications*, W. Ni, X. Wang, W. Song, and Y. Li, Eds. Cham, Switzerland: Springer, 2019, pp. 119–125.

[17] J. Bang, S. Son, H. Kim, Y.-S. Moon, and M.-J. Choi, "Design and implementation of a load shedding engine for solving starvation problems in Apache Kafka," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–4.

[18] M. Gütlein and A. Djanatliev, "Modeling and simulation as a service using Apache Kafka," in *Proc. 10th Int. Conf. Simulation Modeling Methodol., Technol. Appl. (SIMULTECH)*, 2020, pp. 171–180.

[19] T. Aung, H. Y. Min, and A. H. Maw, "Enhancement of fault tolerance in Kafka pipeline architecture," in *Proc. 11th Int. Conf. Adv. Inf. Technol. (IAIT)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–8.

[20] L.-P. Chen, L.-F. Yei, and Y.-R. Chen, "An efficient disaster recovery mechanism for multi-region Apache Kafka clusters," in *Innovative Mobile and Internet Services in Ubiquitous Computing*, L. Barolli, Ed. Cham, Switzerland: Springer, 2022, pp. 297–306.

[21] H. Wu, Z. Shang, G. Peng, and K. Wolter, "A reactive batching strategy of Apache Kafka for reliable stream processing in real-time," in *Proc. IEEE 31st Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2020, pp. 207–217.

[22] H. Wu, "Research proposal: Reliability evaluation of the Apache Kafka streaming system," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2019, pp. 112–113.

[23] G. Hesse, C. Matthies, and M. Uflacker, "How fast can we insert? An empirical performance evaluation of Apache Kafka," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 641–648.

[24] H. Wu, Z. Shang, and K. Wolter, "Learning to reliably deliver streaming data with Apache Kafka," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2020, pp. 564–571.

[25] H. Wu, Z. Shang, and K. Wolter, "TRAK: A testing tool for studying the reliability of data delivery in Apache Kafka," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2019, pp. 394–397.

[26] M. H. Javed, X. Lu, and D. K. D. Panda, "Characterization of big data stream processing pipeline: A case study using Flink and Kafka," in *Proc. 4th IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol. (BDCAT)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1–10.

[27] E. Falk, V. K. Gurbani, and R. State, "Query-able Kafka: An agile data analytics pipeline for mobile wireless networks," *Proc. VLDB Endowment*, vol. 10, no. 12, pp. 1646–1657, Aug. 2017.

[28] R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, A. González-Muñiz, J. P. Fernández-Palacios, "Optical network telemetry with streaming mechanisms using transport API and Kafka," in *Proc. Eur. Conf. Opt. Commun. (ECOC)*, 2021, pp. 1–4.

[29] J.-H. Moon and Y.-T. Shine, "A study of distributed SDN controller based on Apache Kafka," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 44–47.

[30] M. Tsenos, N. Zacheilas, and V. Kalogeraki, "Dynamic rate control in the Kafka system," in *Proc. 24th Pan-Hellenic Conf. Informat. (PCI)*, Nov. 2020, pp. 96–98.

[31] T. P. Raptis, C. Cicconetti, M. Falelakis, G. Kalogiannis, T. Kanellos, and T. P. Lobo, "Engineering resource-efficient data management for smart cities with Apache Kafka," *Future Internet*, vol. 15, no. 2, p. 43, Jan. 2023, doi: 10.3390/fi15020043.

[32] Z. Farkas and R. Lovas, "Reference architecture for IoT platforms towards cloud continuum based on Apache Kafka and orchestration methods," in *Proc. 7th Int. Conf. Internet Things, Big Data Secur. (IoTBDS)*, 2022, pp. 205–214.

[33] C. N. Nguyen, J.-S. Kim, and S. Hwang, "KOHA: Building a Kafka-based distributed queue system on the fly in a Hadoop cluster," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FAS*W)*, Sep. 2016, pp. 48–53.

[34] M. Raza, J. Tahir, C. Doblander, R. Mayer, and H.-A. Jacobsen, "Benchmarking Apache Kafka under network faults," in *Proc. 22nd Int. Middleware Conf., Demos Posters (Middleware)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 5–7.

[35] P. Le Noac'h, A. Costan, and L. Bougé, "A performance evaluation of Apache Kafka in support of big data streaming applications," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4803–4806.

[36] H. Kim, J. Bang, S. Son, N. Joo, M.-J. Choi, and Y.-S. Moon, "Message latency-based load shedding mechanism in Apache Kafka," in *Euro-Par 2019: Parallel Processing Workshops*, U. Schwardmann, C. Boehme, D. B. Heras, V. Cardellini, E. Jeannot, A. Salis, C. Schifanella, R. R. Manumachu, D. Schwamborn, L. Ricci, O. Sangyoon, T. Gruber, L. Antonelli, and S. L. Scott, Eds. Cham, Switzerland: Springer, 2020, pp. 731–736.

[37] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21002995

[38] A. Carnero, C. Martín, D. R. Torres, D. Garrido, M. Díaz, and B. Rubio, "Managing and deploying distributed and deep neural models through Kafka-ML in the cloud-to-things continuum," *IEEE Access*, vol. 9, pp. 125478–125495, 2021.

[39] D. R. Torres, C. Martín, B. Rubio, and M. Díaz, "An open source framework based on Kafka-ML for distributed DNN inference over the cloud-to-things continuum," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102214. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138376212100151X

[40] A. K. Lakkad, R. D. Bhadaniya, V. N. Shah, and K. Lavanya, "Complex events processing on live news events using Apache Kafka and clustering techniques," *Int. J. Intell. Inf. Technol.*, vol. 17, no. 1, pp. 39–52, Jan. 2021, doi: 10.4018/IJIIT.2021010103.

[41] S. Bano, N. Tonellotto, P. Cassarà, and A. Gotta, "KafkaFed: Two-tier federated learning communication architecture for Internet of Vehicles," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Other Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 515–520.

[42] E. Alothali, H. Alashwal, M. Salih, and K. Hayawi, "Real time detection of social bots on Twitter using machine learning and Apache Kafka," in *Proc. 5th Cyber Secur. Netw. Conf. (CSNet)*, Oct. 2021, pp. 98–102.

[43] S. Langhi, R. Tommasini, and E. D. Valle, "Extending Kafka streams for complex event recognition," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 2190–2197.

[44] H. Jafarpour and R. Desai, "KSQL: Streaming SQL engine for Apache Kafka," in *Proc. 22nd Int. Conf. Extending Database Technol. (EDBT)*, Lisbon, Portugal, M. Herschel et al., Eds., Mar. 2019, pp. 524–533, doi: 10.5441/002/edbt.2019.48.

[45] C. Prabhu, R. V. Gandhi, A. K. Jain, V. S. Lalka, S. G. Thottempudi, and P. P. Rao, "A novel approach to extend KM models with object knowledge model (OKM) and Kafka for big data and semantic web with greater semantics," in *Complex, Intelligent, and Software Intensive Systems*, L. Barolli, F. K. Hussain, and M. Ikeda, Eds. Cham, Switzerland: Springer, 2020, pp. 544–554.

[46] X. Ren, O. Curé, H. Khrouf, and Z. Kazi-Aoul, "Apache spark and Apache Kafka at the rescue of distributed RDF stream processing engines," in *Proc. ISWC Posters Demonstrations Track Co-Located With 15th Int. Semantic Web Conf. (ISWC)*, in CEUR Workshop Proceedings, Kobe, Japan, vol. 1690, T. Kawamura and H. Paulheim, Eds., Oct. 2016, pp. 1–4. [Online]. Available: http://ceur-ws.org/Vol-1690/paper43.pdf

[47] P. Dobbelaere and K. S. Esmaili, "Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper," in *Proc. 11th ACM Int. Conf. Distrib. Event-Based Syst.*, 2017, pp. 227–238.

[48] K. Taranov, S. Byan, V. Marathe, and T. Hoefler, "KafkaDirect: Zero-copy data access for Apache Kafka over RDMA networks," in *Proc. Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 2191–2204, doi: 10.1145/3514221.3526056.

[49] S. Qian, J. Xu, J. Cao, G. Xue, J. Li, and W. Zhang, "Fat topic: Improving latency in content-based publish/subscribe systems on Apache Kafka," in *Wireless Algorithms, Systems, and Applications*, Z. Liu, F. Wu, and S. K. Das, Eds. Cham, Switzerland: Springer, 2021, pp. 547–558.

[50] F. Versaci, L. Pireddu, and G. Zanetti, "Kafka interfaces for composable streaming genomics pipelines," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat. (BHI)*, Mar. 2018, pp. 259–262.

[51] L. L. Lourenço, G. Oliveira, P. D. M. Plentz, and J. Röning, "Achieving reliable communication between Kafka and ROS through bridge codes," in *Proc. 20th Int. Conf. Adv. Robot. (ICAR)*, Dec. 2021, pp. 324–329.

[52] B. Blamey, A. Hellander, and S. Toor, "Apache spark streaming, Kafka and HarmonicIO: A performance benchmark and architecture comparison for enterprise and scientific computing," in *Benchmarking, Measuring, and Optimizing*, W. Gao, J. Zhan, G. Fox, X. Lu, and D. Stanzione, Eds. Cham, Switzerland: Springer, 2020, pp. 335–347.

[53] J. Peng and X. Liu, "Disease and death monitoring on Amazon managed streaming for Apache Kafka," in *Proc. 2nd Int. Symp. Artif. Intell. Med. Sci. (ISAIMS)*. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 24–27, doi: 10.1145/3500931.3500937.

[54] K. Kato, A. Takefusa, H. Nakada, and M. Oguchi, "Construction scheme of a scalable distributed stream processing infrastructure using ray and Apache Kafka," in *Proc. 34th Int. Conf. Comput. Their Appl.*, in EPiC Series in Computing, vol. 58, G. Lee and Y. Jin, Eds., 2019, pp. 368–377. [Online]. Available: https://easychair.org/publications/paper/LFCL

[55] A. Ichinose, A. Takefusa, H. Nakada, and M. Oguchi, "A study of a video analysis framework using Kafka and spark streaming," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 2396–2401.
[56] G. Radchenko, A. B. A. Alaasam, and A. Tchernykh, "Micro-workflows: Kafka and Kepler fusion to support digital twins of industrial processes," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, Dec. 2018, pp. 83–88.
[57] Å. Hugo, B. Morin, and K. Svantorp, "Bridging MQTT and Kafka to support C-ITS: A feasibility study," in *Proc. 21st IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2020, pp. 371–376.
[58] S. Kul, I. Tashiev, A. Sentas, and A. Sayar, "Event-based microservices with Apache Kafka streams: A real-time vehicle detection system based on type, color, and speed attributes," *IEEE Access*, vol. 9, pp. 83137–83148, 2021.
[59] S. Bano, E. Carlini, P. Cassara, M. Coppola, P. Dazzi, and A. Gotta, "A novel approach to distributed model aggregation using Apache Kafka," in *Proc. 2nd Workshop Flexible Resource Appl. Manag. Edge (FRAME)*. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 33–36, doi: 10.1145/3526059.3533621.
[60] M. Gütlein and A. Djanatliev, "On-demand simulation of future mobility based on Apache Kafka," in *Simulation and Modeling Methodologies, Technologies and Applications*, M. S. Obaidat, T. Oren, and F. D. Rango, Eds. Cham, Switzerland: Springer, 2022, pp. 18–41.
[61] A. Akanbi, "ESTemd: A distributed processing framework for environmental monitoring based on Apache Kafka streaming engine," in *Proc. 4th Int. Conf. Big Data Research (ICBDR)*, Nov. 2020, pp. 18–25, doi: 10.1145%2F3445945.3445949.
[62] Y. Lou, L. Chen, F. Ye, Y. Chen, and Z. Liu, "Research and implementation of an aquaculture monitoring system based on Flink, MongoDB and Kafka," in *Computational Science—ICCS 2019*, J. M. F. Rodrigues, P. J. S. Cardoso, J. Monteiro, R. Lam, V. V. Krzhizhanovskaya, M. H. Lees, J. J. Dongarra, and P. M. A. Sloot, Eds. Cham, Switzerland: Springer, 2019, pp. 648–657.
[63] R. Wiska, N. Habibie, A. Wibisono, W. S. Nugroho, and P. Mursanto, "Big sensor-generated data streaming using Kafka and impala for data storage in wireless sensor network for CO_2 monitoring," in *Proc. Int. Workshop Big Data Inf. Secur. (IWBIS)*, Oct. 2016, pp. 97–102.
[64] 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.
[65] L. Shen, Y. Lou, Y. Chen, M. Lu, and F. Ye, "Meteorological sensor data storage mechanism based on TimescaleDB and Kafka," in *Data Science*, X. Cheng, W. Jing, X. Song, and Z. Lu, Eds. Singapore: Springer, 2019, pp. 137–147.
[66] X.-C. Chai, Q.-L. Wang, W.-S. Chen, W.-Q. Wang, D.-N. Wang, and Y. Li, "Research on a distributed processing model based on Kafka for large-scale seismic waveform data," *IEEE Access*, vol. 8, pp. 39971–39981, 2020.
[67] N. V. Patil, C. R. Krishna, and K. Kumar, "KS-DDoS: Kafka streams-based classification approach for DDoS attacks," *J. Supercomput.*, vol. 78, no. 6, pp. 8946–8976, Apr. 2022, doi: 10.1007/s11227-021-04241-1.
[68] Z. Liu, B. Mah, Y. Kumar, C. Guok, and R. Cziva, "Programmable per-packet network telemetry: From wire to Kafka at scale," in *Proc. Syst. Netw. Telemetry Anal. (SNTA)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 33–36, doi: 10.1145/3452411.3464443.
[69] C. Giblin, S. Rooney, P. Vetsch, and A. Preston, "Securing Kafka with encryption-at-rest," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 5378–5387.
[70] H. Zhang, L. Fang, K. Jiang, W. Zhang, M. Li, and L. Zhou, "Secure door on cloud: A secure data transmission scheme to protect Kafka's data," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 406–413.
[71] S. Rooney, P. Urbanetz, C. Giblin, D. Bauer, F. Froese, L. Garcés-Erice, and S. Tomic, "Kafka: The database inverted, but not garbled or compromised," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 3874–3880.
[72] B. Leang, S. Ean, G.-A. Ryu, and K.-H. Yoo, "Improvement of Kafka streaming using partition and multi-threading in big data environment," *Sensors*, vol. 19, no. 1, p. 134, Jan. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/1/134
[73] T. Toliopoulos, A. Gounaris, K. Tsichlas, A. Papadopoulos, and S. Sampaio, "Continuous outlier mining of streaming data in flink," *Inf. Syst.*, vol. 93, Nov. 2020, Art. no. 101569. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437920300594
[74] M. Haghifam, M. N. Krishnan, A. Khisti, X. Zhu, W.-T. Tan, and J. Apostolopoulos, "On streaming codes with unequal error protection," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1165–1179, Dec. 2021.
[75] K. C. Okafor, M. C. Ndinechi, and S. Misra, "Cyber-physical network architecture for data stream provisioning in complex ecosystems," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, p. e4407, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4407
[76] P. Thakur and V. K. Sehgal, "Emerging architecture for heterogeneous smart cyber-physical systems for industry 5.0," *Comput. Ind. Eng.*, vol. 162, Dec. 2021, Art. no. 107750. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835221006549
[77] G. Weichhart, H. Panetto, and A. Molina, "Interoperability in the cyber-physical manufacturing enterprise," *Annu. Rev. Control*, vol. 51, pp. 346–356, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578821000146
[78] C. A. Torres-Charles, D. E. Carrizales-Espinoza, D. D. Sanchez-Gallegos, J. L. Gonzalez-Compean, M. Morales-Sandoval, and J. Carretero, "SecMesh: An efficient information security method for stream processing in edge-fog-cloud," in *Proc. 7th Int. Conf. Cloud Comput. Internet Things (CCIOT)*. New York, NY, USA: Association for Computing Machinery, Sep. 2022, pp. 8–16, doi: 10.1145/3569507.3569509.
[79] D. Wang, J. Ren, Z. Wang, Y. Zhang, and X. Shen, "PrivStream: A privacy-preserving inference framework on IoT streaming data at the edge," *Inf. Fusion*, vol. 80, pp. 282–294, Apr. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521002384

THEOFANIS P. RAPTIS received the Ph.D. degree from the University of Patras, Greece. He was an Associate Researcher with the Computer Technology Institute and Press Diophantus, Greece. He is currently a Senior Researcher with the National Research Council, Italy. He has published in journals, conference proceedings, and books, more than 80 articles on industrial, wirelessly powered, and sensor networks. He is also regularly involved in international IEEE and ACM sponsored conference and workshop organization committees, in the areas of networks, computing, and communications. He has been serving as Editorial Board Member for *Ad Hoc Networks* (Elsevier) and Associate Editor for *IET Networks* (Wiley) and IEEE Access.

ANDREA PASSARELLA received the Ph.D. degree in computer engineering from the University of Pisa, Italy, in 2005. He is currently the a Research Director with the Institute of Informatics and Telematics, National Research Council (CNR). Previously, he was a Research Associate with the Computer Laboratory, University of Cambridge, U.K. He has published more than 180 articles on self-organizing networks, opportunistic networks, online and mobile social networks, distributed AI, and the IoT. He has received multiple best paper awards, including IFIP Networking 2011 and IEEE WoWMoM 2013. He is the Co-Founder and an Associate Editor-in-Chief of the *Online Social Networks and Media* (Elsevier). He has been General Chair of IEEE PerCom 2022 and Program Co-Chair of IEEE WoWMoM 2011, in addition of many international workshops. He is a PI of the EU CHIST-ERA SAI Project and a CNR Co-PI of several projects starting since FP7.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE-ITALY' within the CRUI CARE Agreement