Shojaee Rad and Ghobaei-Arani
Journal of Big Data (2024) 11:82
https://doi.org/10.1186/s40537-024-00939-0

Journal of Big Data

SURVEY Open Access

Data pipeline approaches in serverless computing: a taxonomy, review, and research trends

Zahra Shojaee Rad[1] and Mostafa Ghobaei-Arani[1*]

*Correspondence:
mo.ghobaei@iau.ac.ir

[1] Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

Abstract

Serverless computing has gained significant popularity due to its scalability, cost-effectiveness, and ease of deployment. With the exponential growth of data, organizations face the challenge of efficiently processing and analyzing vast amounts of data in a serverless environment. Data pipelines play a crucial role in managing and transforming data within serverless architectures. This paper provides a taxonomy of data pipeline approaches in serverless computing. Classification is based on architectural features, data processing techniques, and workflow orchestration mechanisms, these approaches are categorized into three primary methods: heuristic-based approach, Machine learning-based approach, and framework-based approach. Furthermore, a systematic review of existing data pipeline frameworks and tools is provided, encompassing their strengths, limitations, and real-world use cases. The advantages and disadvantages of each approach, also the challenges and performance metrics that influence their effectuality have been examined. Every data pipeline approach has certain advantages and disadvantages, whether it is framework-based, heuristic-based, or machine learning-based. Each approach is suitable for specific use cases. Hence, it is crucial assess the trade-offs between complexity, performance, cost, and scalability, while selecting a data pipeline approach. In the end, the paper highlights a number of open issues and future investigations directions for data pipeline in the serverless computing, which involve scalability, fault tolerance, data real time processing, data workflow orchestration, function state management with performance and cost in the serverless computing environments.

Keywords: Data pipeline, Data parallelism, Serverless computing, Function as a service, Big data, Data analysis, Data stream processing, Cloud computing

Introduction

Serverless computing, as a new form of cloud computing execution paradigm, has gained popularity. Such that, the cloud provider takes care of running the server and handles the allocation of resources in a dynamic manner. The serverless does not mean not having a server; indeed, it means an architecture in which the control and management of the server is the responsibility of the cloud service providers [1–4]. The serverless computing platforms offer the advantages of automatic scaling, on-demand computational

resources, high availability, fault tolerance, and cost-effective billing based on actual compute time. These platforms require minimal setup and configuration, making it easier for developers to focus on their applications. Serverless computing has gained immense popularity due to its ability to simplify the deployment and management of applications.

Research motivation

With the increasing adoption of serverless architectures, the need for efficient data processing and analysis has become paramount. This has led to the development of various data pipeline approaches in serverless computing, aimed at enabling seamless data integration, transformation, and delivery. In data-driven era, efficient and reliable data pipelines are crucial for handling the seamless flow of information from diverse sources to target systems. And provides a compelling approach to creating efficient, scalable, and cost-effective data processing workflows. Data pipelines in serverless computing orchestrate the flow of data through various processing stages without the need to manage servers directly. Serverless is presented for linear algebra problems [5], matrix multiplication [6], large-scale optimization [7] and distributed computing [8]. Serverless methodologies have been also applied in the fields of DNA and RNA computing [9, 10], as well as in the development of on-demand high-performance serverless infrastructures and approaches for biomedical computing [11].

Our contribution

This study aims to offer insights into the current of data pipeline approaches in serverless computing through a taxonomy and review. This review guides developers and researchers in the selection of data pipeline approaches so that they can make decisions based on their specific use. The paper provides a thorough examination of existing data pipeline approaches in the realm of serverless computing, assessing their advantages and disadvantages, and presenting potential avenues for future directions. The review's main contributions can be described as follows:

- Offering a comprehensive overview of the current state in data pipeline approaches for serverless computing.
- Introducing a taxonomy that categorizes and defines the various types of approaches available, aiding readers in understanding and selecting the most appropriate approach for their specific requirements.
- Providing a review of the latest research advancements in this field, and updating readers with the latest developments.
- Discussing open issues and suggesting future research directions, encouraging researchers to explore and enhance data pipeline approaches in the context of serverless computing.

Organization of the paper

The structure of this paper is as follows: "Background" section provides an explanation of serverless computing concepts and data pipeline in the serverless context. "Related

works" section discusses related works and provides background information on serverless computing. "Research methodology" section outlines the research methodology employed in the study of serverless computing. "Data pipeline approaches in serverless computing" section examines various data pipeline approaches within serverless computing. In "Discussion" section, discussions and comparisons of the approaches are presented. "Open issues and challenges" section highlights previously unexplored data pipeline issues in serverless computing, introducing them as new challenges for future exploration. Finally, "Conclusions" section presents the conclusion of the paper.

Background

In this section, we explain a conception of serverless computing and data pipeline in serverless computing.

Serverless computing

Serverless computing allows developers to focus on writing code without the need to manage or provision servers, hence the term "serverless". This approach providing a more efficient and scalable model for running applications. Due to its simple management and lightweight nature, serverless computing has gained popularity as an execution model in cloud computing. Within this model, developers can use high-level programming languages such as Java or Python to write functions. They just need to configure some simple parameters and then upload functions to a serverless platform. Then, the applications are broken down into smaller, independent functions or microservices. These functions are event-driven and executed in ephemeral containers that are automatically provisioned and managed by the cloud provider. Each function performs a specific task and can be triggered by events, such as HTTP requests, database updates, or scheduled events. These functions can be invoked through API calls or HTTP requests to perform specific computational tasks. Unlike traditional server-based computing models, developers leveraging serverless computing are relieved from the burden of managing infrastructure resources, as the platforms handle these operational details on their behalf.

Serverless is used to describe the architecture of applications that deploy their services entirely on the infrastructure of a cloud service provider. In this architecture, the management of services is completely in charge of service providers. Serverless companies have developed third-party applications and services that programmers can use (from authentication services to e-mail services to image processing services). These services are called Backend-as-a-Service (BaaS). Another application that can be imagined for serverless computing is that a serverless program is a program that, in order to implement the logic of the program, we must break it into smaller services and execute functions in a stateless state. This computing feature in serverless architecture is called Function-as-a-Service (FaaS).

Serverless can be called an execution model in cloud computing that is implemented in BaaS and FaaS formats. Serverless computing offers developers a simplified approach to deploying code, leveraging high-level abstractions like functions, also known as FaaS, and can be effortlessly deployed without the need for server management [12].

The features of serverless computing are listed below:

- Client transparency: The execution environment is kept hidden from the client. In the serverless model, the client does not need to know the details of the program execution, such as the code storage environment, information about the virtual machine, the container, and the operating system that executes the code. In terms of openness, serverless architecture gives little information to customers.
- Function-centric: Basic elements in serverless architecture are functions, whose resources need to be hidden from the service provider.
- Auto-scaling: The provider must have enabled autoscaling for its services. Resources can be increased upon request.
- Pay-per-use pricing: Serverless services operate on a pay-as-you-go or pay-per-use pricing model. Customers are billed based on the actual usage of resources or the duration of execution. Customers only pay for the resources consumed during the execution of their code, resulting in cost efficiency.
- Compliance with the service level agreement (SLA): The cloud service provider, like all other cloud services, is required to comply with the service level agreement (Service Level Agreement).

Due to its scalability, high elasticity and cost-effective, serverless computing has been widely utilized in various data science applications, including database analysis [13–15] and model training [16–18].

Data pipeline in serverless computing

Data pipelines play a crucial role in contemporary data-driven environments, serving as the foundation for processing, analysis, and decision-making activities. These pipelines facilitate the seamless and dependable movement of data from diverse sources to designated systems, enabling efficient data processing. In other words, a data pipeline refers to a series of processes that extract, transform, and load (ETL) data from various sources to a target destination. It involves steps such as data ingestion, data transformation, data enrichment, and data loading. In a serverless environment, data pipelines can be designed as a series of functions triggered by events like new data arriving in a storage system. Each function performs a specific processing task and can pass the processed data to the next function. This approach provides flexibility and scalability as functions can be added or removed based on workload and processing requirements.

Traditionally, data pipelines were built using on-premises or cloud-based servers, requiring manual provisioning, scaling, and management. But, with the advent of serverless computing, data pipeline management has become more streamlined and efficient. By leveraging serverless architectures, data engineers can build and orchestrate data pipelines using managed services offered by cloud providers. This enables them to focus on business logic and data transformations rather than infrastructure maintenance. A data pipeline typically comprises a sequence of stages or steps that collectively handle the extraction, transformation, and storage of data. There are two primary paradigms commonly used to implement data pipelines: Extract, Transform, Load (ETL) and

Extract, Load, Transform (ELT) [19]. In a serverless data pipeline, each step of the ETL process can be implemented as a function, triggered by events or schedules. These functions are executed in ephemeral compute environments, automatically provisioned and managed by the cloud provider.

Consequently, modern software engineering practices like DevOps [20] and Continuous Integration Continuous Delivery (CI/CD) pipelines [21] have embraced serverless computing to accelerate the development of cloud-native applications. These methodologies endorse the decomposition of an application into multiple functions that are invoked periodically or in response to events. Each function invocation triggers the execution of one or more stateless microservices in the background [22].

Serverless technologies used in data pipeline management include AWS Lambda, Azure Functions, Google Cloud Functions, and Apache OpenWhisk. These services provide the necessary infrastructure to build, deploy, and manage serverless functions.

Related works

This section provides a study of review articles on data pipeline approaches in serverless computing.

Recently, Werner et al. [23] have conducted a study and discusses application-platform co-design, focusing on serverless data processing (SDP). It analyzes the state-of-the-art of FaaS platforms, highlighting differences and ongoing platform (re-)design processes. The article emphasizes the need for specialized serverless platforms and addresses challenges in application design. It proposes the creation of new SDP platforms and stresses the importance of engineering methods and tools for guiding application-platform co-design. But the review article appears to have the following limitations:

- The advantages and disadvantages of the approaches are not presented in a tabulated format, limiting the clarity of the findings.
- The method used to select the articles has not been specified and no specific classification has been done.
- Performance improvements are not shown in a chart, and the case studies and performance metrics have not been specified.

Garcia-Lopez et al. [24] have conducted a study and review the trade-offs and challenges of serverless data analytics. It highlights the limitations of current serverless computing models in supporting various types of analytics workloads. The article explores three fundamental trade-offs: disaggregation, isolation, and simple scheduling, and how relaxing these trade-offs can improve computing performance but may compromise aspects such as elasticity, security, and sub-second activations. The article suggests that a hybrid approach combining serverless and serverful components, known as ServerMix, may be necessary for efficient data analytics applications. But the key limitations of the reviewed article appear to be:

- The method used for selecting the articles and the specific classification criteria are not mentioned, which could have enhanced the thoroughness and transparency of the review.

- Performance improvements are not demonstrated through charts, and the performance metrics as well as the case studies have not been specified.

Wu et al. [25] have investigated the practicality of using serverless computing as a primary platform for model serving in data science applications. The authors conduct a performance and cost comparison between serverless and other model serving systems on Amazon Web Services and Google Cloud Platform. The findings reveal that serverless outperforms several cloud-based alternatives and can even exhibit superior performance compared to GPU-based systems under specific conditions. The article further delves into the design considerations for serverless model serving and offers recommendations to data scientists on optimizing the utilization of serverless technology. However, limitations of the reviewed article are:

- The advantages and disadvantages are not presented in a clear, tabulated format. That could have improved the comprehensiveness and clarity of the review.
- The method used for selecting the articles and the specific classification criteria are not mentioned.

Cordingly et al. [26] have performed an analysis and explored the impact of programming language selection on serverless data processing pipelines. The authors conducted experiments using Java, Python, Go, and Node.js and found that different languages had varying runtime speeds. They highlight the need to carefully consider programming language choice to optimize performance and cost efficiency in serverless applications. However, main flaws in the reviewed article are:

- A systematic approach was not used to select the reviewed articles, and the method of selecting the articles was not clearly defined.
- The advantages and disadvantages are not presented in a clear, tabulated format. That could have improved the comprehensiveness and clarity of the review.

Grzesik et al. [27] have performed an analysis and explored the use of serverless computing in bioinformatics and omics data analysis. It discusses how serverless computing simplifies resource management and enables scalable and parallel execution. The paper emphasizes the application of serverless solutions in integrating and analyzing multiple omics data sources, particularly in relation to the COVID-19 pandemic. But the reviewed article has the following flaws:

- The articles are not classified and the method of selecting the articles is not specified.
- Performance improvements are not demonstrated through charts or other visualizations.

Patel et al. [28] have introduced the DSServe, a serverless framework for data science workflows. It addresses the fluctuating computational needs and bursty nature of tasks in these workflows. By leveraging serverless computing, DSServe enables on-demand scalability and efficient execution of various steps, including automated model selection,

in popular tools like Jupyter Notebooks. The framework optimizes resource utilization and aims to enhance the efficiency of data science projects. But the reviewed article has the following flaws:

- The advantages and disadvantages are not presented in a clear, tabulated format. That could have improved the comprehensiveness and clarity of the review.
- The articles are not classified and the method of selecting the articles is not specified.
- Performance improvements are not demonstrated through charts or other visualizations. And case studies and performance metrics are not stated.

Ihor et al. [29] have explored the use of serverless computing for data processing in open learning and research environments. It proposes a hybrid serverless cloud architecture and presents a case study on wave file processing. The article discusses the challenges and opportunities of integrating serverless components and envisions a cloud-based learning and research environment that enhances education and research accessibility. But the reviewed article has the following flaws:

- The articles are not classified and the method of selecting the articles is not specified.
- Performance improvements are not demonstrated through charts or other visualizations.

Alonso et al. [30] have discussed the concept of serverless computing and its potential benefits for data analytics. It explores the perspectives of users, cloud providers, and researchers on serverless platforms. The article highlights the limitations of current serverless offerings for data analytics and proposes a research agenda to improve the performance and efficiency of serverless computing, particularly in the context of data analytics. But the reviewed article has the following flaws:

- Performance improvements are not demonstrated through charts or other visualizations. And case studies and performance metrics are not stated.
- A systematic approach was not used to select the reviewed articles, and the method of selecting the articles was not clearly defined.

Other papers in the field of data pipeline approaches in serverless computing are fully researched and studied [6, 13, 31–70].

Based on the conducted studies and investigations, efforts are being made to fix the weaknesses and shortcomings, and research and investigations are carried out in this direction.

- Current research papers do not provide a comprehensive classification or detailed comparison of different data pipeline approaches used in serverless computing environments.
- Some of the existing reviews and studies have unclear organization and do not clearly specify the methodology used to select the reviewed research articles.

- Some research papers do not accurately describe the methods or verification techniques used to validate their proposed approaches or findings and lack sufficient specifications.

The main motivation for writing this paper is the importance of processing, analyzing and managing big data in serverless computing environments, which has attracted a lot of attention in recent years. Due to the relevance of this issue, there is a need to analyze and review the various data pipeline approaches used in serverless processing. Also, it is tried to provide future research directions in this field.

Research methodology

Research articles in this section were obtained through a systematic literature review (SLR) process that included searching for relevant keywords, titles, abstracts, and publications.

Keyword search

The validated articles related to data pipelines in serverless computing were searched in various online databases. The online databases utilized for conducting the search included:

- IEEE (http://ieeexplore.ieee.org/)
- Springer (http://www.springer.com)
- Elsevier (http://www.sciencedirect.com)
- ACM (http://dl.acm.org/)
- arXiv (https://www.arxiv.org)
- Oxford (https://academic.oup.com)
- Usenix (https://www.usenix.org)
- MDPI (https://www.mdpi.com)
- CEUR (http://www.ceur-ws.org)
- ETH (http://www.research-collection.ethz.ch)

In the database, the following keywords are searched:

- ("Data pipeline" OR "Data stream processing" OR "Data science") AND ("Serverless") OR ("Serverless computing") OR ("Serverless data") OR ("Big data") OR ("Function-as-a-Service") OR ("FaaS")).

Question formalization

The goal of this study is to pinpoint the essential elements and techniques highlighted in a range of articles concerning serverless computing and data pipeline strategies. It also examines the primary themes and challenges linked to data pipeline approaches within the realm of serverless computing. To fulfill the main aim of the study, which is to thoroughly explore data pipeline strategies in serverless computing, a series of queries regarding the subject should be addressed.

The following are the research technical questions:

TQ1: In serverless computing, what taxonomy is employed in data pipeline approaches? This query is addressed in "Discussion" section.
TQ2: What performance metrics are commonly used for data pipeline approaches in serverless computing? This question is discussed in "Discussion" section.
TQ3: Which case studies are utilized in data pipeline approaches within serverless computing? This question is discussed in "Discussion" section.
TQ4: What evaluation tools are employed for data pipeline approaches in serverless computing? This is explored in "Discussion" section.
TQ5: What are the advantage and disadvantage of data pipeline approaches in serverless computing? This is explored in "Discussion" section.
TQ6: What future research directions and open issues exist for data pipeline approaches in serverless computing? This is discussed in "Open issues and challenges" section.

Data analysis and papers selection

In the research conducted, both conference and journal articles were reviewed and analyzed. Figure 1 illustrates a total of 50 articles analyzed using the Systematic Literature Review (SLR) method, categorized by their publication year. The highest number of articles related to data pipeline approaches in serverless computing was found to be published in 2023.

Figure 2 displays the trend of published papers over time for select publishers [6, 13, 23–70].

Figure 3 shows how the articles were selected and evaluated. The following guidelines and criteria are used to select articles:

- Only taken into account online articles published after 2017.
- The papers must have undergone review in the field of serverless data pipeline.

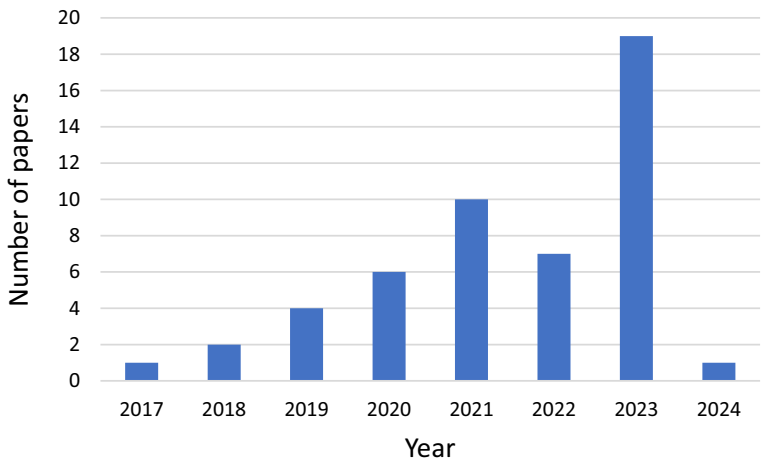

Fig. 1 Number of papers published by year

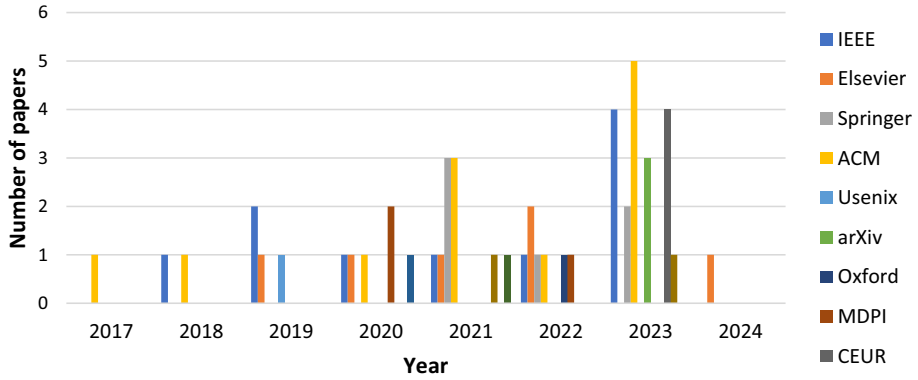

Fig. 2 Breakdown of reviewed articles by publication year and publisher

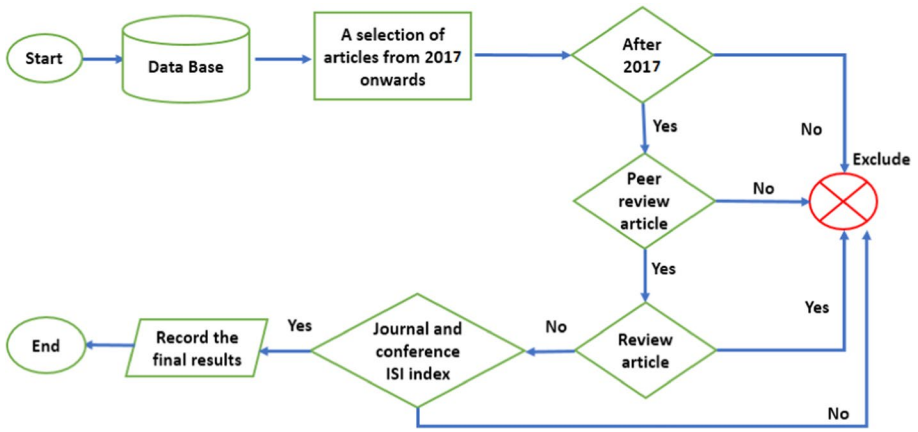

Fig. 3 The process of choosing and assessing articles

- The qualitative aspect of the investigations is considered.

During the exclusion phase, thesis, books, and articles that do not meet the required standards of quality and fail to provide scientifically rigorous and valuable information are excluded. During the elimination stage, the following principles and rules are applied to remove articles:

- Online articles from databases are examined and assessed.
- Articles not included in the ISI (International Scientific Indexing) are deleted.
- Articles that lacking undergone scientific and systematic review are eliminated.
- Articles not written in English are deleted.

Table 1 presents the Reference of the articles, their publication year, and the names of the publications. Only those papers that have been indexed in reputable publications are considered for examination and analysis.

Table 1 Selected articles

No	References	Year	Publisher	Type
1	Related work [23]	2021	Springer	Conference
2	Related work [24]	2021	Springer	Conference
3	Related work [25]	2022	ACM	Conference
4	Related work [26]	2020	IEEE	Conference
5	Related work [27]	2022	Oxford	Journal
6	Related work [28]	2022	IEEE	Conference
7	Related work [29]	2023	CEUR	Conference
8	Related work [30]	2023	CEUR	Conference
9	Machine learning-based [31]	2021	ACM	Conference
10	Machine learning-based [32]	2021	Elsevier	Journal
11	Machine learning-based [33]	2022	Springer	Conference
12	Machine learning-based [34]	2023	ACM	Conference
13	Machine learning-based [35]	2023	IEEE	Journal
14	Machine learning-based [36]	2020	IJSTR	Journal
15	Machine learning-based [37]	2019	Usenix	Conference
16	Heuristic-based [38]	2020	Elsevier	Journal
17	Heuristic-based [39]	2019	IEEE	Conference
18	Heuristic-based [6]	2018	IEEE	Conference
19	Heuristic-based [40]	2022	Elsevier	Journal
20	Heuristic-based [41]	2019	IEEE	Conference
21	Heuristic-based [42]	2023	ACM	Conference
22	Heuristic-based [43]	2023	IEEE	Conference
23	Heuristic-based [44]	2020	MDPI	Journal
24	Heuristic-based [45]	2021	Springer	Conference
25	Heuristic-based [46]	2023	IEEE	Conference
26	Heuristic-based [47]	2021	Frontiers	Journal
27	Heuristic-based [48]	2022	MDPI	Journal
28	Heuristic-based [49]	2023	Springer	Journal
29	Heuristic-based [50]	2023	CEUR	Conference
30	Heuristic-based [51]	2023	IEEE	Conference
31	Framework-based [52]	2024	Elsevier	Journal
32	Framework-based [53]	2022	Elsevier	Journal
33	Framework-based [13]	2020	ACM	Conference
34	Framework-based [54]	2023	Springer	Journal
35	Framework-based [55]	2021	ACM	Conference
36	Framework-based [56]	2023	arXiv	Preprint
37	Framework-based [57]	2023	arXiv	Preprint
38	Framework-based [58]	2023	ACM	Conference
39	Framework-based [59]	2023	arXiv	Preprint
40	Framework-based [60]	2023	ACM	Conference
41	Framework-based [61]	2023	ACM	Conference
42	Framework-based [62]	2018	ACM	Conference
43	Framework-based [63]	2023	CEUR	Conference
44	Framework-based [64]	2021	ETH	Conference
45	Framework-based [65]	2017	ACM	Conference
46	Framework-based [66]	2019	Elsevier	Journal
47	Framework-based [67]	2023	CEUR	Conference
48	Framework-based [68]	2021	IEEE/ACM	Conference
49	Framework-based [69]	2020	MDPI	Journal

Table 1 (continued)

No	References	Year	Publisher	Type
50	Framework-based [70]	2021	ACM	Conference

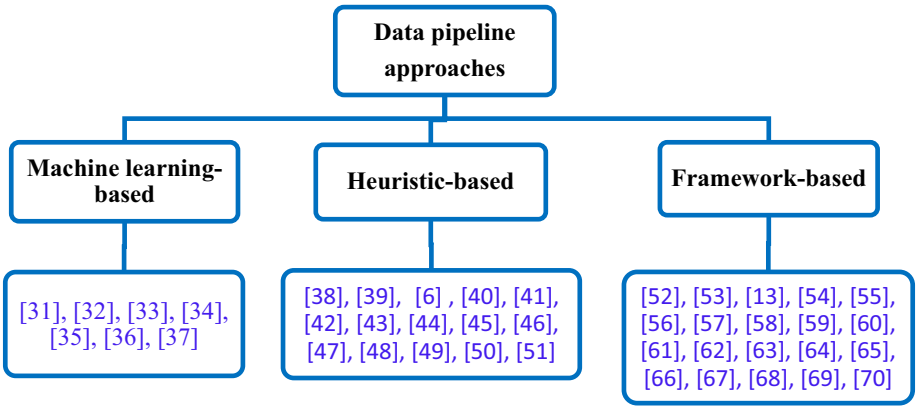

Fig. 4 Taxonomy of data pipeline approaches in serverless computing

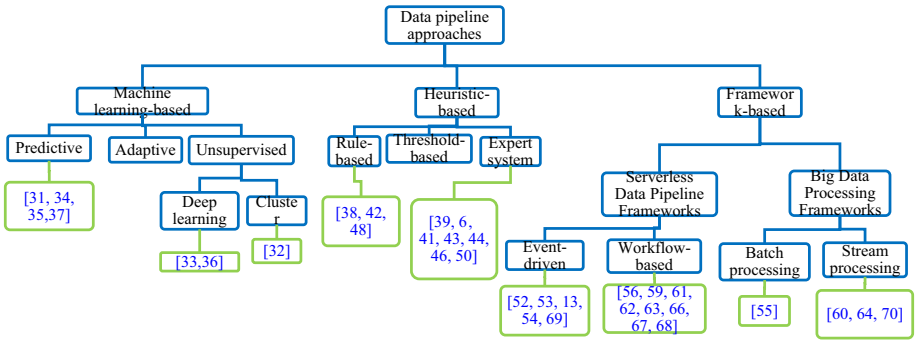

Fig. 5 A detailed taxonomy of data pipeline approaches in serverless computing

Data pipeline approaches in serverless computing

In this section, according to the articles that have used the SLR method; a detailed review and evaluation of various approaches and techniques in serverless computing has been done. Data pipeline approaches in serverless computing reviewed and studied in selected articles are analyzed. These methods are classified into three main categories: heuristic-based, machine learning-based, and framework-based approaches. The taxonomy of data pipeline approaches categorizes the data pipeline approaches in serverless computing into three main types, as shown in Fig. 4. Besides, we provide a more detailed taxonomy in Fig. 5 and the selected papers are shown in this classification:

- Machine Learning-based mechanisms: These mechanisms employing predictive models or adaptive techniques to optimize pipeline performance [31–37]. In this study, we categorized the machine learning-based mechanisms into three classes, as following:

- Predictive: Utilizes machine learning models to predict and optimize pipeline performance.
- Adaptive: Employs machine learning techniques to dynamically adapt the pipeline based on evolving data patterns.
- Unsupervised learning: Identify patterns and relationships in the data to automate pipeline tasks like data selection, feature engineering, or anomaly detection.

- Heuristic-based mechanisms: These mechanisms relied on predefined rules or thresholds to manage and orchestrate the data pipeline [6, 38–51]. In this work, we categorized the heuristic-based mechanisms into three classes, as following:

  - Rule-based: Uses predefined rules or heuristics to manage and orchestrate the data pipeline.
  - Threshold-based: Leverages thresholds or triggers to determine when to scale or modify the pipeline.
  - Expert systems: Capture human expertise and decision-making processes into rules or models to automate specific pipeline tasks.

- Framework-based mechanisms: These mechanisms utilizing serverless data pipeline frameworks or big data processing frameworks to orchestrate the pipeline [13, 52–70]. In this work, we categorized the framework-based mechanisms into two classes, as following:

  - Serverless data pipeline frameworks

    - Event-driven: Orchestrates the pipeline based on event triggers (e.g., AWS Lambda, Google Cloud Functions).
    - Workflow-based: Provides a workflow engine to define and manage the pipeline (e.g., AWS Step Functions, Azure Durable Functions).

  - Big data processing frameworks

    - Batch processing: Leverages batch-oriented big data frameworks (e.g., Apache Spark, Apache Flink).
    - Stream processing: Utilizes stream-oriented big data frameworks (e.g., Apache Kafka, Apache Storm).

Machine learning-based approaches

This section describes the characteristics of the machine learning technique utilized for data pipeline strategies in serverless computing. Machine learning-based approaches in data pipelines in serverless computing involve incorporating machine learning algorithms and techniques within the data processing and transformation stages of the pipeline. These approaches leverage serverless computing capabilities to perform machine

learning tasks efficiently and effectively. In the following, the studied articles in this field are evaluated and analyzed.

Overview on the machine learning-based approaches

Nesen et al. [31] have discusses the importance of extracting useful knowledge from large amounts of data and processing it in a fast and scalable manner. It proposes a framework for processing data from multiple modalities, such as text, video, and sensor data, using serverless computing. Also highlights the application of this framework in public safety solutions to increase situational awareness. Although the article lacks specific implementation details and evaluation of the proposed framework for processing multimodal data. It may have limitations such as possible latency issues and resource access restrictions.

Rausch et al. [32] have introduced Skippy, a container scheduling system that optimizes the placement of serverless edge functions. Skippy addresses limitations of existing serverless platforms in managing data-intensive applications on edge systems by considering factors like data proximity, compute capabilities, and edge/cloud locality. It improves task placement and enables operational goals to be met in edge computing scenarios. Although the proposed method requires manual fine-tuning of timing constraints. This approach may introduce limitations and inefficiencies in the scheduling process. It requires extensive operational data and expert knowledge, and may not scale well in dynamic edge environments.

León-Sandoval et al. [33] have discussed the use of big data and serverless architecture to monitor and measure the emotional response of the Mexican population to the COVID-19 pandemic. The study utilizes a large dataset of public domain tweets Twitter and applies sentiment analysis tools to analyze the changes in sentiment towards the pandemic, news cycles, and government policies. The article highlights the advantages and challenges of implementing serverless cloud-based architectures for large-scale natural language processing projects. Although social media data, such as Twitter, can be a valuable source, it still requires significant computing resources to process and analyze and can be expensive and time-consuming.

Anshuman et al. [34] have introduced a system called Smartpick, which combines serverless (SL) and virtual machine (VM) resources to optimize cost and performance in data analytics systems. Smartpick uses machine learning prediction models to determine the optimal configuration of SL and VM instances based on workload characteristics. It also supports a mechanism called relay-instances to improve performance and offers a simple knob for applications to explore the tradeoff between cost and performance. Experimental results demonstrate significant cost reduction and efficient handling of workload dynamics with Smartpick. However, one disadvantage could be the complexity involved in determining optimal configurations of serverless and virtual machine instances. Factors such as varying compute resource characteristics, workload prediction, diverse cost-performance goals, and workload dynamics make this task challenging. While the Smartpick system utilizes machine learning techniques to address these challenges, accurately predicting and optimizing configurations may pose difficulties or limitations.

Efterpi et al. [35] have discussed the use of serverless computing and Function-as-a-Service (FaaS) to facilitate Machine Learning Functions-as-a-Service (MLFaaS). It presents an approach for creating composite services, or workflows, of ML tasks within a serverless architecture, allowing data scientists to focus on the complete data path functions required for their analysis. Also, addresses the challenge of function selection and recommends an AI-based technique for optimizing the number of functions in a pipeline to improve performance. However, extending functions through attached containers to overcome serverless constraints may introduce additional complexity to the development and deployment process. Besides, the introduction of extended containers may add overhead in terms of latency and overall execution time.

Rahman et al. [36] have discussed the use of serverless computing for big data analytics, focusing on a personalized recommendation system, and how serverless computing can provide a cost-effective and high-performance solution for processing and analyzing large amounts of data. The article proposes a serverless architecture using Amazon Web Services (AWS) and evaluates it using a real-world case study involving the Movielens dataset and Amazon Personalization Hierarchical Recurrent Neural Networks (HRNN) algorithm. However, disadvantage could be the complexity of designing and implementing an effective architecture that ensures scalability, security, and cost efficiency. Additionally, AWS serverless does not have all the capabilities to create a data lake, and processing and analyzing data in the cloud can still be costly.

Bhattacharjee et al. [37] have introduced "Stratum", a serverless framework designed for the lifecycle management of machine learning-based data analytics tasks. It addresses the challenges of ML model development and deployment by providing an end-to-end platform that can deploy, schedule, and dynamically manage various data analytics tools and services across the cloud, fog, and edge computing environments. Stratum aims to simplify ML development, enhance performance, and minimize costs associated with resource management. However, the complexity of deploying and managing ML models across cloud, fog, and edge resources may require a certain level of expertise with the framework.

Summary and discussion

According to reviewed studies, research papers propose approaches and systems of serverless computing in different domains, such as data processing, edge computing, sentiment analysis, data analytics, and machine learning. Some challenges include latency issues, resource access restrictions, manual fine-tuning, complexity in determining optimal configurations, and the introduction of additional complexity or overhead in the development and deployment process. Articles describe the advantages of serverless architectures, such as scalability, cost-effectiveness, and simple development. However, one repeated challenge is the complexity involved in determining optimal configurations of serverless and virtual machine instances. Factors like workload characteristics, diverse cost-performance goals, and workload dynamics make this task challenging. While machine learning techniques can help address these challenges, accurately predicting and optimizing configurations may still pose difficulties. Another limitation is the need for extensive computing resources to process and analyze large datasets. While serverless architectures offer scalability, processing social media data, for example, can

Table 2 Comparison of machine learning-based data pipeline approaches

Article	Main idea	Advantage	Disadvantage
[31]	• Presenting a framework for processing data from multi-modal sources using a serverless computing approach	• Processing it in a fast and scalable • Optimize cost	• Latency • Resource access restrictions
[32]	• Skippy • Introducing a container scheduling system for serverless edge computing	• Efficient edge resource utilization • Cost optimization • Minimizing execution time	• Complexity • Expert knowledge requirement
[33]	• Proposing a resilient and flexible system that utilizes big data and serverless architecture to track and measure the changes in sentiment	• Analyze a vast amount of data in a short period • Availability of real-time analysis	• High cost of acquiring emotional data • Complexity of handling heterogeneous data
[34]	• Introducing Smartpick, a serverless-enabled scalable data analytics system that combines the benefits of serverless (SL) and virtual machine (VM) compute resources	• Prediction accuracy • Cost Reduction	• SL performance and cost (Smartpick aims to mitigate the performance and cost issues with SL, but SL may offer worse performance and be more expensive than VM) • Complexity
[35]	• Proposing an approach for facilitating the provision of Machine Learning Functions-as-a-Service (MLFaaS)	• Scalability • Flexibility • Abstraction and efficiency • Quality-of-service (QoS)-Awareness	• Resource limitations • Complexity for data analysts
[36]	• Investigating the application of serverless computing in the field of big data analytics, particularly in a personalized recommendation system	• Low cost • High performance • Scalability • Security • Focus on application development • Simplicity of the development and deployment process	• Vendor lock-in • Limited control • Cold start latency • Complexity of designing
[37]	• Introducing a serverless framework called "Stratum" for the lifecycle management of machine learning-based data analytics tasks	• Rapid development • Rapid deployment • Minimize cost • Model transfer and flexibility • Extensibility and reusability	• Cold start latency • Limited execution time • Complexity

still be expensive and time-consuming. Additionally, the complexity of designing and implementing effective architectures that ensure scalability, security, and cost efficiency remains a challenge.

Finally, Tables 2 and 3 provide a complete comparison of the main idea, advantages, and disadvantage, case study, performance metric, technique used, evaluation tool, for each paper.

Heuristic-based approaches

This section describes on the characteristics associated with heuristic-based methods utilized for data pipeline approaches in serverless computing. Heuristic-based approaches in data pipelines in serverless computing involve using heuristics method or a novel approach developed by the researchers or rule-based methods to make decisions and perform data processing and transformation tasks within the pipeline. These approaches leverage defined rules or algorithms to guide the data pipeline's behavior and achieve specific objectives.

Table 3 A side-by-side comparison of machine learning-based data pipeline approaches

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[31]	• Machine learning • Feature and pattern extraction • Data fusion	• Cost • Speed	• AWS • EC2 • S3	• Use of multimodal data in police investigations, where information is gathered from surveillance cameras, incident reports, and social networks
[32]	• Skippy, is an online scheduler that implements a greedy multi-criteria decision-making (MCDM) algorithm	• Execution time • Resource utilization • Cost	• OpenFaaS • Apache • Python • Simulator	• Optimized container scheduling for data-intensive serverless edge computing
[33]	• Deep learning	• Efficiency • Scalability • Accuracy	• Google cloud platform (GCP) • Python	• Tracking and measuring the sentiment changes of the Mexican population in response to the COVID-19 pandemic
[34]	• Machine learning • Decision-tree based Random Forest (RF) • Bayesian optimizer (BO) • Relay-instances	• Cost reduction • prediction accuracy • Workload performance	• Amazon AWS • Google cloud platform (GCP) • Spark	• A system called Smartpick, which is a serverless-enabled scalable data analytics system
[35]	• Machine learning • Artificial intelligence techniques	• Latency • Execution time • Scalability • Resource utilization	• Apache OpenWhisk • Python	• Design and realization of a chain/pipeline of Machine Learning (ML) functions using a serverless architecture
[36]	• Hierarchical recurrent neural networks (HRNN) algorithm	• Execution time	• Amazon Web Services (AWS) • Amazon S3 • Amazon EC2	• Using Movielens data for personalized recommendation using Amazon personalized hierarchical recurrent neural networks (HRNN) algorithm
[37]	• Use of machine learning libraries and frameworks	• Performance • Cost	• Modeling language DSML • Python (TensorFlow, Scikit Learn, PyTorch)	• Smart traffic management system where traffic cameras collect videos

Overview on the heuristic-based approaches

Enes et al. [38] have introduced a novel platform for scaling resources in real time for Big Data workloads on serverless environments. It proposes a system that dynamically adjusts container resources without the need for restarts, using operating-system-level virtualization. The platform is evaluated using representative Big Data workloads and demonstrates improved CPU utilization and scalability while maintaining performance. But the platform relies on operating-system-level virtualization, specifically Linux Containers (LXC). While LXC offers lightweight virtualization capabilities, it may have certain limitations compared to other virtualization technologies.

Kuhlenkamp et al. [39] have discussed the evaluation of Function-as-a-Service (FaaS) platforms as a foundation for serverless big data processing. It introduces a Serverless Infrastructure Evaluation Method (SIEM) to understand the impact of automatic infrastructure management on serverless big data applications. The authors propose new metrics and evaluate four major FaaS providers, providing insights for FaaS-based big data processing. But introducing new metrics and a novel evaluation method might add

complexity. Also, implementing SIEM might require specific expertise or custom tools, potentially creating barriers for broader adoption.

Werner et al. [6] have discussed the feasibility and benefits of using serverless computing for big data processing. They use matrix multiplication. They define requirements for serverless big data applications, present a prototype using Function-as-a-Service (FaaS), and conduct extensive experiments to evaluate the performance and scalability. The results show that serverless big data processing can reduce operational and infrastructure costs while maintaining system qualities, and it can even outperform traditional cluster-based distributed compute frameworks. Although relying on a specific FaaS provider might limit portability and flexibility that's mean vendor lock-in. Also, serverless functions execute on ephemeral containers, leading to cold start delays when invoked after inactivity.

Shivananda et al. [40] have explored the use of serverless computing and data pipelines for handling Internet of Things (IoT) data in fog and cloud computing environments. It investigates three different approaches for designing serverless data pipelines and evaluates their performance using real-time fog computing workloads. The study highlights the benefits and challenges of combining serverless computing and data pipelines in IoT applications, considering factors such as computation time, network communication, disk access time, and resource utilization. However, the serverless functions run in a shared environment, which can raise security concerns for sensitive data. Additionally, managing access control and permissions for serverless functions can be complex.

Toader et al. [41] have introduced a serverless graph-processing system called "Graphless" designed to make graph processing more accessible. Graphless combines the serverless computing paradigm with the data-intensive nature of graph processing through an architectural approach and backend services. Real-world experiments show that Graphless performs similarly to existing graph-processing systems but is easier to deploy and offers both push and pull operation. Although rely on a specific serverless provider (e.g., Amazon Lambda) could limit portability and introduce vendor lock-in. Also, Serverless functions may experience a cold start when called, which affects performance for latency-sensitive workloads compared to dedicated graph processing systems.

Bian et al. [42] have discussed the use of cloud function (CF) services, such as AWS Lambda, as accelerators for elastic data analytics. It compares CFs to traditional query engines running on virtual machines (VMs) and explores their limitations in terms of storage, network, and higher resource unit prices. The article proposes a hybrid query engine called Pixels-Turbo, which leverages CFs to accelerate processing during workload spikes while using a scalable VM cluster for regular query processing. The evaluation shows that this approach achieves a higher performance/price ratio compared to existing serverless query engines. However, managing and optimizing a hybrid system can be more complex than using either VMs or CFs alone.

Jarachanthan et al. [43] have introduced ACTS, an autonomous cost-efficient task orchestration framework for serverless analytics. It addresses the challenges in adapting data analytics applications to the serverless environment by mitigating cold-start latency, reducing state sharing overhead, and optimizing cost efficiency. Extensive experiments show that ACTS achieves significant monetary cost reduction while maintaining superior job completion time performance compared to existing baselines. While this

approach may not be efficient for scenarios with high data volumes or complex data dependencies between functions. Also managing and coordinating the execution of functions through calls may introduce complexity and performance bottlenecks.

Pogiatzis et al. [44] have presented a serverless architecture for Extract, Transform, and Load (ETL) pipelines on the AWS platform. The architecture is event-driven, allowing for real-time data processing and scalability. The article includes an evaluation of the architecture's performance and discusses its advantages and limitations. But the utilization of an SQS queue for data transfer can become a bottleneck for large-sized event payloads. This means that when dealing with a high volume of data, the efficiency of data transfer may decrease.

Bharti et al. [45] have proposed a novel design approach for serverless applications that leverages data parallelism in embarrassingly parallel computations. The approach aims to overcome limitations imposed by serverless platforms on compute-intensive tasks, allowing them to be successfully executed. The research presents a design methodology, a case study on distributed matrix multiplication, and validation through load testing and performance comparison. Although proposed method is focuses on embarrassingly parallel computations, which are simple and efficient algorithms that can be easily divided into independent subproblems. Therefore, the proposed method may have limited applicability in scenarios that involve complex dependencies or non-parallelizable tasks.

Sanchez-Gallegos et al. [46] have introduced a model called MeshStore, which is a serverless storage system designed for edge-fog-cloud continuum systems. The model aims to integrate heterogeneous storage resources into a unified storage service that supports data sharing through serverless functions. It provides mechanisms for managing data allocation, load balancing, and synchronization in distributed environments. However, it increases the complexity of managing operations required for functions to retrieve received data and provide results to other functions. Additionally, there are security and privacy concerns associated with storing and sharing sensitive data in a distributed manner.

Mrozek et al. [47] have presented a large-scale and serverless computational approach for improving the quality of next-generation sequencing (NGS) data in support of big multi-omics data analyses. They propose the use of a Data Lake for storing and processing NGS data, along with a dedicated library for cleaning DNA/RNA sequences. Their solution is scalable on the Cloud and provides capabilities for data extraction, processing, and storing, supporting the requirements of NGS-based multi-omics data analyses. However, relies on the serverless nature of the Data Lake Analytics service. While this approach offers benefits in terms of reduced operational overhead, it may also have limitations in terms of the capabilities and performance of the data lake platform.

Pakdil et al. [48] have discussed the design of a serverless geospatial data processing workflow system. It explores how the serverless paradigm can be utilized for geospatial data processes using open standards. They propose a system design and architecture that minimizes human intervention and resource consumption, while also incorporating new models for workflow and task definitions. They implemented the system on a public cloud provider and evaluated it with sample geospatial workflows.

However, may require a certain level of expertise in cloud computing technologies. In addition, relying on cloud computing platforms creates a dependency on external service providers, which can affect system availability and reliability.

Rivera et al. [49] have presented a study on event-driven serverless pipelines for video coding and quality metrics. It discusses the implementation of serverless functions using an adapted version of embedded Tomcat and explores their behavior in terms of scalability and resource consumption. The study shows that the proposed serverless functions perform well in terms of encoding time and distribution of jobs. However, in serverless architectures, control over the infrastructure is lost and the cloud service provider manages the infrastructure, meaning users have limited customization and optimization options. This lack of control is a drawback for applications or organizations that require specific infrastructure configurations or optimizations to meet their unique needs.

Spiegelberg et al. [50] have discussed of hyper-specialized compilation for serverless data analytics. It argues that while existing serverless frameworks generate and compile code on the client, it is more beneficial to generate specialized code on each serverless function based on the specific input data. Preliminary experiments show that hyper specialization outperforms client-based compilation in terms of cost and performance. However, this method lead to the increased overhead and complexity of generating and compiling code on individual serverless functions. This approach requires additional resources and time for code generation and compilation, which could impact the overall efficiency and scalability of the system.

Cinaglia et al. [51] have presented a method for modeling and executing customized pipelines in serverless computing. The method is applied to the transcript-level expression analysis of samples from RNA sequencing (RNA-seq). The authors implemented the method as an Amazon Web Services (AWS) Lambda function within their own serverless architecture. The method demonstrates improved computational time compared to local environments, with potential advantages for parallel analysis of large-scale genomic data. But disadvantage of this method could be the reliance on specific cloud service providers, such as Amazon Web Services (AWS), for the implementation of the serverless architecture. This may limit the portability and flexibility.

Summary and discussion

These articles provide insights into the benefits and challenges of using serverless computing for big data processing and related applications. Serverless computing offers advantages such as improved resource utilization, scalability, reduced operational and infrastructure costs, and ease of deployment. However, there are also limitations and challenges to consider.

One concern is vendor lock-in, where reliance on specific serverless providers may limit portability and flexibility. Another challenge is cold start delays, where serverless functions experience latency when invoked after inactivity. This can impact the performance of latency-sensitive workloads. Security and privacy are also important, as serverless functions run in shared environments and managing access control and permissions can be complex. Additionally, the utilization of certain technologies, such as Linux Containers (LXC), may have limitations compared to other virtualization technologies. Some

Table 4 Comparison of heuristic-based data pipeline approaches

Article	Main idea	Advantage	Disadvantage
[38]	• Introducing a novel platform for scaling resources in real time for Big Data workloads on serverless environments	• Improve CPU utilization • Improve scalability	• Platform compatibility limit
[39]	• SIEM • Introducing a serverless infrastructure evaluation method (SIEM) to understand the impact of automatic infrastructure management on serverless big data applications	• Introducing new metrics and evaluation methods	• Complexity
[6]	• Feasibility and benefits of using serverless computing for big data processing	• Reduce cost • Performance	• Limit portability • Limit flexibility • Vonder lock-in • Cold start delay
[40]	• Use of serverless computing and data pipelines for handling Internet of Things (IoT) data in fog and cloud computing environments	• Auto-scaling • Increased productivity • Migration of tasks between edge and cloud • Dynamic execution of tasks	• Complexity • Security concerns
[41]	• Graphless • Introduce a serverless graph-processing system called "Graphless" designed to make graph processing more accessible	• Easier to deploy	• Limited portability • Vendor lock-in • Cold start delay
[42]	• Use of cloud function (CF) services, such as AWS Lambda, as accelerators for elastic data analytics • Pixels-turbo	• Accelerate processing Scalable	• Complexity • Less security • Less privacy
[43]	• ACTS • ACTS, an autonomous cost-efficient task orchestration framework for serverless analytics	• Reduced cold-start latency • Reduce overhead • Cost reduction	• Complexity • Performance bottleneck in high data volume
[44]	• A serverless architecture for extract, transform, and load (ETL) pipelines on the AWS platform	• Scalable • Easy concurrency control • Easy data slicing	• Reduction of data transmission efficiency in high data volume
[45]	• A novel design approach for serverless applications that leverages data parallelism in embarrassingly parallel computations	• Auto-scalability	• Limited applicability in complex dependencies • Limited applicability in non-parallelizable tasks
[46]	• MeshStore • MeshStore, which is a serverless storage system designed for edge-fog-cloud continuum systems	• Managing data allocation • Load balancing • Data synchronization	• Complexity • Less security • Less privacy
[47]	• A large-scale and serverless computational approach for improving the quality of next-generation sequencing (NGS) data in support of big multi-omics data analyses	• Scalability • Data cleaning • Simplified data analysis	• Limitations of data lake platform
[48]	• Design of a serverless geospatial data processing workflow system	• Minimizing human intervention • Reduce resource consumption	• Vendor lock-in • Reduce availability • Reduce reliability
[49]	• Event-driven serverless pipelines for video coding and quality metrics	• Reduce encoding time • Well distribution of jobs	• Loss of control over infrastructure • Limited customization
[50]	• Hyper-specialized compilation for serverless data analytics	• Cost reduction • Better performance	• Overhead increase • Complexity • Additional resources and time

Table 4 (continued)

Article	Main idea	Advantage	Disadvantage
[51]	• A method for modeling and executing customized pipelines in serverless computing	• Improved run time • High scalability	• Vendor lock-in • Reliance on specific cloud service providers • Limited portability • Limited flexibility

articles propose novel evaluation methods or metrics, to address specific challenges, but these may introduce complexity or require specific expertise or custom tools, creating barriers for broader adoption.

Finally, Tables 4 and 5 provide a complete comparison of the main idea, advantages, and disadvantage, case study, performance metric, technique used, evaluation tool, for each paper.

Framework-based approaches

This section describes the characteristics concerning framework-based approaches employed in data pipeline implementations within serverless computing. Indeed, there exist techniques leveraging framework, platform, architecture, etc. Framework-based approaches in data pipeline in serverless computing refer to the utilization of frameworks or platforms that provide a structured and efficient way to design, develop, and deploy data pipelines.

Overview on the framework-based approaches

Mirampalli et al. [52] have discussed the evaluation of two serverless data pipeline approaches, NiFi and Message Queuing Telemetry Transport (MQTT), in fog computing environments. The study focuses on image streaming data and compares the performance of the two approaches in terms of pipeline execution time, memory usage, and CPU usage. The results indicate that while the NiFi-based serverless pipeline consumes more CPU, it outperforms the MQTT-based pipeline in terms of execution time and memory utilization. Although disadvantage of this method is the higher CPU usage associated with the NiFi-based serverless data pipeline. This could be a limitation in resource-constrained environments where minimizing CPU utilization is crucial.

Dehury et al. [53] proposes an extension to the TOSCA standard called TOSCAdata, which focuses on modeling data pipeline-based cloud applications. Keeping the requirements of modern data pipeline cloud applications, TOSCAdata provides a number of TOSCA models that are independently deployable, schedulable, scalable, and re-usable. TOSCAdata provides models that enable the efficient handling of data flow and transformation in a pipeline manner. But may introduce additional complexity to the TOSCA modeling language. While TOSCAdata aims to address the challenge of designing and orchestrating data-intensive cloud applications.

Müller et al. [13] have presented Lambada, a serverless distributed data processing framework designed for data analytics. They explore the suitability of serverless computing for data processing and demonstrate its cost and performance advantages in certain scenarios. They provide examples where serverless outperforms existing solutions.

Table 5 A side-by-side comparison of heuristic-based data pipeline approaches

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[38]	• Operating-system-level virtualization in the form of Linux containers (LXC)	• Execution time overhead • CPU utilization • Scalability • Cost	• BDWatchdog • Python • JSON	• Novel platform for real-time resource scaling of Big Data workloads on serverless environments
[39]	• SIEM (quality-driven evaluation of serverless big data applications)	• Scalability • Consistency • Cost-effectiveness	• AWS • Google Cloud • Function (GCF) • IBM Cloud Function (ICF) • Microsoft Azure Functions (MAF)	• Evaluating • Function-as-a-Service (FaaS) platforms as a foundation for serverless big data processing
[6]	• Distributed matrix multiplication • Hadoop MapReduce	• Scalability • Performance • Tuneability • Cost	• AWS • Apache Spark • Hadoop • Amazon EMR • S3	• Serverless big data processing using matrix multiplication as an example
[40]	• Utilizing of three approaches DFT-based, OSS-based, and MQTT-based	• CPU usage • Memory usage • Disk access time • Computation time • Network access time	• AWS • Open FaaS • Apache Nifi	• Investigating the benefits of building Serverless data pipelines (SDP) for IoT data analytics • Custom video processing application
[41]	• Simplifying computing by allowing developers to focus on small, stateless functions • Using of LDBC Graphalytics to analyze Graphless	• Cost • Performance • Execution time • Scalability	• AWS • Amazon Lambda • Apache Giraph • GraphMat • EC2 • S3	• Design and implementation of the Graphless graph-processing system
[42]	• Pixels-turbo • Utilizes CFs as an auxiliary computing resource to process unpredictable workload spikes	• Cost • Performance • Scalability	• AWS • VM (EC2) • TPC-H	• Using cloud functions (CFs) as accelerators for elastic data analytics
[43]	• ACTS as an autonomous cost-efficient task orchestration framework • Exploits warm containers to avoid cold-start latency, and optimizes the allocation of function resources	• Cost • Performance • Completion time reduction	• AWS • Amazon S3 • REST API • Python	• Autonomous cost-efficient task orchestration for serverless analytics

Table 5 (continued)

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[44]	• AWS simple queue service (SQS) • Use of an SQS queue for data transfer	• Consistency • Reliability • Maximum payload size per pipeline • Economic scalability (cost of chargeable tasks) • Throughput	• AWS • Amazon S3 • Python	• Implementation an event-driven extract, transform, and load (ETL) pipeline architecture
[45]	• Data parallelism • Breaking down a large computation into small serverless functions and executing them in parallel	• Performance • Execution time • Cost effective • Resilience	• AWS • AWS S3 • AWS X-Ray • Apache JMeter • Apache Bench • Python	• Design for serverless applications that exploit data parallelism in embarrassingly parallel computations
[46]	• MeshStore • Utilization of a serverless architecture approach, where functions are deployed on edge, fog, and cloud infrastructures to process data	• Cost-efficiency • Integrity • Reliability • Confidentiality	• Open FaaS • Amazon EC2	• A general framework and model called MeshStore for serverless storage in edge-fog-cloud systems
[47]	• Utilization of a Data Lake approach for storing and processing big NGS data • Dedicated library for cleaning DNA/RNA sequences	• Scalable	• Azure cloud • U-SQL • C#	• Improving the quality of Next-Generation Sequencing (NGS) data and supporting big multi-omics data analyses
[48]	• Using of serverless technologies	• Resource consumption • Scalability • Execution time	• AWS •.NET • C#	• Design of a serverless geospatial data processing workflow system
[49]	• Implementing serverless functions using Tomcat and deploys them on a serverless platform based on Knative	• Scalability • Resource consumption (CPU and memory) • Reduction encoding time	• CloudEvents • Knative • JSON	• Development and analysis of event-driven serverless pipelines for video coding and quality metrics
[50]	• Hyper specialization, which involves generating and compiling bespoke code on each serverless function • Viton	• Cost • Performance	• AWS • Tuplex • JSON • EC2	• Hyperspecialized compilation for serverless data analytics
[51]	• Using parallel instances invoked on AWS Lambda to optimize computational time for RNA-seq data analysis	• Run time	• AWS Lambda • Python	• Analysis of RNA-seq data, specifically the mapping of reads to a reference genome

However, disadvantage of the proposed method is the limited network connectivity and stateless operation of serverless functions. Additionally, the lack of control over the scheduling of functions can introduce uncertainty in the execution and response times of queries.

Sedlak et al. [54] have proposed a novel approach for sharing privacy-sensitive data within federations of independent organizations. The approach combines data meshes and serverless computing to streamline data sharing processes and address the specific requirements of variable data sharing constellations, with a focus on flexibility and efficiency. However, it may introduce additional complexity and dependencies on serverless computing infrastructure. Additionally, there may be challenges in managing and coordinating the serverless functions across multiple organizations within the federation.

Romero et al. [55] have introduced Llama, a heterogeneous and serverless framework for auto-tuning video analytics pipelines. Llama optimizes pipeline configurations to meet latency and cost targets by dynamically assigning configurations across different hardware resources. The framework addresses the challenges of handling input-dependent behavior, conditional branches in the pipeline, and execution variability. Experimental results demonstrate that Llama outperforms existing video analytics and processing systems in terms of latency reduction and cost efficiency. Although complexity and computational overhead involved in dynamically optimizing configurations for each operation invocation. Additionally, the offline profiling phase required for the framework's initial setup may be time-consuming and may need to be repeated for different pipelines or input videos.

Ríos-Monje et al. [56] have discussed the application of FaaS or Serverless computing in the context of scientific data processing, focusing on the Square Kilometer Array (SKA), a large radio telescope project. They explore the feasibility of designing and deploying functions and applications commonly used in radio astronomy workflows within a FaaS platform. They emphasize the scalability, cost-effectiveness, and potential value of FaaS models for scientific data processing in distributed projects like SKA. But disadvantage is the complexity of integrating and deploying scientific functions and applications within a FaaS platform. And it is likely that integrating and adapting existing scientific workflows and applications to a serverless architecture may require significant effort and expertise.

Tagliabue et al. [57] have discussed the design and implementation of Bauplan, a serverless platform aimed at realizing the vision of a Data Lakehouse architecture. Bauplan is built by reusing existing tools and focusing on improving developer experience. The article covers user experience, architecture, and future development plans. However, disadvantage of this method is the reliance on reusing existing tools rather than building a system from scratch. It may also limit the flexibility and customization options available. Additionally, using multiple tools and integrating them into a platform can introduce complexity and compatibility issues.

Zahra et al. [58] have introduced Laminar, a new serverless framework called Laminar that enhances serverless computing by efficiently managing streaming workflows and components. It incorporates semantic code search, code summarization, and code completion using large language models. Laminar aims to simplify the execution of streaming computations, improve data stream management. Although it heavily

relies on large language models for semantic code search and completion. These models also come with computational and resource requirements. Depending on the size and complexity of the codebase, using large language models for real-time code search and completion may introduce latency and performance issues. Additionally, the reliance on external language models may introduce dependencies and compatibility issues with future model versions or updates.

Li et al. [59] have proposed an architecture called Marvel that integrates serverless platforms and Apache Hadoop to enable stateful big data analytics. It addresses the challenges of supporting stateful workloads in serverless platforms by leveraging modern storage technologies such as Intel Optane DC Persistent Memory. The evaluation shows that Marvel significantly reduces the execution time of big data applications compared to current implementations on AWS Lambda. But it relies on specific hardware technology, Intel Optane DC Persistent Memory (PMEM). Also, HDFS storage integration with PMEM support is more complex to configure and manage compared to standard HDFS.

Spenger et al. [60] have introduced Portals, a framework for multi-dataflow stateful serverless applications. Portals enable the dynamic composition of dataflow pipelines and provide services for inter-dataflow communication. Portals supports decentralized runtime execution on both cloud and edge devices, offering end-to-end exactly-once processing guarantees. But disadvantage of this method is the complexity of managing and coordinating multiple dataflow pipelines and their interactions. Additionally, the scalability challenges may arise when dealing with a large number of interconnected dataflow pipelines.

Lei et al. [61] have proposed a method called asynchronous state replication pipelines (ASRP) to accelerate serverless workflows. The ASRP approach is based on delivering changes of differentiable data types (DDT) in real-time, enabling downstream functions to consume the objects without waiting for upstream functions to finish. The authors implemented their approach in a framework called Chitu, compared it with other serverless workflow frameworks, and evaluated it with different cases, showing improvements in data transmission and end-to-end application speed. Although may introduce additional complexity and overhead. Implementing asynchronous state replication pipelines and managing the continuous delivery of changes in real-time can require careful design and coordination.

Sampé et al. [62] have introduced IBM-PyWren, a serverless framework designed for data analytics on IBM Cloud. The framework extends the functionality of PyWren and enables users to run MapReduce jobs, perform data discovery and partitioning, and achieve dynamic function composability. However, this method is the relies on specific cloud platforms, such as IBM Cloud Functions and IBM Cloud Object Storage. This may limit the portability and flexibility of the framework, as it may not be easily adaptable to other cloud providers or environments. Users who prefer or are locked into other cloud platforms may not be able to leverage the features and benefits of IBM-PyWren.

Mahling et al. [63] have presented BabelMR, a framework for serverless MapReduce data processing. BabelMR allows arbitrary containerized applications to use the MapReduce programming model on serverless cloud infrastructure. It simplifies the

development process and integrates efficiently with serverless storage systems. The evaluation shows that BabelMR reduces the entry hurdle to analyzing data in a distributed serverless environment and performs competitively with other serverless MapReduce systems. However, this method relies on serverless cloud infrastructure. While serverless computing offers benefits such as automatic scaling and reduced operational overhead, it also introduces dependencies on cloud providers and their services. Additionally, the cost structure of serverless computing can be complex and may result in costs for users.

Wawrzoniak et al. [64] have presented a system called Boxer that enables direct function-to-function communication in serverless platforms. It addresses the limitations of existing serverless platforms, such as short-lived functions, lack of communication between functions, and limited caching options. They demonstrate that Boxer improves the efficiency of data processing on serverless platforms, resulting in faster query execution and cost reduction. But this method relies on TCP hole-punching techniques, which may introduce additional complexity and security risks. TCP hole-punching involves bypassing network constraints to establish direct communication between functions.

Sampé et al. [65] have presented a data-driven serverless computing middleware called Zion for object storage. It argues that traditional active storage techniques are not well-suited for cloud storage due to lack of elasticity and resource contention. Zion aims to provide painless scalability and simplify the development of disaggregated computing tasks by allowing users to create small, stateless functions that intercept and operate on data flows in a scalable manner without the need for server or runtime environment management. But it may introduce additional overhead due to the interception and processing of data flows by the serverless functions. While the article mentions that Zion has minimal overhead, to consider that there will always be some level of computational cost associated with intercepting and processing data, which could impact overall system performance.

Giménez-Alventosa et al. [66] have introduced a framework called MARLA (MApReduce on LAmbda) that enables the execution of MapReduce jobs on AWS Lambda, a serverless computing platform. The framework is implemented in Python and utilizes Amazon S3 as the storage backend. The article also presents a performance assessment of AWS Lambda, highlighting its suitability for general-purpose applications but noting performance fluctuations that may hinder its adoption for tightly coupled computing jobs. Although disadvantage of this method is the inhomogeneous performance behavior of AWS Lambda identified in the performance assessment. This variability in performance can affect the timeliness of tightly coupled jobs, such as MapReduce applications, which rely on consistent and predictable execution times. This limitation may limit the applicability of AWS Lambda for certain types of computing workloads.

Wawrzoniak et al. [67] have discussed the concept of ephemeral per-query engines (EPQE) for serverless analytics. It challenges the traditional approach of using pre-configured, long-running query engines and proposes dynamically instantiating a data processing engine for each query using Function-as-a-Service (FaaS) platforms. The goal is to optimize engine selection and configuration on a per-query basis, providing flexibility, cost efficiency, and improved performance in data processing. Although ensuring data security and isolation in a dynamic environment with multiple engines running might

require additional considerations. and FaaS platforms might have resource constraints or limitations unsuitable for complex data processing tasks. Also, adapting existing data processing engines for dynamic deployment on FaaS platforms could be complexity.

Bhat et al. [68] have discussed the evaluation of serverless architecture for big data enterprise applications. It explores the use of serverless computing environments, such as AWS Lambda, for large-scale data processing. The paper highlights the benefits of serverless architecture, including better resource utilization, lower costs, and simplified infrastructure management. But disadvantage of this method is the learning curve and complexity. developers new to these platforms may face a steep learning curve to get started. Also, serverless functions are often relying on a specific cloud provider's serverless platform can create vendor lock-in, making it difficult to switch providers in the future. Data processed in serverless environments might be stored and processed across servers managed by the cloud provider, which could raise security concerns.

Bebortta et al. [69] have discussed the use of serverless computing frameworks, such as Amazon Web Services Lambda, Google Cloud Functions, and Microsoft Azure Functions, for managing geospatial big data. They address the limitations of existing systems in terms of reliability, scalability, and computational costs. The proposed framework aims to improve the performance of geospatial data processing and includes case studies using mineral resources data and household prediction data. Although, it may rely heavily on cloud computing infrastructure and services, which can introduce dependencies and issues related to data security. Also, may require a learning curve for developers and users.

Zhang et al. [70] proposes a system called CEVAS (Cloud-Edge collaborative Video Analytics with fine-grained serverless pipelines) that aims to address the challenges of cloud-edge collaborative online video analytics. CEVAS leverages serverless computing to achieve fine-grained resource partitioning and adaptive workload management between the cloud and edge, resulting in reduced costs, improved analysis throughput, and real-time responses to dynamic input workloads. However, this method is rely on serverless computing. It may introduce additional complexity in terms of deployment, monitoring, and debugging compared to traditional computing approaches. Additionally, serverless platforms may have limitations on resources and execution time.

Summary and discussion

These papers discuss various aspects of serverless computing and its application in different domains including performance evaluation, modeling data pipelines, privacy-sensitive data sharing, video analytics, scientific data processing, and more. While serverless computing offers benefits such as cost efficiency, scalability, and reduced management overhead, there are also challenges and considerations to address.

Challenges include complexity in managing serverless functions, potential performance variability, compatibility issues with existing workflows and applications, reliance on specific cloud providers, and security concerns. Additionally, some methods introduce additional complexity or dependencies, such as using large language models or specific hardware technologies.

Table 6 Comparison of framework-based data pipeline approaches

Article	Main idea	Advantage	Disadvantage
[52]	• Evaluating of two serverless data pipeline approaches, NiFi and MQTT, in fog computing environments	• Flexibility • Cost-effectiveness • Event-driven processing	• CPU usage • Complexity
[53]	• Extension to the TOSCA standard called TOSCAdata, which focuses on modeling data pipeline-based cloud applications	• Easy migration • Scalable • Reusable • Deployable • Reducing data lock-in	• Complexity
[13]	• Lambada • Lambada, a serverless distributed data processing framework designed for data analytics	• Cost advantage • Performance • Elasticity • Operational simplicity	• Restricted network connectivity • Limited running time
[54]	• A novel approach for sharing privacy-sensitive data	• Flexibility • Efficiency • Scalability	• Vendor lock-in • Complexity
[55]	• Llama • Llama, a heterogeneous and serverless framework for auto-tuning video analytics pipelines	• Latency reduction • Cost efficiency	• Complexity • Overhead • Cold start • Dependency on serverless environment (vendor lock-in)
[56]	• Application of Function-as-a-Service (FaaS) or serverless computing in the context of scientific data processing, focusing on SKA, a large radio telescope project	• Scalability • Cost-effectiveness • Abstraction and ease of use	• Complexity • Security
[57]	• Design and implementation of Bauplan, a serverless platform aimed at realizing the vision of a Data Lakehouse architecture	• Flexibility • Reproducibility and versioning • Separation of storage and compute • Full auditability	• Complexity • Dependency on external projects
[58]	• Laminar • A new serverless framework called Laminar that enhances serverless computing by efficiently managing streaming workflows and components	• Efficient streaming workflows • Stateful computations • Easy deployment and development	• Dependency on Dispel4py • Limitations or compatibility issues with workflows
[59]	• Marvel • Addressing the challenges posed by the stateless nature of serverless platforms when supporting stateful I/O intensive workloads in big data applications	• Improved I/O throughput • Better performance • Reduce execution time	• Complexity • Relies on specific hardware
[60]	• Portals • Portals, a framework for multi-dataflow stateful serverless applications	• Flexible composition • Decentralized Execution	• Complexity • Low scalability • Overhead
[61]	• A method called asynchronous state replication pipelines (ASRP) to accelerate serverless workflows	• Speed improvement • Improve data transmission	• Complexity • Overhead
[62]	• IBM-PyWren • IBM-PyWren, a serverless framework designed for data analytics on IBM Cloud	• Automatic data discovery and partitioning • Dynamic function composability • Performance improvement • Democratization of massive-scale data parallelism	• Limited portability • Limited flexibility • Limited to IBM Cloud

Table 6 (continued)

Article	Main idea	Advantage	Disadvantage
[63]	• BabelMR, a framework for serverless MapReduce data processing	• Simple development • Language flexibility • Efficient integration	• Dependency on serverless cloud infrastructure (vendor lock-in) • Learning curve • Cost
[64]	• A system called Boxer that enables direct function-to-function communication in serverless platforms	• Performance improvement • Low cost • Speedup queries	• Complexity • Security risks • Dependency on TCP hole-punching techniques • Limited caching possibilities
[65]	• Data-driven serverless computing middleware called Zion for object storage	• Scalability • Simplify the development • Resource contention	• Limitations of active storage • Overhead
[66]	• Framework called MARLA (MApReduce on LAmbda) that enables the execution of MapReduce jobs on AWS Lambda, a serverless computing platform	• Highly scalable • Cost-effective (pay-per-use) • High throughput	• Inhomogeneous performance behavior of AWS Lambd
[67]	• Ephemeral per-query engines (EPQE) for serverless analytics	• Flexibility • Cost effective • Improve performance	• FaaS platform limitations • Security and isolation • Complexity
[68]	• Evaluation of serverless architecture for big data enterprise applications	• Resource utilization • Lower costs • Simplified infrastructure management	• Learning curve • Complexity • Vendor lock-in • Security
[69]	• Use of serverless computing frameworks, for managing geospatial big data	• Scalability • Cost-effectiveness • Reduced latency • Versatility	• Learning curve • Limited execution time • Vendor lock-in
[70]	• System called CEVAS that aims to address the challenges of cloud-edge collaborative online video analytics	• Cost reduction • Real-time responses • Adaptability • Scalability	• Vendor lock-in • Complexity • Limitation on resource • Cold start delay

They highlight the need for careful consideration and evaluation when adopting serverless architectures, taking into account factors such as performance, scalability, complexity, and compatibility with existing systems.

Finally, Tables 6 and 7 provide a complete comparison of the main idea, advantages, and disadvantage, case study, performance metric, technique used, evaluation tool, for each paper.

Discussion

This section provides a coherent examination and assessment of data pipeline methodologies within serverless computing. The analytical insights and evaluations with respect to the technical questions (TQ) of "Research methodology" section are presented:

- TQ1: In serverless computing, what taxonomy is employed in data pipeline approaches?

Table 7 A side-by-side comparison of framework-based data pipeline approaches

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[52]	• Use of Apache NiFi and MQTT as techniques for designing and implementing serverless data pipelines	• Execution time • Memory usage • CPU usage	• Apache NiFi • MQTT • Python • OpenFaaS	• Evaluating NiFi and MQTT based serverless data pipelines in fog computing environments • Object detection
[53]	• TOSCA • TOSCAdata, to model data pipeline-based cloud applications	• Handle the flow and processing of data • Scalable	• AWS • Azure • Google cloud	• A web-based cloud application in the context of tourism promotion, called Viarota
[13]	• Data-parallel query plans	• Invocation time • Invocation rate • Performance • Cost	• AWS • Amazon S3 • Amazon SQS • Python	• Implementation of a serverless cloud infrastructure for interactive data analytics on cold data
[54]	• Combining serverless computing advantages with data mesh	• Flexible • Scalability	• NA	• Data sharing for medical studies within a federation of hospitals
[55]	• Cost-based optimizer • Heterogeneous hardware support • Dynamic computation of per-invocation latency targets	• Latency • Cost	• AWS • Google cloud • Python • C++ • JSON	• Development and evaluation of Llama, a heterogeneous and serverless framework for auto-tuning video analytics pipelines • Face and car recognition
[56]	• Using FaaS as a serverless computing paradigm	• Scalability • Cost	• OpenFaaS • Python • Wsclean	• Application of serverless computing, specifically FaaS, within the context of the square kilometre array (SKA), a radio telescope project
[57]	• Reusing existing tools and investing in differentiating features	• Flexibility • User experience	• AWS • OpenWhisk • Python	• Bauplan, a serverless platform designed to implement the Data Lakehouse (DLH) vision
[58]	• Using dispel4py support for parallelization and abstract workflow descriptions in Python to build stream processing pipelines • Deep learning code search	• Latency • Code search accuracy • Speed • Resource utilization	• OpenFaaS • Python	• Laminar, a new serverless stream-based framework with semantic code search and code completion
[59]	• Integration of serverless platforms Apache OpenWhisk with Apache Hadoop for running stateful big data applications	• Execution time • Throughput	• AWS • Apache OpenWhisk • Apache Hadoop • S3	• "Marvel" that explores the feasibility of performing stateful big data analytics on serverless platforms using Intel Optane DC Persistent Memory (PMEM) as a storage technology

Table 7 (continued)

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[60]	• Portals • State-full dataflow composition framework for serverless applications • Using of asynchronous two-phase • Snapshotting	• Flexibility	• SQL • Java script	• Use of Portals for composing complex serverless multi-dataflow applications
[61]	• ASRP • As a method to accelerate serverless workflows by continuously delivering changes of differentiable data types (DDT) objects in real-time	• Speed • Data transmission acceleration	• OpenFaaS • AWS • EC2 • Rust • C++ • Python	• Serverless workflows and proposes a novel framework called Chitu
[62]	• IBM-PyWren as an extension of PyWren	• Speed (execution time)	• AWS • Python • IBM Cloud Functions	• Design and implementation of IBM-PyWren, which is an open-source serverless framework for data analytics
[63]	• Using of MapReduce programming model and leverages serverless cloud infrastructure, for executing containerized applications	• Flexibility • Easier to use	• AWS • Python • C# • S3	• BabelMR, a polyglot framework for serverless MapReduce
[64]	• Using TCP hole-punching techniques related to peer-to-peer systems to circumvent the network constraints of current serverless offerings	• Speed • Throughout • Latency	• AWS	• Implementation of Boxer, a system enabling direct function-to-function communication in existing public cloud infrastructure
[65]	• Zion as a data-driven serverless computing middleware for object storage • Creating small, stateless functions	• Execution time • Overhead	• OpenStack Swift • Java	• Design and implementation of a data-driven serverless computing middleware called Zion for object storage
[66]	• A Python-based framework called MARLA for running MapReduce jobs on AWS Lambda	• Execution time • Throughput • CPU performance	• AWS • Amazon S3 • Python	• Suitability of AWS Lambda as a platform for executing MapReduce jobs
[67]	• EPQE • Selecting the optimal engine and configuration on a per-query basis, eliminating the inefficiencies of using all-purpose configurations and resource overprovisioning	• Execution time • Cost • Resource utilization	• AWS • EC2/ECS • Apache Drill • SQL	• Concept of ephemeral per-query engines (EPQE) for serverless analytics

Table 7 (continued)

Article	Utilized technique	Performance metric	Evaluation tools	Case study
[68]	• Using serverless computing environments for performing MapReduce processing	• Resource utilization • Cost	• AWS • Python • Amazon S3 • SQL	• Evaluating serverless architecture for big data enterprise applications
[69]	• Applicability of serverless frameworks, specifically Amazon Web Services (AWS) Lambda, Google Cloud Functions, and Microsoft Azure Functions, for the management of geospatial big data	• Reliability • Scalability • Speed • Security • Cost	• AWS • Python • Jupyter	• Mineral resources data system (MRDS) dataset, which represents the worldwide density of mineral resources in a country-wise fashion • Fairfax forecast households dataset, which predicts parcel-level household data for 30 consecutive years
[70]	• CEVAS • Utilization the serverless computing paradigm which enables fine-grained resource partitioning with minimum dependency	• Overhead • Throughput • Cost	• AWS • Python	• Design and implementation of a cloud-edge collaborative online video analytics system called CEVAS

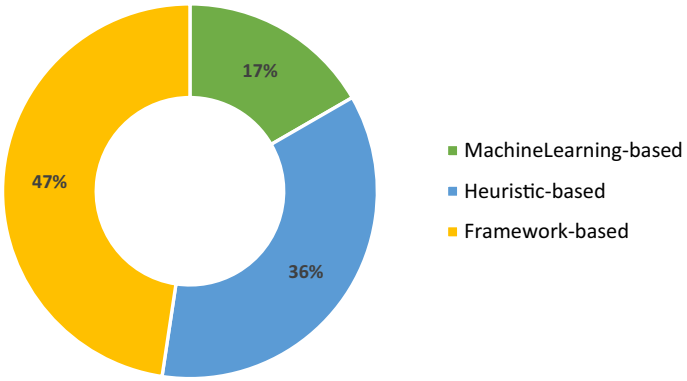

Fig. 6 Classification of data pipeline approaches in serverless computing

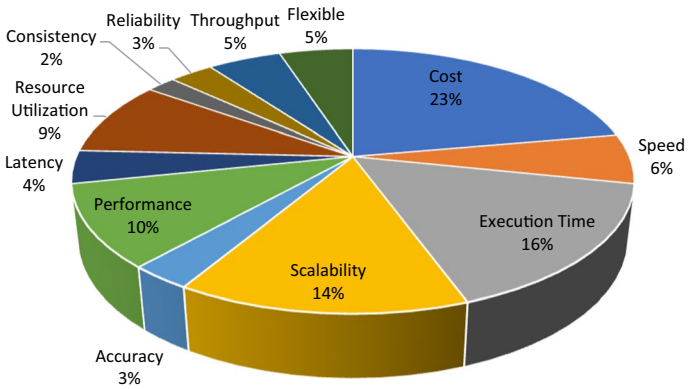

Fig. 7 Performance metrics of the data pipeline in serverless computing

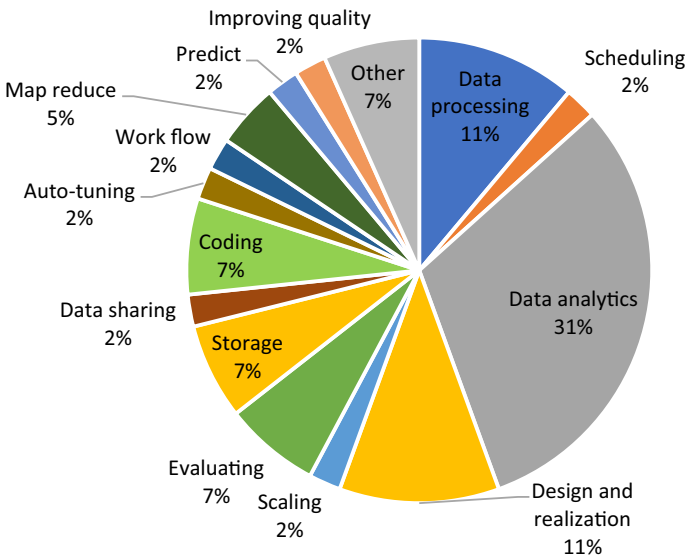

Fig. 8 Some case studies related to the data pipeline in serverless computing

According to the proposed taxonomy Fig. 4, three approaches are examined based on the classification presented: machine learning-based approaches, heuristic techniques-based approaches, and framework-based approaches. Figure 6 demonstrates that the majority of the selected research articles are associated with the framework-based approach, covering 47% of the articles.

- TQ2: What performance metrics are commonly used for data pipeline approaches in serverless computing?
  Figure 7 shows various performance metrics for data pipeline approaches within serverless computing. Since some research papers had multiple objectives, certain criteria may overlap in the papers. Analysis of these metrics illustrate that cost 23% and execution time 16% are the most commonly examined aspects in data pipeline approaches within serverless computing. Scalability with 14% follows behind. Consequently, features like overhead, data transmission acceleration, and invocation time receive less attention due to their limited coverage, while open challenges remain a in data pipeline approaches within serverless computing.
- TQ3: Which case studies are utilized in data pipeline approaches within serverless computing?
  Figure 8 shows the case studies used to investigate data pipeline approaches within serverless computing. This collection of case studies consists of tasks such as data analytics, data processing, data storage, data sharing, workflow, evaluation, predict, design and realizing, scaling, map reduce, and improving quality within serverless platforms. The methodologies proposed in these studies serve diverse objectives, ranging from specific to more versatile applications. Consequently, some articles have multiple case studies in their research. In the reviews, data analytics was determined as the most frequently utilized case study, representing 31% of the studies analyzed.

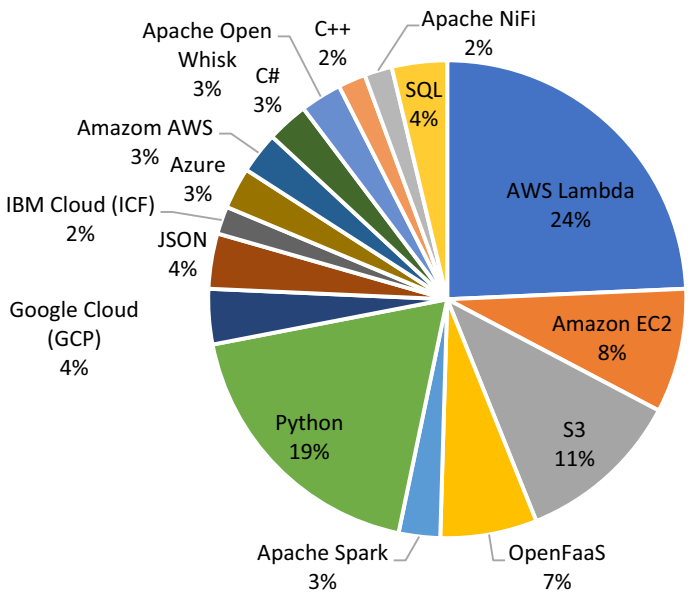

Fig. 9 Evaluation tools the data pipeline in serverless computing

Table 8 Comparison of advantages and disadvantages of data pipeline approaches

Approach	Advantage	Disadvantage
Machine learning-based	• Improving accuracy and efficiency: ML models can learn and adapt to changing data patterns, leading to better data processing and reduced resource utilization • Automated optimization: ML algorithms can automatically identify and optimize pipeline parameters, reducing manual intervention • Scalability and elasticity: ML models can scale to handle large data volumes efficiently and automatically adjust to varying workloads	• Complexity: Building and maintaining ML models can be complex • Development cost: building and maintaining ML models is complex and requires expertise and increases development costs • Require specialized expertise: ML models can be difficult to understand and explain their decision-making process
Heuristic-based	• Simplicity and ease of implementation: heuristics-based pipelines are typically easier to understand and implement compared to ML-based approaches • Predictability and control: heuristics are well-defined rules that provide more control over the steps and results of data processing • Less data dependency: heuristics often require less data to train compared to ML models, making them suitable for smaller data sets	• Limited adaptability: heuristics struggle to adapt to changing data patterns or unexpected scenarios, leading to inaccuracies • Manual maintenance: heuristics must be updated manually as data or requirements change, increasing maintenance costs • Performance inefficient for complex data processing tasks: heuristics may not achieve optimal performance for complex data or tasks
Framework-based	• Integrations: frameworks often provide built-in integrations with various tools and services, simplifying deployment and management • Faster development: frameworks can speed up pipeline development by providing ready-made solutions for common tasks • Reusability and standardization: frameworks provide pre-built components and workflows that enable code reuse and standardization	• Vendor lock-in: Choosing a specific framework can limit flexibility and customization options • Learning curve: frameworks may have their own syntax and complexities to learn. Familiarity with the chosen framework is necessary for development and troubleshooting • Performance overhead: frameworks can introduce additional overhead compared to custom-built pipelines

- TQ4: What evaluation tools are employed for data pipeline approaches in serverless computing?

  Figure 9 shows that 24% of the research papers utilized the AWS Lambda tools. Additionally, certain articles utilized various tools for their proposed models. The ranking of other tools is shown in the chart. Certain evaluation tools were excluded due to limited coverage.
- TQ5: What are the advantage and disadvantage of data pipeline approaches in serverless computing?

  Finally, Table 8 displays the advantages and disadvantages of different data pipeline methods in serverless computing. Based on this comparison and analysis, it can be concluded that each data pipeline approach, heuristic-based, framework-based, or machine learning-based, has set of advantages and disadvantages, and each approach is suitable for a specific task.

  Machine learning-based data pipeline methods can handle complex data processing tasks, such as feature extraction and pattern recognition. And improve the accuracy and predictive capabilities through iterative model modification. Heuristic-based data pipeline methods can handle specific data processing requirements that may

not require complex models. And Faster and more efficient processing as it doesn't involve training and deploying models. Framework-based data pipeline methods can support integration with various data sources and tools for seamless data processing. And offers a wide range of pre-built components and functionalities for data pipeline development.

In the following, we discuss the three approaches in terms of complexity, flexibility, and vendor lock-in, as following:

- Complexity vs. simplicity: Heuristic-based approaches are simpler to implement, while ML-based solutions offer more advanced features and therefore higher complexity.
- Flexibility vs. rigidity: Machine learning-based pipelines are highly adaptable, while heuristic-based approaches are more rigid and dependent on predefined rules.
- Vendor lock-in: Framework-based approaches often limit the data pipeline to a specific vendor or platform, while heuristic and ML-based approaches may provide more vendor unknown solutions.

Each approach has its own advantages and disadvantages, and the selection of the appropriate approach depends on the specific requirements of the use case, such as data volume, processing needs, real-time requirements, and cost considerations. And it should be based on the specific requirements of the use case, such as the complexity of the data, the need for adaptability, the importance of performance and reliability, and the available resources and expertise within the organization.

Open issues and challenges

This section addresses open issues regarding data pipeline approaches and barriers that exist in serverless computing systems. Data pipeline management is very important in serverless computing and can be effectively managed using different approaches such as

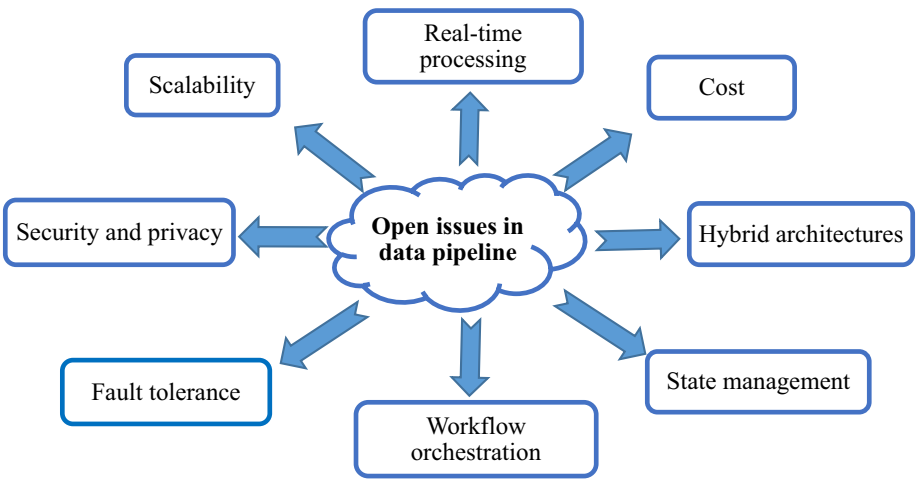

Fig. 10 Open issues of data pipeline

machine learning, heuristics and frameworks. TQ6 outlines open issues and challenges ahead.

- TQ6: What future research directions and open issues exist for data pipeline approaches in serverless computing?

Figure 10 provides a discussion of the challenges and open issues in the realm of data pipelines in serverless computing and explores different aspects.

- Scalability: Serverless computing needs to guarantee function scalability and elasticity [38]. And explore techniques for efficiently scaling data pipelines in serverless computing environments to manage large and complex data sets. This involves investigating auto-scaling mechanisms, load balancing strategies, and resource allocation algorithms to optimize pipeline performance.
- Fault tolerance: Fault mostly occurs when some containers fail. To overcome this challenge, a basic retry mechanism is used [71, 72]. Research can focus on developing mechanisms to handle failures, such as automatic retry mechanisms, error handling strategies, and fault detection and recovery techniques, ensuring robustness and reliability of the pipeline.
- Security and privacy: Investigate methods to ensure data security and privacy in serverless data pipelines. This includes exploring techniques for secure data transfer and storage, encryption methods, access control mechanisms, and compliance with privacy regulations to protect sensitive data throughout the pipeline. Isolation is also a security issue, as functions are executed on a shared platform by many users. Therefore, strong isolation is required [73–75].
- Cost optimization: Cost is a fundamental challenge [76–78]. Cost optimization that's mean Developing approaches to optimize the cost of executing data pipelines in serverless environments. This involves analyzing the cost implications of different pipeline configurations, considering factors such as resource allocation, function sizing, and data transfer costs, to minimize overall expenses while maintaining performance.
- Workflow orchestration: Exploring techniques for managing and orchestrating complex workflows in serverless data pipelines. This includes investigating workflow specification languages, coordination mechanisms, vector machine to predict, dynamic programming and task scheduling algorithms to streamline the execution and coordination of multiple functions within the pipeline [79–81].
- Real-time processing: Investigating techniques to enable real-time data processing in serverless data pipelines. This involves exploring mechanisms for event-driven processing, stream processing, and near real-time analytics, allowing for timely insights and decision-making based on streaming data sources [82, 83].
- Hybrid architectures: Integration of serverless computing with other computing paradigms, such as edge computing or hybrid cloud approaches [84]. This can involve exploring hybrid architectures that leverage the strengths of serverless computing for data processing while considering data locality, latency, and data governance requirements.

- State management: Serverless functions are inherently stateless, but pipelines often require data persistence. Research into efficient state management solutions that integrate seamlessly with serverless architecture is needed.

Conclusions

Serverless computing offers a scalable and cost-effective solution for handling data pipelines, as it eliminates the need for managing and provisioning servers. This paper, provides a taxonomy of data pipeline approaches in the context of serverless computing. Approaches are classified based on architectural features, data processing techniques, and workflow orchestration mechanisms. These methods are divided into three categories: a machine learning-based approach, a heuristic-based approach, and a framework-based approach. Each of these methods have been examined and Its advantages and disadvantages have been highlighted along with key factors affecting their effectiveness. Optimal data pipeline strategy whether it's heuristic-driven, framework-centric, or machine learning-based, carries its own set of pros and cons. The suitability of each method varies depending on the specific use case. Therefore, careful evaluation of the trade-offs between performance, cost, and complexity is essential when choosing a data pipeline strategy. There are several open issues and future directions for investigating in the field of data pipelines in serverless computing. These include exploring techniques to enable real-time data processing in serverless data pipelines, hybrid architectures, ensuring data security and privacy, the challenge of fault tolerance, cost optimization, scalability and state management. The hybrid approaches offer a solution for building real-time, scalable, and cost-efficient serverless data pipelines while addressing issues like fault tolerance and data security. Overall, event-driven architectures and serverless stream processing are emerging areas that can enhance the real-time processing capabilities of data pipelines and managing complex workflows, ensuring data consistency and reliability, and optimizing resource allocation.

Author contributions
Zahra Shojaee Rad, Mostafa Ghobaei-Arani conducted this research. Zahra Shojaee Rad: methodology, software, validation, writing original draft. Mostafa Ghobaei-Arani: investigation, resources, data curation, visualization.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials
The datasets used or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests
We certify that there is no actual or potential conflict of interest in relation to this article.

Received: 1 February 2024 Accepted: 31 May 2024
Published online: 11 June 2024

References

1. Dukic V, Bruno R, Singla A, Alonso G. Photons: Lambdas on a diet. In: Proceedings of the 11th ACM symposium on cloud computing. 2020. p. 45–59.
2. Fuerst A, Sharma P. FaasCache: keeping serverless computing alive with greedy-dual caching. In: Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems. 2021. p. 386–400.
3. Ebrahimi A, Ghobaei-Arani M, Saboohi H (2024) Cold start latency mitigation mechanisms in serverless computing: taxonomy, review, and future directions. J Syst Architect 151:103115. https://doi.org/10.1016/j.sysarc.2024.103115
4. Ustiugov D, Petrov P, Kogias M, Bugnion E, Grot B. Benchmarking, analysis, and optimization of serverless function snapshots. In: Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems. 2021. p. 559–72.
5. Shankar V, Krauth K, Pu Q, Jonas E, Venkataraman S, Stoica I, Recht B, Ragan-Kelley J. Numpywren: serverless linear algebra. arXiv preprint. 2018. arXiv:1810.09679.
6. Werner S, Kuhlenkamp J, Klems M, Müller J, Tai S. Serverless big data processing using matrix multiplication as example. In: 2018 IEEE international conference on Big Data (Big Data). IEEE; 2018. p. 358–65.
7. Aytekin A, Johansson M. Harnessing the power of serverless runtimes for large-scale optimization. arXiv preprint. 2019. arXiv:1901.03161.
8. Carver B, Zhang J, Wang A, Anwar A, Wu P, Cheng Y. LADS: a high-performance framework for serverless parallel computing. In: Proceedings of the ACM symposium on cloud computing, SoCC. 2020.
9. Lee BD, Timony MA, Ruiz P. DNAvisualization. org: a serverless web tool for DNA sequence visualization. Nucleic Acids Res. 2019;47(W1):W20–5.
10. Hung LH, Kumanov D, Niu X, Lloyd W, Yeung KY. Rapid RNA sequencing data analysis using serverless computing. bioRxiv. 2019. https://doi.org/10.1101/576199.
11. Kumanov D, Hung LH, Lloyd W, Yeung KY. Serverless computing provides on-demand high performance computing for biomedical research. arXiv preprint. 2018. arXiv:1807.11659.
12. Ghorbian M, Ghobaei-Arani M, Esmaeili L. A survey on the scheduling mechanisms in serverless computing: a taxonomy, challenges, and trends. Cluster Comput (2024). https://doi.org/10.1007/s10586-023-04264-8.
13. Müller I, Marroquín R, Alonso G. Lambada: interactive data analytics on cold data using serverless cloud infrastructure. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2020. p. 115–30.
14. Cao C, Wang J, Kwok D, Cui F, Zhang Z, Zhao D, Li MJ, Zou Q. webTWAS: a resource for disease candidate susceptibility genes identified by transcriptome-wide association study. Nucleic Acids Res. 2022;50(D1):D1123–30.
15. Salimian M, Ghobaei-Arani M, Shahidinejad A (2021) Toward an autonomic approach for internet of things service placement using gray wolf optimization in the fog computing environment. Softw Pract Exp 51(8):1745–1772. https://doi.org/10.1002/spe.2986.
16. Shojaee Rad Z, Ghobaei-Arani M, Ahsan R. Memory orchestration mechanisms in serverless computing: a taxonomy, review and future directions. Cluster Comput (2024). https://doi.org/10.1007/s10586-023-04251-z.
17. Jiang J, Gan S, Liu Y, Wang F, Alonso G, Klimovic A, Singla A, Wu W, Zhang C. Towards demystifying serverless machine learning training. In: Proceedings of the 2021 international conference on management of data. 2021. p. 857–71
18. Shahidinejad A, Farahbakhsh F, Ghobaei-Arani M et al. Context-aware multi-user offloading in mobile edge computing: a federated learning-based approach. J Grid Computing 19:18 (2021). https://doi.org/10.1007/s10723-021-09559-x.
19. Introduction to dbt. https://docs.getdbt.com/docs/introduction. (cit. on pp. 8, 15).
20. Ebert C, Gallardo G, Hernantes J, Serrano N. DevOps. IEEE Softw. 2016;33(3):94–100.
21. Arachchi SAIBS, Perera I. Continuous integration and continuous delivery pipeline automation for agile software project management. In: 2018 Moratuwa engineering research conference (MERCon). IEEE; 2018. p. 156–61.
22. Lloyd W, Ramesh S, Chinthalapati S, Ly L, Pallickara S. Serverless computing: an investigation of factors influencing microservice performance. In: 2018 IEEE international conference on cloud engineering (IC2E). IEEE; 2018. p. 159–69.
23. Werner S, Tai S. Application-platform co-design for serverless data processing. In: Service-oriented computing: 19th international conference, ICSOC 2021, virtual event, November 22–25, 2021, proceedings 19. Springer International Publishing; 2021. p. 627–40.
24. García-López P, Sánchez-Artigas M, Shillaker S, Pietzuch P, Breitgand D, Vernik G, Sutra P, Tarrant T, Juan-Ferrer A, París G. Trade-offs and challenges of serverless data analytics. In: Technologies and applications for big data value. Cham: Springer International Publishing; 2021. p. 41–61.
25. Wu Y, Dinh TT, Hu G, Zhang M, Chee YM, Ooi BC. Serverless data science-are we there yet? A case study of model serving. In: Proceedings of the 2022 international conference on management of data. 2022. p. 1866–75.
26. Cordingly R, Yu H, Hoang V, Perez D, Foster D, Sadeghi Z, Hatchett R, Lloyd WJ. Implications of programming language selection for serverless data processing pipelines. In: 2020 IEEE Intl Conf on dependable, autonomic and secure computing, Intl Conf on pervasive intelligence and computing, Intl Conf on cloud and big data computing, Intl Conf on cyber science and technology congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE; 2020. p. 704–11.
27. Grzesik P, Augustyn DR, Wyciślik Ł, Mrozek D. Serverless computing in omics data analysis and integration. Brief Bioinform. 2022;23(1): bbab349.
28. Patel D, Lin S, Kalagnanam J. DSServe-data science using serverless. In: 2022 IEEE international conference on big data (big data). IEEE; 2022. p. 2343–5.
29. Bezverbnyi IA, Shyshkina MP. Serverless computing for data processing in open learning and research environments. In: CEUR workshop proceedings. 2023. p. 229–36.
30. Alonso G, Klimovic A, Kuchler T, Wawrzoniak M. Rethinking serverless computing: from the programming model to the platform design. In: Joint proceedings of workshops at the 49th international conference on very large data bases (VLDB 2023); 2023.

31. Nesen A, Bhargava B. Towards situational awareness with multimodal streaming data fusion: serverless computing approach. In: Proceedings of the international workshop on big data in emergent distributed environments. 2021. p. 1–6.
32. Rausch T, Rashed A, Dustdar S. Optimized container scheduling for data-intensive serverless edge computing. Futur Gener Comput Syst. 2021;114:259–71.
33. León-Sandoval E, Zareei M, Barbosa-Santillán LI, Morales LE. Using big data and serverless architecture to follow the emotional response to the COVID-19 pandemic in Mexico. In: Latin American high performance computing conference. Cham: Springer International Publishing; 2022. p. 145–59.
34. Mohapatra AD, Oh K. Smartpick: workload prediction for serverless-enabled scalable data analytics systems. In: Proceedings of the 24th international middleware conference on ZZZ. 2023. p. 29–42.
35. Paraskevoulakou E, Kyriazis D. ML-FaaS: towards exploiting the serverless paradigm to facilitate machine learning functions as a service. IEEE Trans Netw Serv Manag. 2023. https://doi.org/10.1109/TNSM.2023.3239672.
36. Rahman MM, Hasan MH. Big data analytics using serverless computing—a personalized recommendation system case study. Int J Sci Technol Res. 2020;9(9):288–293.
37. Bhattacharjee A, Barve Y, Khare S, Bao S, Gokhale A, Damiano T. Stratum: a serverless framework for the lifecycle management of machine learning-based data analytics tasks. In: 2019 USENIX conference on operational machine learning (OpML 19). 2019. p. 59–61.
38. Enes J, Expósito RR, Touriño J. Real-time resource scaling platform for big data workloads on serverless environments. Futur Gener Comput Syst. 2020;105:361–79.
39. Kuhlenkamp J, Werner S, Borges MC, El Tal K, Tai S. An evaluation of faas platforms as a foundation for serverless big data processing. In: Proceedings of the 12th IEEE/ACM international conference on utility and cloud computing. 2019. p. 1–9.
40. Poojara SR, Dehury CK, Jakovits P, Srirama SN. Serverless data pipeline approaches for IoT data in fog and cloud computing. Futur Gener Comput Syst. 2022;130:91–105.
41. Toader L, Uta A, Musaafir A, Iosup A. Graphless: toward serverless graph processing. In: 2019 18th international symposium on parallel and distributed computing (ISPDC). IEEE; 2019. p. 66–73.
42. Bian H, Sha T, Ailamaki A. Using cloud functions as accelerator for elastic data analytics. Proc ACM Manag Data. 2023;1(2):1–27.
43. Jarachanthan J, Chen L, Xu F. ACTS: autonomous cost-efficient task orchestration for serverless analytics. In: 2023 IEEE/ACM 31st international symposium on quality of service (IWQoS). IEEE; 2023. p. 1–10.
44. Pogiatzis A, Samakovitis G. An event-driven serverless ETL pipeline on AWS. Appl Sci. 2020;11(1):191.
45. Bharti U, Bajaj D, Goel A, Gupta SC. A novel design approach exploiting data parallelism in serverless infrastructure. In: Advances in computing and network communications: proceedings of CoCoNet 2020, vol. 1. Springer Singapore; 2021. p. 247–60.
46. Sanchez-Gallegos DD, Carrizales-Espinoza D, Gonzalez-Compean JL, Carretero J. eScience serverless data storage services in the edge-fog-cloud continuum. In: 2023 IEEE 19th international conference on e-science (e-science). IEEE; 2023. p. 1–4.
47. Mrozek D, Stępień K, Grzesik P, Małysiak-Mrozek B. A large-scale and serverless computational approach for improving quality of NGS data supporting big multi-omics data analyses. Front Genet. 2021;12: 699280.
48. Pakdil ME, Çelik RN. Serverless geospatial data processing workflow system design. ISPRS Int J Geo-Inf. 2022;11(1):20.
49. Moina-Rivera W, Garcia-Pineda M, Claver JM, Gutiérrez-Aguado J. Event-driven serverless pipelines for video coding and quality metrics. J Grid Comput. 2023;21(2):20.
50. Spiegelberg L, Kraska T, Schwarzkopf M. Hyperspecialized compilation for serverless data analytics. 2023.
51. Cinaglia P, Cannataro M. A method for modelling and executing customized pipelines in serverless computing. In: 2023 IEEE international conference on bioinformatics and biomedicine (BIBM). IEEE; 2023. p. 3453–8.
52. Mirampalli S, Wankar R, Srirama SN. Evaluating NiFi and MQTT based serverless data pipelines in fog computing environments. Futur Gener Comput Syst. 2024;150:341–53.
53. Dehury CK, Jakovits P, Srirama SN, Giotis G, Garg G. TOSCAdata: modeling data pipeline applications in TOSCA. J Syst Softw. 2022;186: 111164.
54. Sedlak B, Pujol VC, Donta PK, Werner S, Wolf K, Falconi M, Pallas F, Dustdar S, Tai S, Plebani P. Towards serverless data exchange within federations. In: Symposium and summer school on service-oriented computing. Cham: Springer Nature Switzerland; 2023. p. 144–53.
55. Romero F, Zhao M, Yadwadkar NJ, Kozyrakis C. Llama: a heterogeneous & serverless framework for auto-tuning video analytics pipelines. In: Proceedings of the ACM symposium on cloud computing. 2021. p. 1–17.
56. Ríos-Monje C, Parra-Royón M, Moldón J, Sánchez-Expósito S, Garrido J, Darriba L, Mendoza M, Sánchez J, Verdes-Montenegro L, Salgado J. An approach to provide serverless scientific pipelines within the context of SKA. arXiv preprint. 2023. arXiv:2306.09728.
57. Tagliabue J, Greco C, Bigon L. Building a serverless Data Lakehouse from spare parts. arXiv preprint. 2023. arXiv:2308.05368.
58. Zahra Z, Li Z, Filgueira R. Laminar: a new serverless stream-based framework with semantic code search and code completion. In: Proceedings of the SC'23 workshops of the international conference on high performance computing, network, storage, and analysis. 2023. p. 2009–20.
59. Li Y, Assogba K, Tripathy A, Arif M, Rafique MM, Butt AR, Nikolopoulos D. Towards persistent memory based stateful serverless computing for big data applications. arXiv preprint. 2023. arXiv:2309.01662.
60. Spenger J, Huang C, Haller P, Carbone P. Portals: a showcase of multi-dataflow stateful serverless. Proc VLDB Endowment. 2023;16(12):4054–7.
61. Lei Z, Shi X, Lv C, Yu X, Zhao X. Chitu: accelerating serverless workflows with asynchronous state replication pipelines. In: Proceedings of the 2023 ACM symposium on cloud computing. 2023. p. 597–610.
62. Sampé J, Vernik G, Sánchez-Artigas M, García-López P. Serverless data analytics in the IBM cloud. In: Proceedings of the 19th international middleware conference industry. 2018. p. 1–8.
63. Mahling F, Rößler P, Bodner T, Rabl T. BabelMR: a polyglot framework for serverless mapreduce. 2023.

64. Wawrzoniak M, Müller I, Fraga Barcelos Paulus Bruno R, Alonso G. Boxer: data analytics on network-enabled serverless platforms. In: 11th annual conference on innovative data systems research (CIDR 2021). 2021.
65. Sampé J, Sánchez-Artigas M, García-López P, París G. Data-driven serverless functions for object storage. In: Proceedings of the 18th ACM/IFIP/USENIX middleware conference. 2017. p. 121–33.
66. Giménez-Alventosa V, Moltó G, Caballer M. A framework and a performance assessment for serverless MapReduce on AWS Lambda. Futur Gener Comput Syst. 2019;97:259–74.
67. Wawrzoniak M, Fraga Barcelos Paulus Bruno R, Klimovic A, Alonso G. Ephemeral per-query engines for serverless analytics. In: Oint workshops at 49th international conference on very large data bases (VLDBW'23)—workshop on serverless data analytics (SDA'23). 2023.
68. Bhat A, Park H, Roy M. Evaluating serverless architecture for big data enterprise applications. In: 2021 IEEE/ACM 8th international conference on big data computing, applications and technologies (BDCAT'21). 2021. p. 1–8.
69. Bebortta S, Das SK, Kandpal M, Barik RK, Dubey H. Geospatial serverless computing: architectures, tools and future directions. ISPRS Int J Geo-Inf. 2020;9(5):311.
70. Zhang M, Wang F, Zhu Y, Liu J, Wang Z. Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines. In: Proceedings of the 12th ACM multimedia systems conference. 2021. p. 80–93.
71. Palade A, Kazmi A, Clarke S. An evaluation of open source serverless computing frameworks support at the edge. In: 2019 IEEE world congress on services (SERVICES), vol. 2642. IEEE; 2019. p. 206–211
72. Yussupov V, Breitenbücher U, Leymann F, Wurster M. A systematic mapping study on engineering function-as-a-service platforms and tools. In: Proceedings of the 12th IEEE/ACM international conference on utility and cloud computing. 2019. p. 229–40.
73. Wu M, Mi Z, Xia Y. A survey on serverless computing and its implications for jointcloud computing. In: 2020 IEEE international conference on joint cloud computing. IEEE; 2020. p. 94–101.
74. Benedict S. Serverless blockchain-enabled architecture for IoT societal applications. IEEE Trans Comput Soc Syst. 2020;7(5):1146–58.
75. Tan B, Liu H, Rao J, Liao X, Jin H, Zhang Y. Towards lightweight serverless computing via unikernel as a function. In: 2020 IEEE/ACM 28th international symposium on quality of service (IWQoS). IEEE; 2020. p. 1–10.
76. Eismann S, Grohmann J, Van Eyk E, Herbst N, Kounev S. Predicting the costs of serverless workflows. In: Proceedings of the ACM/SPEC international conference on performance engineering. 2020. p. 265–76.
77. Grogan J, Mulready C, McDermott J, Urbanavicius M, Yilmaz M, Abgaz Y, McCarren A, et al. A multivocal literature review of function-as-a-service (faas) infrastructures and implications for software developers. In: Systems, software and services process improvement: 27th European conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, proceedings 27. Springer International Publishing; 2020. p. 58–75.
78. Reuter A, Back T, Andrikopoulos V. Cost efficiency under mixed serverless and serverful deployments. In: 2020 46th Euromicro conference on software engineering and advanced applications (SEAA). IEEE; 2020. p. 242–5.
79. Zhao H, Zhao N, Zong G, Zhao X, Xu N. Sliding-mode surface-based approximate optimal control for nonlinear multiplayer Stackelberg-Nash games via adaptive dynamic programming. Commun Nonlinear Sci Numer Simul. 2024;132: 107928.
80. Liu S, Wang H, Liu Y, Ning Xu, Zhao X. Sliding-mode surface-based adaptive optimal nonzero-sum games for saturated nonlinear multi-player systems with identifier-critic networks. Neurocomputing. 2024;584: 127575.
81. Zhang H, Zou Q, Ying Ju, Song C, Chen D. Distance-based support vector machine to predict DNA N6-methyladenine modification. Curr Bioinform. 2022;17(5):473–82.
82. Wu X, Ding S, Xu N, Niu B, Zhao X. Periodic event-triggered bipartite containment control for nonlinear multi-agent systems with iuput delay. Int J Syst Sci. 2024. https://doi.org/10.1080/00207721.2024.2328780.
83. Liu S, Niu B, Xu N, Zhao X. Zero-sum game-based decentralized optimal control for saturated nonlinear interconnected systems via a data and event driven approach. IEEE Syst J. 2024. https://doi.org/10.1109/JSYST.2024.3350771.
84. Huang S, Zong G, Ning Xu, Wang H, Zhao X. Adaptive dynamic surface control of MIMO nonlinear systems: a hybrid event triggering mechanism. Int J Adapt Control Signal Process. 2024;38(2):437–54.

Publisher's Note