KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express xxx (xxxx) xxx

www.elsevier.com/locate/icte

The role of microservice approach in edge computing: Opportunities, challenges, and research directions

Md. Delowar Hossain[a,b], Tangina Sultana[a,c], Sharmen Akhter[a], Md Imtiaz Hossain[a], Ngo Thien Thu[a], Luan N.T. Huynh[d], Ga-Won Lee[a], Eui-Nam Huh[a,*]

[a] *Department of Computer Science and Engineering, Kyung Hee University, Global Campus, Yongin-si 17104, Republic of Korea*
[b] *Department of Computer Science and Engineering, Hajee Mohammad Danesh Science & Technology University, Dinajpur 5200, Bangladesh*
[c] *Department of Electronics and Communication Engineering, Hajee Mohammad Danesh Science & Technology University, Dinajpur 5200, Bangladesh*
[d] *Institute of Engineering Technology, Thu Dau Mot University, Binh Duong, Viet Nam*

Received 2 March 2023; received in revised form 25 May 2023; accepted 17 June 2023
Available online xxxx

Abstract

Edge computing has emerged as a promising computing paradigm that enables real-time data processing and analysis closer to the data source and boosts decision-making applications in a safe manner. On the other hand, the microservice is a new type of architecture that can be dynamically deployed, migrating across edge clouds on demand. Therefore, the combination of these two technologies can provide numerous benefits, including improved performance, reduced latency, and better resource utilization. In this paper, we present a thorough analysis of state-of-the-art research on the use of microservices in edge computing environments. We take into consideration several distinct microservice research directions, including coordination, orchestration, repositories, scheduling, autoscaling, deployment, resource management, and different security issues. Furthermore, we explore the potential applications of microservices in edge computing across various domains. Finally, the unsolved research issues and future directions of emerging trends in this area are also discussed.

Keywords: Edge computing; Microservices; Monolithic architectures; Microservice security; AI

Contents

* Corresponding author.
E-mail addresses: delowar@khu.ac.kr (M.D. Hossain), tangina@khu.ac.kr (T. Sultana), sharmen@khu.ac.kr (S. Akhter), hossain.imtiaz@khu.ac.kr (M.I. Hossain), thu.ngo@khu.ac.kr (N.T. Thu), luanhnt@tdmu.edu.vn (L.N.T. Huynh), gawon@khu.ac.kr (G.-W. Lee), johnhuh@khu.ac.kr (E.-N. Huh).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2023.06.006

Please cite this article as: M.D. Hossain, T. Sultana, S. Akhter et al., The role of microservice approach in edge computing: Opportunities, challenges, and research directions, ICT Express (2023), https://doi.org/10.1016/j.icte.2023.06.006.

1. Introduction

With the evolution of information and communications technologies, the world has lately seen the birth of significant technical advancements in the fields of smart living and data transmission. However, smart navigation, smart surveillance, smart actuators, smart sensors, information entertainment, networking technologies, and low-power circuits have all grown significantly during the last decade [1,2]. As a result, data transmission across the internetwork backhaul has greatly increased. The current cloud infrastructure has undoubtedly been exploited in order to effectively handle all of these demands in a timely manner [3–5]. Due to data processing and transmission, cloud computing will lead to data leakage, network congestion, and a significant burden on the resources available for cloud storage. For delay-sensitive applications like autonomous driving, real-time video streaming, and augmented reality, it is challenging to meet the requirements. These applications often have to deal with a substantial quantity of data; hence, a significant number of resources are required during execution.

A significant prominent strategy was introduced in December 2014 to address these issues. This strategy was first referred to as mobile edge computing (MEC), but in September 2017, it was later renamed multi-access edge computing [6,7]. Data processing as well as system monitoring and control are all possible with edge computing. Edge computing may be thought of as a kind of cloud computing extension that applies to edge networking [8]. In addition, the Internet of Things (IoT) and the edge have combined to form a unique ecosystem known as the edge-IoT ecosystem, which has intrinsic advantages over previous computing paradigms, including grid computing, cloud computing, and fog computing [9–13]. However, the computational and storage capabilities of edge servers as well as edge devices are rather constrained in comparison to cloud computing [14], making them unsuitable for cloud computing applications and systems. Therefore, comparable system architectures and critical technologies are required for edge applications. The essential technologies are a lightweight database, a lightweight web front end and back end, and an expandable and flexible microservice system architecture.

Previously, monolithic architecture was preferred due to its simplicity and low overhead. Monolithic architecture is an older software design approach where all the components of an application are tightly integrated and run as a single, cohesive unit. This means that changes to any one component can impact the entire application. It is simple to develop and deploy but can be difficult to scale and maintain. On the other hand, due to its ability to decouple an application into several lightweight services and reuse capabilities, the microservice is a novel architecture that has been adopted by many service providers, such as Netflix, Apple, YouTube, Facebook, and Amazon. Therefore, research on migrating from monolithic to microservice architecture is important because it can help the organizations to understand the best practices, challenges, and benefits of making the switch. Migrating from a monolithic architecture to microservices can offer several advantages, including improved scalability, flexibility, and maintenance. However, it also requires careful planning, testing, and implementation. By researching the migration process, businesses can better understand the trade-offs and risks associated with the transition and provide guidance on how to navigate the process effectively. There are numerous advantages to utilizing microservices, some of which are illustrated in Fig. 1.

- **Better scalability:** The microservices architecture is designed in a way that allows for easy scalability due to the independent functioning of each module. This makes the overall system design simpler to use and makes it easy to add new features quickly. Moreover, because microservices are frequently stateless, they may allow for quick horizontal scalability if they are properly deployed.
- **High resiliency:** Microservices architecture offers not only simplified maintenance but also increased resilience due to the separation of systems. This enhanced resilience enables quicker troubleshooting, resulting in faster system recovery. As a result, if one service fails, the others will continue to function normally without having any effect on the overall system.
- **Faster deployment:** Microservices architecture simplifies the design, enabling faster development and deployment. The individual modules can be built, tested, and deployed independently. Moreover, due to the smaller size of a microservice, it is faster to include additional features.
- **Greater flexibility:** Microservices architecture provides greater flexibility by enabling independent operation of each module, allowing for easier implementation of changes and updates, and allowing the ability to choose the most suitable technology stack for each service, resulting in better overall performance and increased efficiency.
- **Reliability and reusability:** Microservices increase the reliability of the application because each service can be designed independently. Therefore, it is possible to make minor changes and reuse them. This makes it possible for different services to consume the same feature without having to implement the code twice, which saves time and money.
- **Smaller development teams:** The development of microservices typically involves small and decentralized

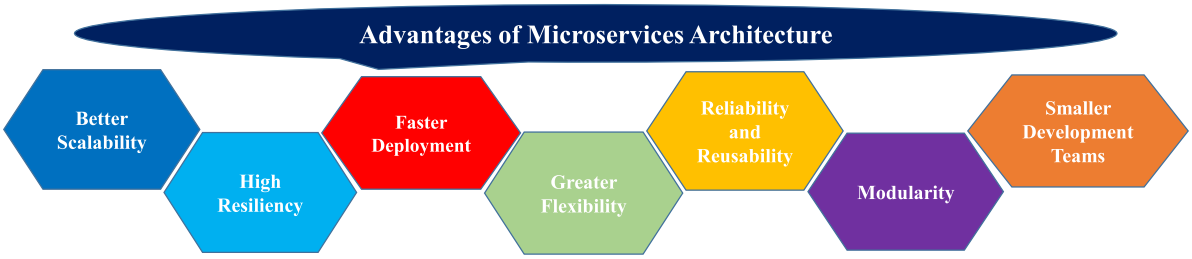

Fig. 1. Advantages of a microservices architecture.

teams, which makes them well-suited for implementing the agile methodology. Microservices are made to be broken up into multiple components that are easy for small teams to build. Each service component is focused on a specific module and can be developed and deployed on Docker independently. This makes it easier for smaller team members to grasp the functionality in less time.

- **Modularity:** Modularity in microservices allows the system to be composed of small, independent modules that communicate with each other through APIs. This results in increased agility, simplified development, increased scalability, and easier maintenance.

Additionally, the microservice architecture (MSA), powered by container technology, divides a monolithic program into several distinct microservices that may cooperate with one another but that operate independently [15]. In the computing platform, loosely linked microservices can be dynamically deployed, launched rapidly, migrated easily between edge clouds on demand, and maintained [16]. In order to provide services with extremely low latency using IoT devices, such as identification of defects in industrial equipment using intelligent systems, and object identification microservices for autonomous vehicles, the microservice-oriented edge computing platform is a potential solution [17]. The combination of these two technologies has the potential to revolutionize the way we design and deploy distributed systems by improving performance, reducing latency, and enabling real-time decision-making. Although the MSA makes it easier to design applications for edge computing, it has become difficult to deploy microservices on edge nodes with limited resources to increase service computing performance [18].

A variety of research has been focused on either microservices or edge computing separately, without considering the integration of the two. However, to the best of our knowledge, the majority of the surveys, concentrated either on a very specific topic or did not include the full scope of microservice-enabled edge computing. As a result, there is a lack of comprehensive understanding of the opportunities, challenges, and future directions of the microservices approach in edge computing. To address this gap, this paper provides a comprehensive analysis of the microservice approach in edge computing with a concentration on research opportunities, challenges, and future research directions. Moreover, this paper examines the existing literature concerning edge computing and microservices, and reviews recent work on the use of microservices in edge computing environments. The contributions of this study are summarized as follows.

- We present a comprehensive background on edge computing, discussing its introduction, architecture, benefits, and challenges in detail.
- A comprehensive overview of microservices is presented, introducing its different characteristics, microservice technology development tools and their URLs, benefits, and challenges. In particular, we compare the monolithic architecture to the microservice architecture, and summarize research published on the migration of monolithic systems to the microservice-based architecture.
- We review and summarize state-of-the-art research on the role of microservice-based applications in edge computing environments, looking at the AI-based microservice framework, and security concerns of the microservice-enabled architecture in edge computing. In addition, we investigate the prospective uses of microservices in edge computing across various domains.
- Finally, we highlight the opportunities and further research directions in this area. These research directions include blockchain as a service, integration with cloud services, microservice orchestration platforms for the edge, DevOps and self-adaptive deployment, unified frameworks, resource allocation, security challenges, and so on.

Fig. 2 provides an overview of the content in this paper. Section 2 presents a comprehensive background study and the challenges for edge computing and microservices. In addition, a summary of the tools used to construct microservices is found in this section, as well as a comparison between the monolithic architecture and the microservice architecture. In Section 3, we discuss and summarize the existing survey literature as well as the different roles for microservices in edge computing environments. The potential application of microservices in edge computing across various domains is outlined in Section 4. Unresolved issues and directions for further research are discussed in Section 5, and Section 6 concludes the paper.

2. Background and fundamentals

2.1. Edge computing

The new computing paradigm called edge computing has emerged to overcome the problem of communication delays in latency-sensitive applications. This paradigm shifts the storage and processing capabilities of cloud computing to the network's edge and closer to the end users (Fig. 3). The edge

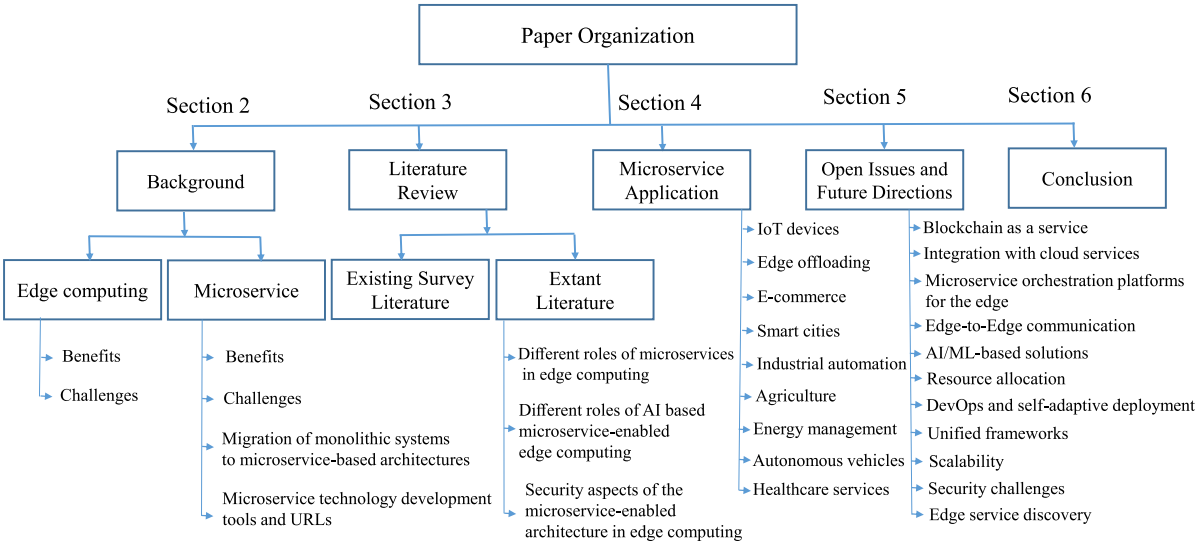

Fig. 2. Organization of this paper's content.

computing architecture typically consists of three key layers: the device layer, the edge node layer, and the cloud layer. When compared to the conventional cloud, edge computing is unique. Researchers have many definitions for the term edge computing. Shi et al. [19,20] are credited with first using the term. It is an innovative computing strategy for the execution of network edge tasks, located between the cloud computing center and source of the data. In this context, the downlink data represent a cloud service, and the uplink data represent the Internet of Everything that edge computing entails. However, Satyanarayanan defined edge computing as a new strategy in which storage and computer resources (such as cloudlets, fog nodes, or micro-data centers) are deployed at the network's periphery, close to user devices [21]. Based on the above two definitions, Zhao et al. [22] defined edge computing as a new kind of computing that consolidates processing, storage, and network resources that are physically or virtually near the end user.

Furthermore, the edge computing industry in China has defined it as an open platform that combines fundamental capabilities, including networking, computation, storage, and applications, that delivers edge intelligent services nearby to fulfill an industry's real-time business needs [23]. Moreover, the purpose of edge computing is to do the computations and provide services where data are created. When it comes to information technology, low latency and high bandwidth are necessities, and edge computing is the process by which storage and computing resources are moved to the network's periphery in order to meet these demands. The field of study known as edge computing has recently exploded in popularity [24–32].

2.1.1. Benefits and challenges of edge computing

Edge computing enables the processing, analysis, and transfer of data at the edge of a network in real time. In the edge computing paradigm, data are stored and processed locally instead of being sent to the cloud. Edge computing greatly benefits from this characteristic in the following ways.

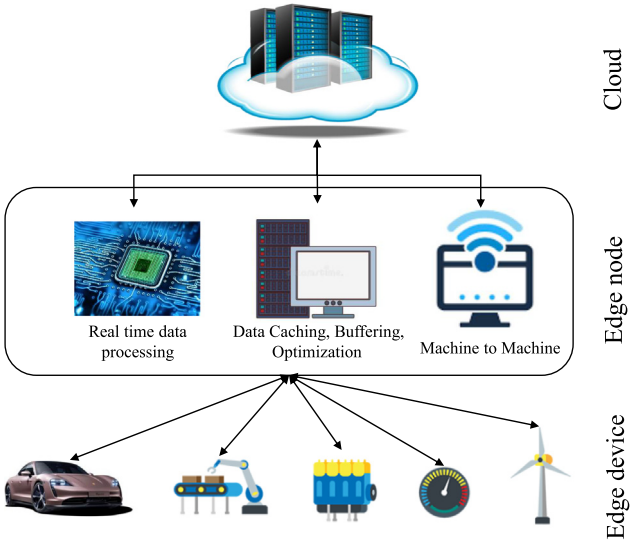

Fig. 3. An example of the edge computing architecture.

- **Speed:** In edge computing, data processing is faster because it allows processing on the spot or at a local data center. Bringing data analysis tools and applications closer to the source greatly reduces network congestion, and increases responsiveness, speed, and overall service quality.
- **Minimized latency:** An information highway bottleneck may form if everyone uploads files to the cloud at once. However, edge computing alleviates part of the burden on the cloud platform, which in turn decreases latency. Moreover, it eliminates the round trip to the cloud, which reduces latency, improves the user experience, and makes it possible to respond in real time.
- **Enhanced privacy protections and data security:** Edge computing reduces the quantity of data transfers between a device and a centralized cloud, which results in increased cloud security. In addition, while sending the

data across international borders via networks, a significant percentage of raw data is processed near protected edge devices. The result is a higher level of data security and privacy protection because data are processed at the edge rather than on central servers.

- **Scalability and versatility:** Expanding computing resources and enabling more consistent performance may be accomplished by utilizing edge data centers in conjunction with dedicated Internet of Things devices. In addition, edge computing makes it possible to collect huge quantities of different and important data, and easily handle raw data.
- **Enhanced reliability and resiliency:** Edge computing enables data to be retrieved and processed with minimal to no interruptions, even when there is poor Internet connectivity. Because operational processes take place in close proximity to the user, the system is less reliant on the functioning of the central network. In addition, even if a problem arises at one of the system's edge devices, it will not affect the functionality of any other edge device in the system. This contributes to an increase in overall system reliability.

On the other hand, edge computing brings the following challenges.

- **Bandwidth bottlenecks:** Traditionally, enterprises allocate more bandwidth to their data centers, and provide less bandwidth to their endpoints. On the other hand, processing data at the network's periphery requires a lot of bandwidth for proper workflow; hence, the aforementioned dynamics shift drastically when edge computing is deployed. The challenge is to maintain a balance between the two while maintaining high performance. In an edge computing server, more bandwidth is required across all individual ends of the server. This creates a need for more bandwidth, compared to traditional networks.
- **Cost and storage:** There is an extra cost on the client side, even if cloud storage is cheaper overall. Much of this comes from creating storage capacity for edge devices. Moreover, there is a cost component for replacement of, or upgrading, the existing IT network infrastructure to handle edge devices and storage. On the other hand, troubleshooting and repairing any issue within the framework requires a significant amount of logistics and manual input. As a result, the cost of maintenance increases because multiple edge receivers are placed at certain distances from the data center.
- **Distributed computing:** Due to the limited processing capacity, a large number of distributed edge nodes are unable to provide all services in their entirety and on their own. The majority of the edge nodes are distributed, meaning they are a significant distance apart. On the other hand, edge computing has a tendency to bring all systems closer to where computations are made. There will be conflict because a business server will need to take the edge server into account as an additional factor while computing.
- **Lost data:** The edge computing model is typically driven by where data are created. Data loss can be prevented if careful planning and programming go into implementing the system. After data collection, many edge computing devices rightly trash useless data. However, if the data removed are really important, they are lost, and the cloud analysis is inaccurate.
- **Security:** Just as there is a benefit to security in the cloud and in large organizations, there is danger at the local level. Information technology departments have been dealing with this issue for years, but the lack of monitoring has now spread to the network's new edge nodes. The most recent paradigm in information security, known as the *zero trust security architecture*, is based on the idea that no part of data communication is secure since there are no trusted individuals. The implementation of a real-time operation with minimal latency requirements continues to show itself to be a significant difficulty.

2.2. Microservices

Microservices are independent pieces of code (similar to functions) that may be merged to make a larger whole [33]. As a result of its decentralized nature, programs may be more reliable and extensible by eliminating any single point of failure. Modular design makes it simple to add new features, reduces the severity of bugs, and streamlines the process of fixing existing bugs. The architecture encourages a method of software development in which frequently used operations are exposed as services and can be scaled automatically [34]. Microservices are independent components that can be called by an application, just like any other function, and they can be hosted remotely on virtual machines or in containers. This makes it possible for applications to function, regardless of the underlying technology or programming language used to create the microservice that the application relies on. In addition, microservices allow applications to be built such that they use a wide variety of back-end databases, including cloud, relational, and NoSQL systems. Docker, Kubernetes, and other infrastructures [33,34] are often used for deploying microservices. The five main characteristics of the microservice architecture are illustrated in Fig. 4.

The majority of existing applications have been built using a monolithic (single-solution) approach to software design and deployment. The monolithic architecture allows the use of a small number of programming languages to create applications or processes that are composed of multiple classes, procedures, and packages. The entire application or process is carried out by a single server, regardless of the application's needs. This architecture is easy to build, test, and scale horizontally because it does not rely on particular modules [35]. As a result, its scalability is low; an excessive load on any one function might cause a bottleneck, and modifying a single function can have unintended consequences for other related functions. In addition, the architecture makes it harder to redeploy and maintain, and it is far more challenging to find a solution to the

Table 1
Monolithic versus microservice architectures.

Point of difference	Monolithic architecture	Microservice architecture
Approach	Unified	Modular
Language	.NET, Java, PHP, Ruby, Python	DevOps, Docker, Kubernetes, Lambda
Size and complexity	Tends to be large and complex, with many interdependent parts	Smaller and less complex, with each service focused on a specific task
Scalability	The scalability of the system is restricted, because it is not feasible to scale individual modules	Each element can be scaled independently owing to the many independent services
Flexibility	Less flexible, as new features require changes to the entire application	More flexible, as new services can be added or removed without affecting the rest of the application
Code	A huge single codebase for the entire module	Each microservice module has its own codebase
Failure	A single failure can bring the entire system crashing down	The failure of one service will not have any impact on other services
Costs	Starting expenses are lower but can increase over time	Initial expenses are greater but can decrease while operating
Testing	End-to-end testing of entire modules	Each component has to be tested individually
Development	Teams have to work on the development process simultaneously	Different teams can work on different process
Updates	Updates take time due to internal dependencies	Updates are fast owing to the independent nature of services
Deployment	The entire system is deployed at once	System deployment for each service is conducted individually

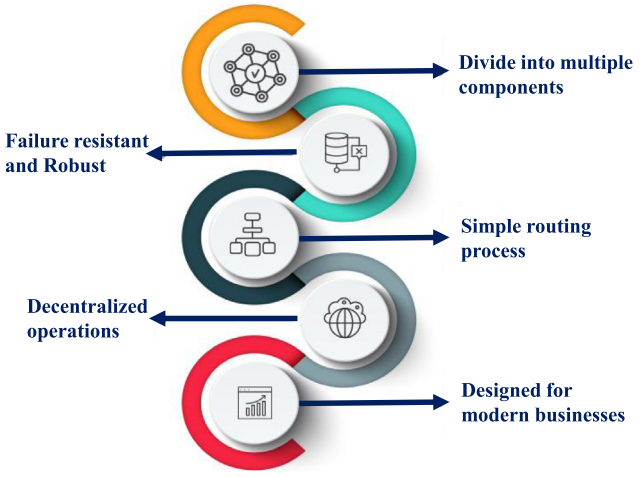

Fig. 4. Characteristics of the microservices architecture.

issue of physical heterogeneity [36,37]. Fig. 5 demonstrates an example of monolithic and microservices architectures.

The recent adoption of microservice designs in which a single solution or application is broken down into smaller, more manageable services addresses these flaws. Every microservice is dedicated to a specific task and operates independently. In addition, any microservice may be implemented in any language [36]. Although the microservices can be deployed separately, they still work together to achieve the larger business objective. Like modular monolithic architectures, a microservice employs the divide-and-conquer strategy to deal with the complexity of software systems, reducing large, monolithic programs into smaller, more manageable components that interact with one another through APIs. With a modular monolithic architecture, all modules are often delivered together, whereas with a microservices architecture, the individual microservices are deployed separately and can be quickly tweaked and altered, making redeployment and maintenance much simpler. Furthermore, support for multiple technology stacks and fault tolerance improves the scalability of the microservices architecture [38]. The differences between monolithic and microservice architectures are shown in Table 1 [39,40].

2.2.1. Challenges of microservices

With so many positive aspects, it is no wonder that the microservice architecture has been adopted by so many businesses and people. While the adoption of microservices is on the rise, there are certain limitations to this design style. When designing microservices, organizations face a higher degree of complexity than when developing monolithic applications. The architecture poses a variety of challenges, some of which include deciding the size of each individual microservice, identifying the optimum boundaries between microservices, and integrating microservices with one another. In addition, microservices present the following challenges.

- **Resiliency:** The implementation of microservices is made more challenging by the fact that application functionality is decomposed into a large number of independent deployable components. One of the difficulties that arise as a consequence of this complexity is failure detection. A failure might be caused by a variety of factors, including the microservice itself, the network used to link microservices, or the container it executes on.
- **Communication complexities:** Since everything is now a service, requests traveling between modules need to be handled with care, which adds complexity to communication between services. Developers may need to add extra code to prevent disruption.
- **Difficulties for global testing:** The testing phase is more challenging for microservice-based applications because

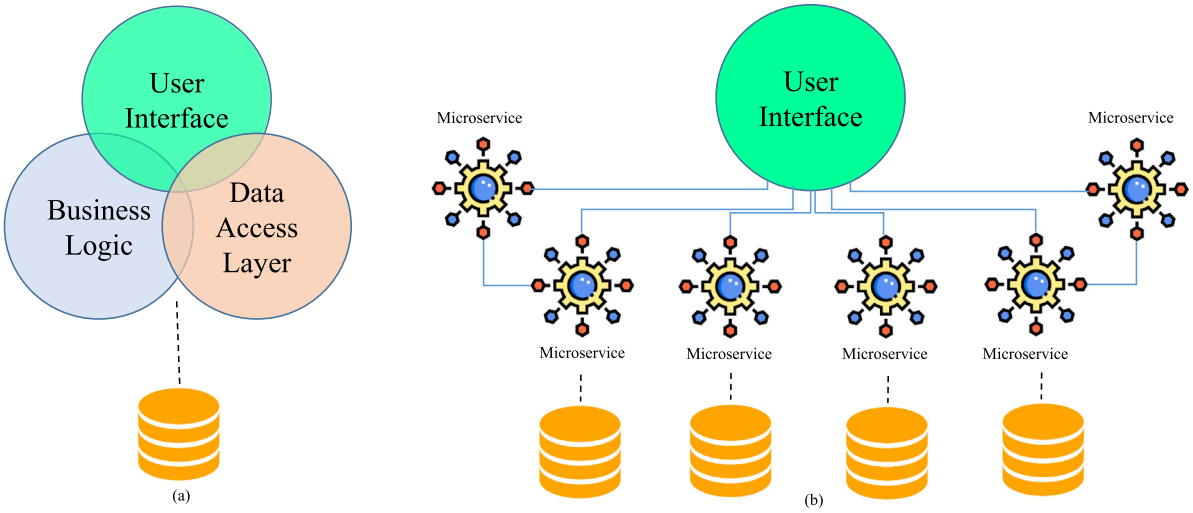

Fig. 5. (a) The monolithic architecture versus (b) the microservice architecture.

of the time and effort required. Because each microservice operates independently, you need to test them individually to ensure everything works properly. Moreover, testing cannot begin until all dependent services have been verified.

- **Large versus small production companies:** While microservices are ideal for enterprises, they can be cumbersome for smaller businesses that need to innovate and iterate rapidly without wasting time on intricate orchestrations.
- **High complexity requiring more resources:** Microservices are small and modular, but in an application comprised of hundreds, they call each other, which may become difficult to manage. Moreover, as the demand for services grows, so does the need for additional resources. Keeping track of multiple databases and transactions at once can be a hassle.

2.2.2. Migration of monolithic systems to microservice-based architectures

The microservice architecture has emerged in recent years as a viable alternative to monolithic design, and has the potential to successfully address the drawbacks of the monolithic approach. This is accomplished by decomposing the monolith into a group of smaller services and ensuring that these services interact with one another via lightweight techniques, such as a RESTful API. The benefits of utilizing microservices have been widely recognized in both industry and academia. Some of these advantages include automated deployment, scalability, reusability, availability, and maintainability. In order to take numerous benefits, provided by a microservice design, well-known companies like eBay, Amazon, and Netflix have all migrated their systems to microservice architectures. However, decomposing a monolith architecture into a set of autonomous services can be a very difficult task and can have a significant negative effect on performance, particularly when migrating large applications to a new architecture. As a result, it is necessary to carefully design such modernizations with regard to performance. In order to overcome the problems discussed above, a number of researchers have proposed a systematic and easy-to-understand microservice-oriented decomposition strategy [41–50], which is shown in Table 2.

The majority of the approaches are the proposed solutions; nevertheless, they do not give well-structured direction for software developers and maintainers who are working on the implementation level. In the fields of architectural migration and refactoring, there are still unresolved problems that need to be researched further. The process of migrating from monoliths to microservices may be time-consuming and challenging. Moreover, it is necessary to make a decision on which components may be migrated and which components need to be rebuilt from the beginning. Taking the aforementioned discussions into consideration, we propose four significant series of steps (feature analysis, information extraction, refactoring, and orchestration) that can be taken to migrate monolithic systems to microservice-oriented architecture, as shown in Fig. 6.

- **Feature analysis:** This step is very crucial for analyzing and identifying whether or not monolithic modules are loosely or tightly coupled. We proposed using both static analysis and dynamic analysis to figure out the characteristics of a monolithic application's runtime behavior as well as the structure of its static representation. A static analysis of the program is performed in order to acquire information on the structure of the legacy application. This information includes the possible linkages that exist between classes, methods, and variable members. Then, we use dynamic analysis to extract characteristics of runtime behavior, such as how functions and classes interact with each other. We take advantage of the features that we can acquire from the program's source codes, developer information, and design documents in order to make the application partitioning as efficient as possible.
- **Information extraction:** Applications that are monolithic are comprised of several interconnected modules. In order to identify these modules, it is necessary to

Table 2
Summary of existing microservice-oriented decomposition articles.

Reference	Proposed strategy	Contributions	Year published
[41]	Dataflow-driven approach	The proposed mechanism simplifies microservice-oriented decomposition and provides a systematic methodology, reducing complexity.	2017
[42]	Dataflow-driven semi-automatic decomposition approach	The proposed approach offers a systematic and easy-to-understand methodology for developing microservice-based systems that effectively manages the uncertainty and effort of decomposition practices.	2019
[43]	Domain-driven design (DDD) approach	This paper introduces step-by-step domain-driven design (DDD) concepts to migrate from legacy systems to microservices.	2019
[44]	Graph-clustering methodology	The proposed method employs both static and evolutionary code coupling information along with graph clustering techniques to automatically extract microservices from monolithic applications.	2018
[45]	Graph clustering algorithms for automatic decomposition	Static, semantic, and evolutionary approaches are used to decompose monolithic software into microservices. For each approach, the authors calculate a weighted graph and aggregate all three approaches into a single graph.	2020
[46]	Microservices extraction based on evaluation metrics and an interface analysis	This paper introduces an evaluation metrics strategy that allows developers to decompose and evaluate microservices applications by analyzing the application programming interface.	2020
[47]	A metric-based evaluation framework for decomposition approach	The proposed framework utilizes process mining of the original system's logs to decompose a monolithic system into distinct microservices.	2019
[48]	Functional and non-functional metrics-based microservice identification	The proposed strategy extracts microservices from legacy systems while taking into consideration both functional and non-functional characteristics.	2020
[49]	Functionality-oriented service candidate identification framework	A search-based functional atom grouping strategy is used to extract potential service candidates from a monolithic system.	2021
[50]	Topic-modeling-based microservice identification from monolithic applications	This paper introduces a method for identifying microservices that involves utilizing topic modeling to detect domain-specific terms and clustering techniques to generate a set of services from the original software.	2021

extract lexical and structural dependency information extraction metadata. This metadata contains a variety of information, including data flow, boundary context, and operating platform, among others. The extraction of metadata has several advantages, one of which is that it is beneficial in the process of separating a program into presentation, business logic, and data access layers. Lexical extraction is responsible for the process of extracting all of the lexical and textual keywords from the source code, while structural dependencies provide a more accurate representation of the software architecture.

- **Refactoring:** When a monolithic system has been separated into layers and smaller parts, called microservices, each microservice may then be fixed, optimized, upgraded, or even reimplemented as needed. At this stage, the source code can be turned into real microservices either automatically or manually by creating standard APIs. When each feature has been converted into a microservice, then those microservices must be registered in a service catalogue as abstract services before the service orchestration phase can proceed.
- **Orchestration:** After the transformation of each monolithic operation into a microservice, services can be orchestrated on the basis of particular data in order to satisfy the requirements of the software's business logic. In many cases, business logic is frequently obtained from monolithic software systems. It is also possible to modify it in accordance with the quality standards of the target microservice architecture.

2.2.3. *Microservice technology development tools and URLs*

The latest trend in software services, the microservices architectural paradigm is derived from a service-oriented architecture for improving software agility, scalability, and independence. Design, development, and delivery of software services are revolutionized by the rise of microservices. Table 3 shows the wave of microservice technology development tools as well as URLs for each tool. The majority of the tools in Table 3 originated in industry, and they are freely accessible to the public as open source projects [36,51].

The first wave of technologies was made up of lightweight container technologies such as LXC and Docker. These technologies make it feasible for individual services to be packaged, deployed, and maintained more efficiently during runtime. Service discovery technologies like eureka and consul enable different types of services to communicate with one another without having to make direct reference to the specific locations of each service on the network. Monitoring tools such as graphite and influxDB make it possible to monitor and analyze the activity of microservice resources in real time at varying degrees of granularity. Container orchestration provides automatic container allocation and task management by using different tools like mesos and kubernetes. Fault-tolerant communication tools finagle and resilience4j allow

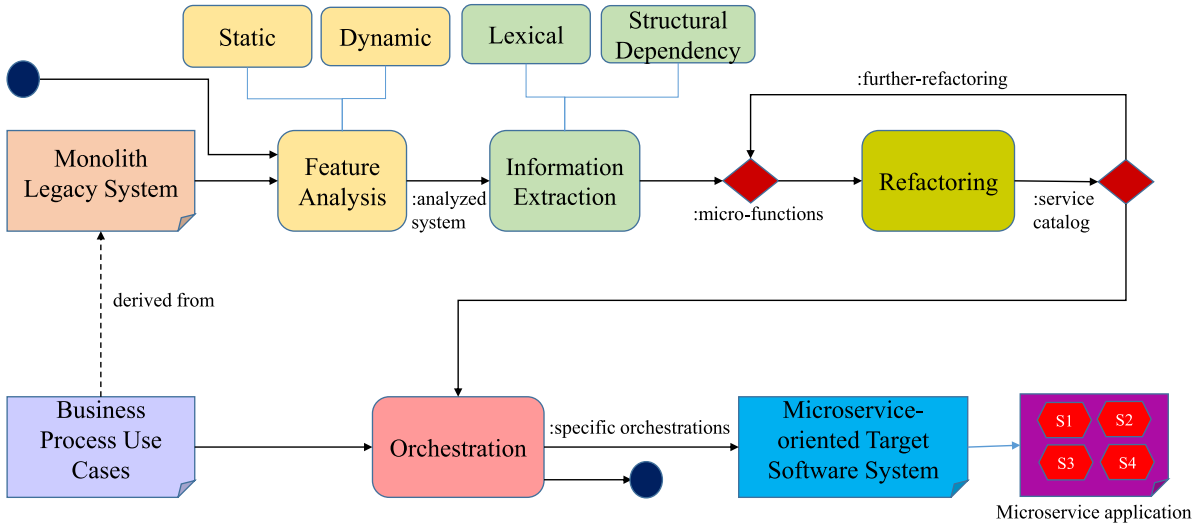

Fig. 6. The migration steps from monolithic to microservices.

Table 3
Microservice technology development tools and URLs.

Purpose of the microservice	Tool name	Year developed	URL
Containerization	LXC	2008	linuxcontainers.org
	Docker	2013	www.docker.com
	rkt	2014	coreos.com/rkt
Service discovery	ZooKeeper	2008	zookeeper.apache.org
	Eureka	2012	github.com/Netflix/eureka
	Consul	2014	www.consul.io
Monitoring	Graphite	2008	graphiteapp.org
	InfluxDB	2013	github.com/influxdata/influxdb
	Prometheus	2014	prometheus.io
Container orchestration	Mesos	2009	mesos.apache.org
	Kubernetes	2014	kubernetes.io
	Docker swarm	2014	docs.docker.com/engine/swarm
	Amazon elastic container service	2015	aws.amazon.com/ecs
Fault tolerance	Finagle	2011	twitter.github.io/fnagle
	Resilience4j	2016	github.com/resilience4j/resilience4j
Serverless computing	Amazon web services lambda	2014	aws.amazon.com/lambda
	Azure functions	2016	azure.microsoft.com/services/functions
	Google cloud functions	2016	cloud.google.com/functions
	Spring cloud function	2017	cloud.spring.io/spring-cloud-function

services to communicate more efficiently. Finally, serverless computing technologies such as azure functions and google cloud functions are used to create the function-as-a-service (FaaS) cloud paradigm.

3. Surveys and the extant literature

3.1. Surveys of microservices in edge computing

Edge computing is a new paradigm that brings cloud services closer to the network's periphery in order to facilitate the development of decentralized architectures. Due to the confined resource capacity and distributed nature of edge computing, edge applications are required to be built as a collection of lightweight components that are linked together. On the other hand, the microservice architecture is widely used to accommodate the ever-changing requirements of IoT applications. Because of its fine-grained modularity as well as its independently deployable and scalable nature, the microservice architecture shows great promise in leveraging both cloud and edge resources to fulfill the numerous QoS requirements of IoT application services. Therefore, the combination of edge computing and microservices has attracted a lot of interest from both academia and industry. Some survey articles were published that mostly focus on the migration to microservices from a monolithic architecture, whereas the focus of other survey articles is on the use of secure microservices in edge computing [40,51–56]. There is still a lack of research that completely analyzes the role of microservices in edge computing, and a significant amount of research needs to be done in different directions. Unlike [40,51–56], the purpose of our study is to conduct an extensive survey covering various

Table 4
A summary of existing survey articles.

Reference	Aspects	Contributions	Year published
[52]	Pains and gains of microservices	The authors focus attention on the operational and technical benefits, as well as drawbacks, of the architectural pattern built on microservices.	2018
[51]	Microservice journeys and challenges	An overview of microservice evolution, benefits, architectures, different tools, and some future challenges is discussed in this survey.	2018
[53]	Migration to microservices	The authors focus attention on the process of upgrading legacy and monolithic systems to designs that are based on microservices.	2022
[40]	Microservices in edge and fog computing	The authors focus mainly on taxonomy of the current work with regard to scheduling microservice-based IoT applications in fog and edge computing environments.	2022
[54]	Observability in distributed edge microservices	The authors explore the concept of observability, in a distributed edge microservices environment, and outline future research directions to stimulate progress in this field.	2022
[55]	Security challenges in microservices-enabled fog applications	The authors investigate various security vulnerabilities that endanger microservices-based fog applications.	2019
[56]	AI-enabled Microservices in edge computing	The authors focus mainly on security and privacy issues when deploying microservices at the IoT edge.	2023

aspects of state-of-the-art research, encompassing topics such as microservice-based applications in edge computing environments, the framework of AI-based microservices, and security considerations associated with the architecture of microservices in edge computing. Survey articles on microservices in edge computing are summarized in Table 4.

There has been a growing trend in the information technology sector toward designing, developing, and running microservices. However, academic research on the topic is just getting started, and there is a long way to go before we can extract the true pains and gains from using microservices as an architectural approach. In light of this deficiency, Soldani et al. [52] conducted a comprehensive review of the so-called grey literature on microservices throughout the industry in order to determine the technical/operational drawbacks and benefits of this architectural approach. The final section of their research is devoted to a discussion of prospective new avenues of investigation that may be pursued in the future. Most of the IT industry began focusing on expanding their microservice consulting services. Therefore, Jamshidi et al. [51] focused on the introduction of microservice architectures in which software components are treated as autonomous units for development, deployment, and scalability, contributing to a rise in software agility. Microservice evolution, different tools, and some future challenges were also discussed in this survey.

The current trend in system migration is to replace large, monolithic applications with smaller, more modular applications that are developed on top of a microservices architecture. The purpose of a survey article by Capuano and Muccini [53] was to determine how changes made to enhance quality features directly motivate the migration process. The authors also identified several challenges and best practices for successful migration to microservices, including the use of proper tools and methodologies, architectural patterns, and testing strategies. Overall, the paper provides valuable insights into the state of the art in microservices migration and highlights the need for more research on certain quality attributes. However, Pallewatta et al. [40] provided a complete taxonomy of current research on scheduling microservice-based IoT applications in edge and fog computing. That research organized several taxonomies in order to capture the key features of the scheduling problem, to identify research gaps within each category, and to analyze and categorize similar works, highlighting research gaps within each category and talking about potential future research initiatives.

The deployment of containerized workloads as microservices in distributed edge infrastructures adds complexity, making it difficult to detect and troubleshoot outages, especially in critical use cases like industrial automation processes. Therefore, Usman et al. [54] presented a survey of observability in distributed edge microservices. Observability plays a crucial role in aiding operators in effectively managing and operating complex distributed infrastructures and microservices architectures by offering comprehensive insights into runtime performance across the entire system. Furthermore, the authors address open research issues and outline future research directions in this survey.

On the other hand, when deploying microservices at the edge of the Internet of Things, there are a number of security and privacy issues to keep in mind. Due to advances in artificial intelligence and easy access to resources with powerful computation capabilities, it is now possible to make exact models and run a variety of smart applications at the edge of the network. Therefore, the authors of [55] explore different security risks that arise in the service communication of microservices-based fog applications. On the other hand, Al-Doghman et al. [56] presented a comprehensive overview on the topic of securing AI microservices that run on edge computing to shed light on the difficulties of IoT management, making it possible to implement trustworthy decision-making infrastructures at the network's periphery. The key requirements and problems of protecting microservices at the IoT edge are presented, together with the results of current research on edge AI and the orchestration of microservices. Their

Table 5
A summary of different microservice roles in edge computing.

Reference	Design objectives	Method applied	Performance metrics	Evaluation method	Computing nodes	Year published
[17]	Microservice coordination problems	Dynamic programming based offline and reinforcement learning based online for microservice coordination	Service delay and cost	Numerical, simulation	Edge cloud, core cloud	2021
[18]	Microservice orchestration problems	Lightweight virtualization based on Docker orchestration in a highly dynamic system	Communication delay	Simulation	Edge cloud fog, and core clouds	2018
[57]	Microservice repositories	Topology and orchestration specification for cloud applications (TOSCA) is used for microservice repositories	Latency and energy consumption	Simulation	Edge cloud	2019
[58]	Dynamic microservice scheduling	Lagrangian multipliers to solve the optimization problems in microservice scheduling	Network delay and energy consumption	Simulation	Edge cloud	2020
[59]	Microservice-based task offloading and scheduling	Cost-aware resource matching and a task scheduling CACOTS algorithm is used for scheduling strategy	Server utilization and minimizing costs	Simulation	Edge cloud	2020
[60]	Hybrid microservice autoscaling problems	A Kubernetes horizontal autoscaling algorithm is used to develop two novel hybrid scaling techniques	Execution time and failure requests	Simulation	Edge cloud	2019
[61]	Collaborative layer-aware microservice deployment strategies	Integer linear programming and a randomized rounding-based algorithm are utilized to deal with the problems of microservice deployment and layer storage	Throughput	Numerical, simulation	Edge cloud	2022
[62]	Microservice-based service sharing	Using system and application containers allows several IoT applications to share microservices	Service time	Simulation	Edge cloud	2022
[63]	Microservice-based dynamic service migration	Lyapunov optimization and randomized rounding-based service migration and request routing strategy is used for service migration	Network utility and migration costs	Numerical simulation	Edge cloud, core cloud	2022
[64]	Distributed placement scenarios for microservice-based applications	Stochastic optimization-based distributed redundant strategy is used to placement scenarios	Response times	Simulation	Edge cloud	2022

research also presented a microservice-based edge computing architecture that makes use of containerization technology to deliver safe and automated AI-based applications at the network's periphery.

3.2. Different roles of microservices in edge computing

Microservices have a significant role to play in the development and deployment of edge computing applications. In recent years, the combination of microservices and edge computing technologies has gained significant attention from researchers and industry practitioners. Researchers have proposed various microservice roles on the platform of edge computing with various design objectives. Table 5 provides a summary of the different roles of microservices in an edge computing environment.

In order to provide smooth and immediate replies to service requests from mobile users, Wang et al. [17] examined microservice coordination among edge clouds. In their system, an offline microservice coordination approach based on dynamic programming is suggested as a way that can deliver the best performance. The microservice coordination problem is then reformulated using a Markov decision process (MDP) framework, and an online microservice coordination method based on reinforcement learning (RL) is presented to determine the best strategy. The results of experiments showed that the suggested online algorithm performed better than current methods in terms of migration costs and service latency, and that performance and came very close to ideal performance that can be reached by using an offline method. On the other hand, Alam et al. [18] suggested a lightweight, virtualization-based modular and scalable Docker Edge Orchestration (DEO) framework for orchestrating microservices

in an IoT environment using edge computing. The authors present how microservices can be used to build complex applications for the IoT and how Docker can be used to package and deploy these microservices. They evaluate the performance of the DEO framework through experiments and compare it with existing solutions.

Deploying microservices close to the edge of the network offers the benefit of ultra-low latency. Therefore, a novel offloading technique was developed by Gedeon et al. [57] in their work based on the microservices paradigm, which may be accessed through a central hub known as the microservice store. The authors provide a comprehensive overview of the use of a microservice store for efficient edge offloading and propose a new approach that enables the discovery, selection, and deployment of microservices at the edge. Their findings demonstrate that the microservice store is able to provide efficient edge offloading of computation and is able to save up to 94% of battery life while reducing latencies by as much as 14 times.

On the other hand, to reduce overall network delay and cost, it is necessary to schedule microservices dynamically. Therefore, Samanta and Tan [58] suggested a dynamic microservice scheduling approach for MEC-enabled IoT environments, which takes into account both the resource availability and the user's QoS requirements. Their strategy significantly enhanced performance with regard to the percentage of failed tasks and the overall cost. The most recent cloud frameworks provide services built on top of virtual machines. However, these frameworks suffer from a number of issues, including a slow start-up time, extra overhead, and needless expense to operate IoT applications. To execute delay-sensitive applications efficiently, Zhao and Huang [59] suggested a new approach for efficient computation offloading that consists of a microservice-based architecture, a dynamic resource allocation mechanism, and a cost-efficient task scheduling algorithm. The problem of cost-effective task scheduling across the several fog servers was also studied. Moreover, the authors provided a framework called cost aware computational offloading and task scheduling (CACOTS). This framework divides the process of task scheduling into several processes such as resource matching, job sequencing, and scheduling.

Currently, many organizations decide to host their microservice designs in cloud data centers to reduce operating expenses. But during peak traffic hours, data centers tend to be overburdened, but underutilized during off-peak hours. The horizontal or vertical expansion of a microservice is the exclusive focus of conventional scaling approaches. However, when employed together, these approaches provide benefits that are complementary and that mitigate each other's shortcomings. Therefore, Kwan et al. [60] developed two novel hybrid autoscaling algorithms, as well as a dedicated network scaling algorithm, in order to leverage the high availability of horizontal scaling and the fine-grained resource control of vertical scaling. However, in recent years, the use of lightweight microservices based on containerization proved the most effective method for increasing the elasticity of edge clouds. Therefore, Gu et al. [61] investigated the issue of how to deploy microservices in a collaborative manner by including both intra-server and inter-server layer sharing in order to get the highest possible edge performance. They formulated their issue in the form of an integer linear programming problem and demonstrated that it is NP-hard. On the other hand, Alanezi and Mishra [62] proposed sharing microservices amongst several IoT applications through the utilization of system and application containers. System containers include functionality for communicating with clients, for starting and stopping containers and applications, keeping track of what containers and applications are running, and sharing containers where feasible. However, depending on the sensors and actuators used, application containers implement a variety of different functionalities.

MEC networks are dynamic and stochastic in nature. Therefore, the deployed services might frequently be transferred among edge servers to follow user mobility. This causes a significant increase in network operating costs. Therefore, in order to optimize the long-term network utility of MEC networks, Chen et al. [63] proposed an optimization-based online service migration technique based on Lyapunov optimization. The approach aims to improve the performance of microservices by dynamically migrating services between edge servers based on workload and resource availability. On the other hand, in order to facilitate the implementation of microservice-based applications, Zhao et al. [64] propose a redundant placement strategy that distributes microservices across multiple edge nodes. The proposed approach aims to provide high availability and fault tolerance for microservices by placing redundant replicas at distributed edge nodes. The authors also proposed a heuristic algorithm to optimize the placement of replicas based on multiple criteria, such as network latency, resource utilization, and availability. The proposed approach can effectively improve the availability and fault tolerance of microservices in edge computing environments.

3.3. Different roles of AI-based microservice-enabled edge computing

Artificial intelligence (AI)-based microservices-enabled edge computing is an emerging paradigm in which AI models and services are deployed at the edge of the network, closer to where data is generated and consumed. This approach offers several benefits, including reduced latency, improved data privacy and security, and enhanced scalability. Previously, different AI algorithms like machine learning (ML), reinforcement learning (RL), and deep learning (DL), among others, were deployed as monolithic application services to enable autonomous decision-making processes that were based on data from IoT devices. To the best of our knowledge, few studies have focused on artificial intelligence models operating at the edge as microservices in IoT systems. AI-based macroservices can analyze data in real-time, enabling faster and more efficient decision-making as well as enabling new applications and services that were not previously possible. Microservices and AI techniques inside IoT-edge ecosystems have been analyzed separately in a number of studies. This

Table 6
Summary of different roles in AI-based microservice-enabled architectures.

Reference	Design objectives	Method applied	Technical approach	Performance metrics	Computing nodes	Year published
[65]	Artificial intelligence as a microservice	ML, DL	Deployment of AI functions as microservices is used to develop an IoT data-driven intelligent infrastructure	Latency	Fog and core cloud	2018
[66]	Machine learning deployment in microservice architecture	ML	Machine-learning-based deployment in microservices to transform a monolithic architecture that separates similar processing steps into smaller services	Processing time	Virtual Machine	2019
[67]	Distributed cooperative microservice caching strategies	DL, RL	Deep dueling deep Q-network based (D3QN) reinforcement learning is used to solve the microservice caching problem	Average delay, hit ratio	Edge cloud	2021
[68]	Microservice deployment strategies	RL	A reinforcement learning and neural-network-based service deployment solution	Waiting time	Edge cloud, core cloud	2021
[69]	Efficient deployment of microservices in the cloud–edge continuum	RL	A reinforcement learning-based algorithm is used to determine the optimal deployment strategy	Latency and costs	Edge cloud, core cloud	2021
[70]	Resource management of microservice applications	Fuzzy Logic	Particle swarm optimization and a fuzzy-based algorithm for microservice resource management	Response time	Edge cloud	2021
[71]	Microservice deployment problems	DL, RL	An RL model based on deep, Q-learning and elastic scaling algorithm are used to obtain optimal deployment strategy	Response time	Edge cloud	2022
[72]	Microservice selection problems in collaborative edge–cloud environments	DL	A deep deterministic policy gradient algorithm is used for microservice selection of heterogeneous and dynamic characters in a collaborative edge–cloud environment	Service access delay	Edge cloud, core cloud	2022
[73]	Intelligent autoscaling of microservices	ML, RL	A generic autoscaling algorithm and RL agents are used to determine microservice resource demand and the autoscaling threshold value	Response time	Edge cloud	2021

Notes: AI = artificial intelligence, ML = machine learning, DL = deep learning, RL = reinforcement learning.

section discusses the state-of-the-art literature and summarizes the different roles regarding AI-based macroservice-enabled edge computing architectures, as shown in Table 6.

Lee et al. [65] proposed AI as a microservices (AIMS), which aims to enable efficient AI service delivery by decomposing AI functions into microservices and distributing them across the network, to reduce latency for real-time 5G applications. The proposed AIMS architecture consists of three layers, including an AI function layer, a microservice layer, and a network layer, which are responsible for handling different tasks related to AI service delivery. On the other hand, Ribeiro et al. [66] introduced machine learning in microservices architecture (MLMA), a general framework that may be used to break down a centralized machine learning pipeline into a set of independent microservices. The proposed architecture comprises of multiple microservices that can be deployed across different computing resources and can handle a large number of requests concurrently. The microservices are designed to perform specific functions, such as data pre-processing, feature extraction, and model training. Their proposed scheme examines the architecture's implementation in two case studies deployed for a smart city program.

Caching microservices at the edge of networks driven by the mobile edge computing paradigm is envisioned as a potential way to enhance QoS and reduce the strain on the core network. However, because IoT devices' retrieval requests are stochastic, and the network architecture changes over time, it is

difficult for IoT devices alone to choose a caching node and replace a microservice without full knowledge of their dynamic surroundings. In light of this, Tian et al. [67] proposed a MEC-enabled distributed cooperative microservice caching scheme called DIMA for better QoS and to alleviate the burden on the core network. To specifically improve the fetching latency and hit ratio, the microservice caching problem is treated as a Markov decision process. Additionally, a distributed double dueling deep Q-network (D3QN) method is presented to resolve the stated MDP, in which each IoT device executes operations separately in a decentralized way.

To enable an intelligent IoT system, the edge–cloud hybrid environment needs sophisticated deployment methodologies. The heterogeneous properties of the edge–cloud hybrid environment are disregarded in existing service deployment methodologies, which instead rely on straightforward, generalized heuristics. Therefore, Chen et al. [68] proposed a novel approach for deploying IoT microservices in a hybrid edge–cloud environment. They use reinforcement learning to optimize the deployment of microservices, and their proposed framework can dynamically migrate services between edge and cloud nodes to improve resource utilization and reduce latency. Moreover, a multiple buffer deep deterministic policy gradient (MB_DDPG) was proposed to offer better service deployment options by considering various factors, such as latency, energy consumption, and resource utilization.

The deployment of microservices has been widely used because it reduces the service response time on an edge computing platform. Microservices that are able to communicate with each other are often co-located in order to reduce the communication overhead between them. Fu et al. [69] proposed an architecture for the deployment of microservices in the cloud–edge continuum. Their approach aims to optimize resource utilization and minimize deployment costs by dynamically deploying microservices to appropriate cloud or edge nodes based on the service's requirements and available resources. To achieve this, the authors proposed a reinforcement learning-based algorithm that takes into account workload and resource availability information to determine the optimal deployment strategy. It dynamically identifies congested nodes and migrates the microservices from busy nodes to idle ones, to alleviate resource contention in response to the load dynamics while reducing communication costs.

On the other hand, an important problem in edge computing networks is how to control the resources of microservice applications. Therefore, Li et al. [70] developed a platform for managing microservice resources in edge computing networks. The proposed platform dynamically manages microservice resource allocation by considering resource availability and service-level agreement requirements. The platform employs a fuzzy-based microservice computing resource scaling (FMCRS) mechanism to allocate resources to microservices in an adaptive and efficient manner, and reduces energy consumption by scaling down unused resources.

The overall service response time on the microservice-oriented edge computing platform may be decreased by employing the microservice deployment approach. Existing efforts, however, fail to account for the impact of varying microservice contact frequencies or the decline in service execution speed due to higher node loads. To describe the communication burden, Lv et al. [71] proposed a multi-objective microservice deployment problem (MMDP) in edge computing based on deep Q-learning. The proposed method focuses on optimizing the deployment of microservices to the edge nodes and improving the system performance. The goal of MMDP is to achieve load balancing across edge nodes while minimizing communication costs. To address MMDP and achieve the optimal deployment strategy, this system presents a learning-based Reward Sharing Deep Q Learning (RSDQL) technique.

On the other hand, the network will be delayed or interrupted if users are always moving around in a mobile edge computing environment. This will have a negative impact on the quality of service. Previous research has demonstrated that this issue may be resolved by utilizing container technology to deploy microservices. However, in a cloud–edge hybrid environment, it is challenging to determine how to select the best possible instance of a microservice from among multiple servers. Therefore, a microservice selection and scheduling optimization strategy, often known as an MSSP approach [72], is implemented in order to reduce service access delays. The authors point out that the traditional approach of selecting microservices based on network conditions and resource availability has limitations in terms of efficiency and adaptability. To overcome these limitations, the authors propose a deep deterministic policy gradient approach that can dynamically adjust microservice selection based on user requirements and network conditions.

Developing a cloud application using a microservices architecture faces some challenges, one of which is scalability at the container level. The currently available solutions for autoscaling, such as Kubernetes, allow some degree of customization in a set of threshold values for autoscaling. However, it is necessary to identify the appropriate values for the numerous autoscaling parameters that are required to guarantee the quality of service in an environment that is always changing and dynamic. Khaleq and Ra [73] concentrated their efforts on building a scalable architecture for microservice applications based on RL that run at the container level. The proposed intelligent autoscaling mechanism utilizes machine learning algorithms to learn the performance characteristics of microservices and predict the resource demand of microservices in real-time. Their strategy is divided into two different modules. The first module is responsible for determining the resource demand for microservices, while the second module makes use of reinforcement-learning agents to learn and determine the autoscaling threshold values based on the resource demand.

3.4. Security aspects of the microservice-enabled architecture in edge computing

Microservices are widely used in industry by breaking down massive monolithic programs into a collection of smaller, interacting services that are easier to manage, scale, and test in edge computing environments. Despite the benefits of

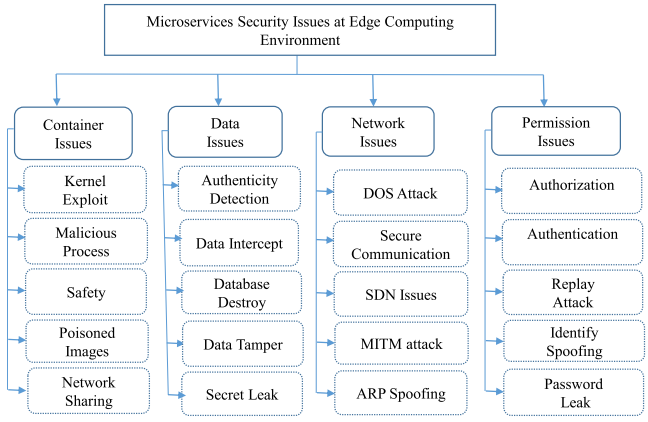

Fig. 7. Microservice security challenges in edge computing.

building and deploying a microservices architecture, the IT community and its customers are always concerned with safety first. Microservice-enabled architecture in edge computing has several security aspects that need to be considered to ensure the integrity, confidentiality, and availability of data and services. We divide the microservice security issues that arise during service communications into the following four aspects; containers, data, networks, and permissions. Fig. 7 shows the different microservice security problems in edge computing [56].

- **Container security:** Containers are used to package microservices and their dependencies. Ensuring container security involves scanning for vulnerabilities, implementing secure configurations, and using secure container registries to store and deploy containers to prevent the deployment of malicious or tampered containers.
- **Data security:** Data is a valuable asset in the microservices architecture. The exchange of information as data across services is fundamental to the success of any business. However, data introduces security challenges, particularly in edge computing, where data is generated and consumed at the edge of the network. Data security involves protecting data at rest and in transit, using encryption, and implementing access controls to ensure that only authorized entities can access and modify the data. Additionally, data must be stored securely, and secure data management practices must be implemented to ensure the integrity, confidentiality, and availability of data.
- **Network security:** Network security is critical in the microservices architecture as it involves the communication between microservices and devices. Network security involves implementing secure communication protocols, such as HTTPS and SSL, to ensure that data transmitted between microservices and devices is protected from eavesdropping and tampering. Additionally, secure network architectures must be implemented to prevent unauthorized access and protect against potential attacks.
- **Permission security:** Security might be guaranteed by using permissions. Ensuring permission security involves implementing access controls to ensure that only authorized entities can access and modify the microservices. This includes using secure authentication and authorization protocols, such as OAuth2 and OpenID Connect, to manage permissions.

There are several security and privacy-related difficulties when deploying microservices at the edge. It is essential to consider security aspects from the initial design stage and throughout the development, deployment, and operation phases of the microservices. In this section, we present a survey of security threats to microservice-based edge applications (see Table 7).

For cloud applications built on top of microservices, Sun et al. [74] suggested a new architecture for security as a service. The proposed framework consists of four main components: service description and deployment, service registry and discovery, service authentication and authorization, and service monitoring and management. A new API primitive called FlowTap is added to the network hypervisor to facilitate the creation of a network traffic monitoring and policy enforcement architecture that can ensure the safety of cloud-based software. The authors use Docker containers to deploy security services and demonstrate their effectiveness in protecting cloud applications against various security threats.

For services to be able to communicate and share data in a trustworthy, stateless manner, the microservices architecture must include suitable access control techniques and authorizations. Therefore, Kallergis et al. [75] investigated a real-world machine-to-machine (M2M) scenario that employs a hybrid cloud-based infrastructure and IoT services, discovering that combining policy-driven authorizations with independent, fine-grained microservices is the best approach. Using the microservices architecture, this system describes the authentication flows that allow communication among the various parties, offering a containerized authorization and policy-driven architecture named CAPODAZ. It aims to address the limitations of traditional monolithic architectures in terms of scalability, flexibility, and maintainability by providing a modular and distributed approach to policy enforcement.

The smart surveillance system, which is a key part of smart cities, brings up new concerns about data security. Traditional surveillance systems use a monolithic architecture to perform base-level tasks like monitoring and recording. But current surveillance systems should be able to handle more advanced analysis of video streams. This analysis is performed on a large number of distributed edge devices. Furthermore, a typical surveillance system's centralized architecture is prone to single points of failure and privacy breaches because the surveillance feed is unprotected. Nagothu et al. [76] presented a new, safe, smart surveillance system using blockchain technology. The use of blockchain technology ensures the security and privacy of data generated by smart surveillance devices. By enclosing the video analysis algorithms as separate microservices, the proposed scheme increases system availability and robustness without compromising data integrity.

Fog applications built with the microservices architecture are susceptible to attacks because of the design's loosely

Table 7
Security aspects of the microservice-enabled architecture.

Reference	Aspects	Contributions	Year published
[74]	Security as a service	The authors develop a policy-enforcement architecture for monitoring network traffic to ensure the safety of cloud-based software by using an API primitive called FlowTap.	2015
[75]	Container authorization	Using the microservices approach, the authors propose containerized authorization and a policy-driven management system called CAPODAZ	2020
[76]	Microservice-enabled blockchain technology	The authors introduce an innovative and safe smart surveillance system that is built on blockchain technology and the microservices architecture	2018
[77]	Security issues in microservices-enabled fog applications	The authors discuss different security problems in microservices communication, and suggest the best ways to fix them	2019
[78]	Security threats in container clouds	The authors propose a two-stage defence approach to protect cloud services from advanced attacks.	2017
[79]	A Docker container security policy	To improve container security, the authors propose a SELinux policy	2015
[80]	A microservice security agent in edge computing	The authors propose a microservice security agent that combines an API gateway with technology of an edge computing platform to provide a safe authentication mechanism	2019
[81]	Secure edge computing based on independent Microservices providers	The authors propose secure edge computing based on deployment of independent microservice providers in an edge gateway that communicates with a security gateway	2020

coupled, widely dispersed interfaces. However, its security flaws are not well known in the business world. Therefore, Yu et al. [77] looked at several security flaws that could affect fog applications built with microservices. They present a comprehensive analysis of various security threats, such as data confidentiality, integrity, availability, and privacy, and describe the security mechanisms used to address these threats. The paper also discusses the limitations of current security mechanisms and proposes potential future research directions for addressing these limitations.

In a multi-tenant container cloud service, multiple containers share a single operating system kernel. This raises some security concerns. Therefore, Gao et al. [78] highlight the emerging security threats of information leakage in container clouds and provide a comprehensive analysis of potential threats and attack scenarios. They discuss the root causes of these threats, including the insecure configuration of container orchestration systems, the vulnerabilities in the container runtime environment, and the lack of effective isolation mechanisms. The authors also present various countermeasures to mitigate these threats, such as container image scanning, access control, network segmentation, and intrusion detection.

Docker uses security features like cgroups, namespaces, and mandatory access control that are built into the Linux kernel to keep containers safe. In order to increase Docker security and flexibility, Bacis et al. [79] proposed an enhancement to the Docker file format to address the lack of fine-grained access control in Docker's default configuration. The authors introduce a new security policy framework and demonstrate its efficacy through the implementation of DockerPolicyModules. The proposed SELinux policy framework provides a flexible and effective solution for enforcing security policies in containerized environments.

Xu et al. [80] developed a microservice security agent that integrates the technology of an API gateway with the edge computing platform in order to offer a safe authentication mechanism. For implementation of the microservice security agent, they make use of a framework called EdgeX Foundry, which is based on the open-source Kong project. The security agent is responsible for enforcing security policies and monitoring the runtime behavior of microservices. The agent employs a lightweight container to provide security services such as authentication, authorization, and secure communication for microservices. Moreover, Jin et al. [81] proposed secure edge computing by utilizing an API gateway and microservices to achieve a flexible and secure edge computing management architecture. The proposed framework includes a security agent based on an API gateway to manage and enforce security policies, as well as a microservices repository to store and manage microservices. In order to perform security services, a security gateway is responsible for managing identification and authorization in order to keep a variety of microservices secure.

4. Application of microservices in edge computing

Microservices in edge computing have emerged as a promising technology for developing and deploying distributed applications. The role of microservices in edge computing can be understood by exploring their applications in different domains. This section analyzes the key applications of microservices in the edge computing framework.

- **IoT devices:** IoT devices generate massive amounts of data that need to be processed and analyzed in real-time. With the help of microservices, edge computing can enable the processing and analysis of data closer to where it is generated, thus reducing network latency and improving response times [62]. For instance, an edge microservice can be deployed on a local server

to perform the video streams in real-time. It can use computer vision algorithms to detect intruders, recognize faces, or perform object tracking. When an intrusion is detected, the microservice can autonomously activate security measures like sounding an alarm or locking doors [82].

- **Edge offloading:** Edge offloading is the process of transferring resource-intensive tasks from the end-user device to a more powerful computing resource at the edge of the network. Microservices play a key role in edge offloading by enabling developers to break down complex applications into smaller, independent services that can be distributed across the network [59]. For instance, a microservice running on an edge server or local GPU can offload the rendering tasks. The microservice can receive the augmented reality (AR) content and perform the rendering computations, delivering the rendered frames to the mobile application for display. This offloading reduces the computational load on the user's device and ensures a smooth AR experience [83].
- **E-commerce:** Microservices revolutionized the e-commerce. In e-commerce, microservice architecture can be used to create independent services for functions such as product catalogs, payment processing, and order fulfillment. With edge computing, these microservices can be deployed to the edge of the network, which is closer to the users, resulting in faster response times and improved performance. Overall, combining microservices and edge computing can offer several benefits to e-commerce applications, including improved scalability, agility, and user experience [84,85].
- **Smart cities:** Smart cities require a vast network of sensors, devices, and applications to collect, process, and analyze data from various sources. Microservices can help in the development and deployment of these applications by breaking them down into smaller, independent services that can be easily managed and scaled. Moreover, microservices can be deployed on edge devices to reduce latency and improve response times, enabling real-time monitoring and control of smart city infrastructure, such as traffic lights, adjusting traffic signal timings, providing alternative routes to alleviate congestion, waste management, real-time threat detection, parking meters, public transportation, and safety [86].
- **Industrial automation:** Microservices can be deployed on the edge to control and monitor industrial automation systems, such as robotic systems, improving productivity, detecting equipment failures, optimizing production schedules, and reducing downtime. For example, if a machine exceeds certain temperature thresholds, the edge microservice can trigger an alert, adjust parameters, and even shut down the machine to prevent damage [87].
- **Agriculture:** Microservices can be used to develop and deploy applications that monitor and control agricultural processes, such as irrigation and fertilization, improving crop yields, pest and disease detection, livestock monitoring, and reducing waste. For example, an edge microservice can determine optimal irrigation schedules, fertilizer application rates, and identify areas of the field that require attention [88].
- **Energy management:** Microservices can be used to monitor and control energy usage in real-time, reducing waste and improving energy efficiency as well as cost saving. For example, the microservice can identify peak demand periods, optimize energy usage based on demand forecasts, detect energy waste, and predict load fluctuations [89].
- **Autonomous vehicles:** Autonomous vehicles generate a massive amount of data that needs to be processed and analyzed in real-time. Microservices can be used to deploy autonomous vehicle services closer to the edge, allowing for faster processing and analysis of data. For example, an edge microservice running on each vehicle can analyze real-time traffic data, detect potential hazards, real-time navigation and route optimization services, detect accidents, and provide real-time feedback to the driver or the autonomous vehicle system. Moreover, if a vehicle exceeds certain speed limits or deviates from the planned route, the microservice can generate alerts [90].
- **Healthcare services:** Microservices can be used to develop and deploy healthcare applications on edge devices that process and analyze patient data in real-time, allowing for early detection of health issues, more efficient diagnosis and treatment, and quick response times. For example, the microservice can detect abnormal vital signs, trigger alerts to caregivers and medical professionals, analyze medical images, and provide remote consultation services [91].

5. Open research issues and future directions

To carry out diverse services, microservices are constantly being deployed on several network nodes. However, research on the utilization of microservices in edge computing environments is still emerging research area. A significant number of issues remain that need to be resolved effectively. Therefore, in this section, we propose a few open research challenges based on research gaps, and these open research issues can be expanded into future studies.

- **Blockchain as a service:** Blockchain technology can enhance the security of scattered and distributed microservice-based IoT applications by acting as a trusted third party. Its implementation offers various advantages, including improved trust, security, fast transaction processing, transparency, and strong encryption. However, traditional blockchain approaches have drawbacks like communication overhead, energy inefficiency, scalability challenges, and synchronization issues. To address these concerns, IoT applications can leverage blockchain as a service (BaaS), where different cryptography algorithms and smart contracts may be made available through a number of cloud providers and can be incorporated into IoT devices [92,93].

- **Integration with cloud services:** Edge computing and cloud computing are complementary technologies, and there is a need for better integration between the two. Future research efforts should concentrate on developing novel architectures and frameworks to seamlessly integrate microservices at the edge with cloud services. This includes exploring hybrid cloud architectures, edge-centric service discovery mechanisms, and cloud-based orchestration techniques [94].
- **Microservice orchestration platforms for the edge:** Many commercial platforms are available, such as Microsoft Azure, Google Cloud, and Amazon Web Services, that may be used for the assessment of scheduling strategies within cloud environments. In light of the fact that edge computing is still at the beginning of commercial adaptation, the research being conducted now involves small, custom-built testbeds. Unfortunately, there is a lack of support for large-scale experiments, and as a result, researchers are unable to adequately capture important aspects associated with the microservices architecture such as reliability, security, load balancing, distribution, and location-aware deployment. To address this gap, it is crucial to develop scalable and adaptable container orchestration systems specifically designed for research purposes [95].
- **Edge-to-Edge communication:** Efficient data processing and decision-making in edge computing heavily rely on effective communication among edge devices. However, existing communication protocols and technologies may not adequately handle the substantial data volumes generated by these devices. Future research endeavors could concentrate on innovating new communication protocols and technologies capable of meeting the demanding requirements of high bandwidth and low latency for edge-to-edge communication. This research area may explore decentralized communication approaches, peer-to-peer communication models, and edge-based routing protocols to enhance the efficiency of communication in edge computing environments [96].
- **AI/ML-based solutions:** In the context of the IoT, AI and ML have proven to be effective in analyzing data acquired and stored in edge infrastructures, enabling rapid decision-making and accurate predictions. Although ML approaches are employed for a range of application domains, the use of these techniques in microservice-based systems is still in its early stages. In this regard, there is a significant window of opportunity to incorporate intelligence into the various tiers of microservice applications. Therefore, the development of AI-based micro-algorithms as services (MAaS) at the network edge remains an ongoing research challenge [56].
- **Resource allocation:** Resource allocation is a critical issue in microservices architecture, especially in the edge environment where resource constraints are prevalent. Future research could focus on developing resource allocation techniques that can dynamically allocate resources to microservices based on the available resources and workload. Potential approaches to explore include containerization for efficient resource allocation, leveraging edge caching for optimized resource utilization, and employing machine learning for dynamic resource allocation [97].
- **DevOps and Self-adaptive deployment:** It is becoming more difficult to deploy application updates or microservices in production as an increasing number of businesses adopt DevOps tools and methods, such as constructing agile cloud-native systems, continuous delivery, and integration. However, there is a limited focus from researchers on utilizing DevOps technologies to enable self-adaptation capabilities in microservice-based systems. This might open the door to a wide variety of opportunities for the automation of deployments and the development of new self-adaptive systems for microservices [98,99].
- **Unified frameworks:** The use of containers has simplified the construction of algorithms as microservices. This allows code to be developed in any programming language, and these microservices can be seamlessly unified through an API. By executing algorithms as microservices, code interoperability, composability, and independence are ensured. Therefore, a comprehensive framework that combines all of these ideas is needed to resolve interoperability challenges.
- **Scalability:** One of the key challenges in microservices architecture is scalability. As more and more edge devices are added to networks, there will be a need for more scalable microservices architectures that can handle the increasing demand. Future research could focus on developing new approaches to scaling microservices at the edge, such as distributed caching, load balancing, and auto-scaling [100].
- **Security challenges:** The distributed deployment of microservices across edge clouds creates a substantial security threat to the processing and transmission of sensitive data. To address this concern, it is crucial to develop lightweight cryptosecure and placement algorithms, privacy-preserving techniques, and secure data sharing mechanisms. These advancements are necessary to enhance security measures, detect intrusions, and mitigate threats while making decisions regarding microservice deployment [101].
- **Edge service discovery:** The dynamic nature of edge devices and services makes it challenging to discover and utilize them efficiently. As the number of edge devices increases, the discovery of available services becomes more complex. Future research could focus on developing novel algorithms and techniques for edge service discovery that can efficiently discover and utilize edge services [102].

6. Conclusion

With continuous development of the IoT, monolithic applications are getting much bigger, have poor extensibility

and scalability, and even their structures are becoming more complicated. To overcome these challenges, the microservices architecture was introduced owing to its adaptability, lightweight nature, and loose coupling. On the other hand, the resource-constrained nature of edge computing necessitates the development of edge applications as a collection of lightweight and interconnected modules. Since this idea is in line with the goals of the microservices architecture, effectively implementing microservice-based edge applications has the potential to fully utilize the capabilities of edge nodes. However, the design of microservices poses a variety of challenges, and the use of microservices in edge computing is an emerging research area with many aspects yet to be explored and developed, which is the main motivation of this paper.

The microservices approach in edge computing presents significant opportunities for enhancing the performance and scalability of edge computing systems. In this paper, recent advancements in the various strategies for deploying microservices within an edge computing platform were investigated. We first presented a comprehensive overview of edge computing, explained microservices technologies and tools, and looked at the benefits and difficulties of delivering microservices at the network edge. Then, several research topics on the use of microservices in edge computing were introduced. After that, we presented a comprehensive survey of different issues in state-of-the-art research, including microservice-based applications in edge computing environments, the AI-based microservices framework, and the security concerns of the microservice-enabled architecture in edge computing. The paper has identified potential applications of microservices in various domains. Finally, we discussed and highlighted several unresolved issues as well as future research directions for this hot topic.

Declaration of competing interest

The authors declare that there is no conflict of interest in this paper.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00047, Development of Microservices Development/Operation Platform Technology that Supports Application Service Operation Intelligence) and (No.RS-2023-00220631, Edge Cloud Reference Architecture Standardization for Low Latency and Lightweight Cloud Service).

References

[1] P.P. Ray, A survey on internet of things architectures, J. King Saud Univ.-Comput. Inf. Sci. 30 (3) (2018) 291–319.

[2] D. Aishwarya, R.I. Minu, Edge computing based surveillance framework for real time activity recognition, ICT Express 7 (2) (2021) 182–186.

[3] H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: Architecture, applications, and approaches, Wireless Commun. Mob. Comput. 13 (18) (2013) 1587–1611.

[4] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog computing: A platform for internet of things and analytics, in: Proceedings of Big Data and Internet of Things: A Roadmap for Smart Environments, Springer, Jeju Island, Korea, 2014, pp. 169–186.

[5] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, B. Amos, Edge analytics in the internet of things, IEEE Pervasive Comput. 14 (2) (2015) 24–31.

[6] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tut. 19 (4) (2017) 2322–2358.

[7] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, IEEE Commun. Surv. Tut. 19 (3) (2017) 1657–1681.

[8] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet Things J. 5 (1) (2018) 450–465.

[9] H. Sabireen, V. Neelanarayanan, A review on fog computing: Architecture, fog with IoT, algorithms and research challenges, ICT Express 7 (2) (2021) 162–176.

[10] M. Ishtiaq, N. Saeed, M.A. Khan, Edge computing in IOT: A 6 g perspective, 2021, arXiv preprint arXiv:2111.08943.

[11] H.C. Hsieh, J.L. Chen, A. Benslimane, 5G virtualized multi-access edge computing platform for IoT applications, J. Network Comput. Appl. 115 (2018) 94–102.

[12] K. Wang, Y. Wang, Y. Sun, S. Guo, J. Wu, Green industrial internet of things architecture: An energy-efficient perspective, IEEE Commun. Mag. 54 (12) (2016) 48–54.

[13] M. Liyanage, P. Porambage, A.Y. Ding, A. Kalla, Driving forces for multi-access edge computing (MEC) IoT integration in 5G, ICT Express 7 (2) (2021) 127–137.

[14] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D.S. Nikolopoulos, Challenges and opportunities in edge computing, in: Proceedings of 2016 IEEE International Conference on Smart Cloud, SmartCloud, IEEE, 2016, pp. 20–26.

[15] I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen, Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly Media, Inc., 2016.

[16] L. Bao, C. Wu, X. Bu, N. Ren, M. Shen, Performance modeling and workflow scheduling of microservice-based applications in clouds, IEEE Trans. Parallel Distrib. Syst. 30 (9) (2019) 2114–2129.

[17] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, X. Shen, Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach, IEEE Trans. Mob. Comput. 20 (3) (2021) 939–951.

[18] M. Alam, J. Rufino, J. Ferreira, S.H. Ahmed, N. Shah, Y. Chen, Orchestration of microservices for IoT using docker and edge computing, IEEE Commun. Mag. 56 (9) (2018) 118–123.

[19] W. Shi, X. Zhang, Y. Wang, Q. Zhang, Edge computing: State-of-the-art and future directions, J. Comput. Res. Dev. 56 (1) (2019) 69–89.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.

[21] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.

[22] Z. Zhao, F. Liu, Z. Cai, N. Xiao, Edge computing: Platforms, applications and challenges, J. Comput. Res. Dev. 55 (2) (2018) 327–337.

[23] X. Hong, Y. Wang, Edge computing technology: Development and measures, Strategic Study Chin. Acad. Eng. 20 (2) (2018) 1–7.

[24] X. Sun, N. Ansari, EdgeIoT: Mobile edge computing for the internet of things, IEEE Commun. Mag. 54 (12) (2016) 22–29.

[25] P. Ranaweera, C. de Alwis, A.D. Jurcut, M. Liyanage, Realizing contact-less applications with multi-access edge computing, ICT Express 8 (2022) 575–587.

[26] A.R. Sajun, S. Shapsough, I. Zualkernan, R. Dhaouadi, Edge-based individualized anomaly detection in large-scale distributed solar farms, ICT Express 8 (2) (2022) 174–178.

[27] M.D. Hossain, T. Sultana, M.A. Hossain, M.I. Hossain, L.N.T. Huynh, J. Park, E.-N. Huh, Fuzzy decision-based efficient task offloading management scheme in multi-tier MEC-enabled networks, Sensors 21 (4) (2021) p1484.

[28] A. Alrawais, A. Alhothaily, C. Hu, X. Cheng, Fog computing for the internet of things: Security and privacy issues, IEEE Internet Comput. 21 (2) (2017) 34–42.

[29] M.D. Hossain, T. Sultana, M.A. Hossain, M.A. Layek, M.I. Hossain, P.P. Sone, G.W. Lee, E.N. Huh, Dynamic task offloading for cloud-assisted vehicular edge computing networks: A non-cooperative game theoretic approach, Sensors 22 (10) (2022) p3678.

[30] J. Kang, R. Yu, X. Huang, Y. Zhang, Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles, IEEE Trans. Intell. Transp. Syst. 19 (8) (2018) 2627–2637.

[31] Khan, et al., A survey on mobile edge computing for video streaming: Opportunities and challenges, IEEE Access 10 (2022) 120514-120550.

[32] S. Wang, J. Xu, N. Zhang, Y. Liu, A survey on service migration in mobile edge computing, IEEE Access 6 (2018) 23511–23528.

[33] C. Surianarayanan, G. Ganapathy, R. Pethuru, Essentials of Microservices Architecture: Paradigms, Applications, and Techniques, Taylor & Francis, 2019, http://dx.doi.org/10.1201/9780429329920.

[34] B. Christudas, Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud, A Press, 2019.

[35] L. Sun, Y. Li, R.A. Memon, An open IoT framework based on microservices architecture, China Commun. 14 (2) (2017) 154–162.

[36] A. Benayache, A. Bilami, S. Barkat, P. Lorenz, H. Taleb, MsM: A microservice middleware for smart WSN-based IoT application, J. Network Comput. Appl. 144 (2019) 138–154, http://dx.doi.org/10.1016/j.jnca.2019.06.015.

[37] J.L. Ribeiro, M. Figueredo, A. Araujo, N. Cacho, F. Lopes, A microservice based architecture topology for machine learning deployment, in: Proceedings of 2019 IEEE International Smart Cities Conference, ISC2, IEEE, 2019, pp. 426–431.

[38] A. Power, G. Kotonya, A microservices architecture for reactive and proactive fault tolerance in IoT systems, in: Proceedings of 2018 IEEE 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM, IEEE, 2018, pp. 588–599.

[39] G. Blinowski, A. Ojdowska, A. Przybyłek, Monolithic vs. microservice architecture: A performance and scalability evaluation, IEEE Access 10 (2022) 20357–20374.

[40] S. Pallewatta, V. Kostakos, R. Buyya, Microservices-based IoT applications scheduling in edge and fog computing: A taxonomy and future directions, 2022, arXiv preprint arXiv:2207.05399.

[41] R. Chen, S. Li, Z. Li, From monolith to microservices: A dataflow-driven approach, in: Proceedings of 2017 24th Asia-Pacific Software Engineering Conference, APSEC, IEEE, 2017, pp. 466–475, http://dx.doi.org/10.1109/APSEC.2017.53.

[42] S. Li, H. Zhang, Z. Jia, Z. Li, C. Zhang, J. Li, Q. Gao, J. Ge, Z. Shan, A dataflow-driven approach to identifying microservices from monolithic applications, J. Syst. Software 157 (2019) 110380.

[43] H.H.S. da. Silva, G. de F. Carneiro, M.P. Monteiro, An experience report from the migration of legacy software systems to microservice based architecture, in: Proceedings of the 16th International Conference on Information Technology-New Generations, ITNG 2019, Springer, 2019, pp. 183–189.

[44] S. Eski, F. Buzluca, An automatic extraction approach: Transition to microservices architecture from monolithic application, in: Proceedings of 19th International Conference on Agile Software Development: Companion, 2018, pp. 1–6.

[45] J. Löhnertz, A. Oprescu, Steinmetz: Toward automatic decomposition of monolithic software into microservices, in: Proceedings of the Seminar Series on Advanced Techniques & Tools for Software Evolution, SATToSE, 2020, pp. 1–8.

[46] O. Al-Debagy, P. Martinek, Extracting microservices' candidates from monolithic applications: Interface analysis and evaluation metrics approach, in: Proceedings of 2020 IEEE 15th International Conference of System of Systems Engineering, SoSE, IEEE, 2020, pp. 289–294.

[47] D. Taibi, K. Systa, A decomposition and metric-based evaluation framework for microservices, in: Proceedings of the International Conference on Cloud Computing and Services Science, Springer, 2019, pp. 133–149.

[48] Y. Zhang, B. Liu, L. Dai, K. Chen, X. Cao, Automated microservice identification in legacy systems with functional and non-functional metrics, in: Proceedings of 2020 IEEE International Conference on Software Architecture, ICSA, IEEE, 2020, pp. 135–145.

[49] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, Q. Zheng, Service candidate identification from monolithic systems based on execution traces, IEEE Trans. Software Eng. 47 (5) (2021) 987–1007.

[50] M. Brito, J. Cunha, J. Saraiva, Identification of microservices from monolithic applications through topic modelling, in: Proceedings of 36th Annual ACM Symposium on Applied Computing, 2021, pp. 1409–1418.

[51] P. Jamshidi, C. Pahl, N.C. Mendonça, J. Lewis, S. Tilkov, Microservices: The journey so far and challenges ahead, IEEE Software 35 (3) (2018) 24–35.

[52] J. Soldani, D.A. Tamburri, W.-J.V.D. Heuvel, The pains and gains of microservices: A systematic grey literature review, J. Syst. Software 146 (2018) 215–232.

[53] R. Capuano, H. Muccini, A systematic literature review on migration to microservices: ?a quality attributes perspective, in: Proceedings of 2022 IEEE 19th International Conference on Software Architecture Companion, ICSA-C, 2022, pp. 120–123.

[54] M. Usman, S. Ferlin, A. Brunstrom, J. Taheri, A survey on observability of distributed edge & container-based microservices, IEEE Access 10 (2022) 86904–86919.

[55] D. Yu, Y. Jin, Y. Zhang, X. Zheng, A survey on security issues in services communication of microservices-enabled fog applications, Concurrency Comput.: Pract. Exp. 31 (22) (2019) e4436.

[56] F. Al-Doghman, N. Moustafa, I. Khalil, Z. Tari, A. Zomaya, AI-enabled secure microservices in edge computing: Opportunities and challenges, IEEE Trans. Serv. Comput. 16 (2) (2023) 1485–1504.

[57] J. Gedeon, M. Wagner, J. Heuschkel, L. Wang, M. Muhlhauser, A microservice store for efficient edge offloading, in: Proceedings of 2019 IEEE Global Communications Conference, GLOBECOM, IEEE, 2019, pp. 1–6.

[58] A. Samanta, J. Tan, Dyme: Dynamic microservice scheduling in edge computing enabled IoT, IEEE Internet Things J. 7 (7) (2020) 6164–6174.

[59] X. Zhao, C. Huang, Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network, IEEE Access 8 (2020) 56680–56694.

[60] A. Kwan, J. Wong, H.-A. Jacobsen, V. Muthusamy, Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres, in: Proceedings of 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, IEEE, Dallas, TX, USA, 2019, pp. 80–90.

[61] L. Gu, Z. Chen, H. Xu, D. Zeng, B. Li, H. Jin, Layer-aware collaborative microservice deployment toward maximal edge throughput, in: Proceedings of IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, IEEE, London, United Kingdom, 2022, pp. 71–79.

[62] K. Alanezi, S. Mishra, Utilizing microservices architecture for enhanced service sharing in IoT edge environments, IEEE Access 10 (2022) 90034–90044.

[63] Chen, et al., Dynamic service migration and request routing for microservice in multicell mobile-edge computing, IEEE Internet Things J. 9 (15) (2022) 13126–13143.

[64] H. Zhao, S. Deng, Z. Liu, J. Yin, S. Dustdar, Distributed redundant placement for microservice-based applications at the edge, IEEE Trans. Serv. Comput. 15 (3) (2022) 1732–1745.

[65] G.M. Lee, T.-W. Um, J.K. Choi, AI as a microservice (AIMS) over 5G networks, in: Proceedings of 2018 ITU Kaleidoscope: Machine Learning for A 5G Future, ITU K, IEEE, 2018, pp. 1–7.

[66] J.L. Ribeiro, M. Figueredo, A. Araujo, N. Cacho, F. Lopes, A microservice based architecture topology for machine learning deployment, in: Proceedings of 2019 IEEE International Smart Cities Conference, ISC2, 2019, pp. 426–431.

[67] H. Tian, X. Xu, T. Lin, Y. Cheng, C. Qian, L. Ren, M. Bilal, DIMA: Distributed cooperative microservice caching for internet of things in edge computing by deep reinforcement learning, World Wide Web 25 (5) (2021) 1769–1792.

[68] Chen, et al., IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning, IEEE Internet Things J. 8 (16) (2021) 12610–12622.

[69] K. Fu, W. Zhang, Q. Chen, D. Zeng, M. Guo, Adaptive resource efficient microservice deployment in cloud-edge continuum, IEEE Trans. Parallel Distrib. Syst. 33 (8) (2021) 1825–1840.

[70] D.C. Li, C.-T. Huang, C.-W. Tseng, L.-D. Chou, Fuzzy-based microservice resource management platform for edge computing in the internet of things, Sensors 21 (11) (2021) 3800.

[71] W. Lv, Q. Wang, P. Yang, Y. Ding, B. Yi, Z. Wang, C. Lin, Microservice deployment in edge computing based on deep Q learning, IEEE Trans. Parallel Distrib. Syst. 33 (11) (2022) 2968–2978.

[72] F. Guo, B. Tang, M. Tang, W. Liang, Deep reinforcement learning-based microservice selection in mobile edge computing, Cluster Comput. (2022) 1–17.

[73] A.A. Khaleq, I. Ra, Intelligent autoscaling of microservices in the cloud for real-time applications, IEEE Access 9 (2021) 35464–35476.

[74] Y. Sun, S. Nanda, T. Jaeger, Security-as-a-service for microservices-based cloud applications, in: Proceeding of 2015 IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom, 2015, pp. 50–57.

[75] D. Kallergis, Z. Garofalaki, G. Katsikogiannis, C. Douligeris, CAPODAZ: A containerised authorisation and policy-driven architecture using microservices, in: Proceeding of 2015 IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom, 2020, pp. 1–12.

[76] D. Nagothu, R. Xu, S.Y. Nikouei, Y. Chen, A microservice-enabled architecture for smart surveillance using blockchain technology, in: Proceeding of 2018 IEEE International Smart Cities Conference, ISC2, 2018, pp. 1–4.

[77] D. Yu, Y. Jin, Y. Zhang, X. Zheng, A survey on security issues in services communication of microservices-enabled fog applications, Concurr. Comput.: Pract. Exper. 31 (22) (2019) e4436.

[78] Gao, et al., Containerleaks: Emerging security threats of information leakages in container clouds, in: Proceeding of 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN, 2017, pp. 237–248.

[79] E. Bacis, S. Mutti, S. Capelli, S. Paraboschi, DockerPolicyModules: Mandatory access control for docker containers, in: Proceeding of 2015 IEEE Conference on Communications and Network Security, CNS, 2015, pp. 2749–750.

[80] R. Xu, W. Jin, D. Kim, Microservice security agent based on API gateway in edge computing, Sensors 19 (22) (2019) 4905.

[81] W. Jin, R. Xu, T. You, Y.-G. Hong, D. Kim, Secure edge computing management based on independent microservices providers for gateway-centric IoT networks, IEEE Access 8 (2020) 187975-187990.

[82] C.H. Chen, C.T. Liu, Person re-identification microservice over artificial intelligence internet of things edge computing gateway, Electronics 10 (18) (2021) 1–23.

[83] P. Ren, X. Qiao, J. Chen, S. Dustdar, Mobile edge computing–A booster for the practical provisioning approach of web-based augmented reality, in: 2018 IEEE/ACM Symposium on Edge Computing, SEC, 2018, pp. 349–350.

[84] M. Gördesli, A. Varol, Comparing interservice communications of microservices for E-commerce industry, in: 2022 10th International Symposium on Digital Forensics and Security, ISDFS, Istanbul, Turkey, 2022, pp. 1–4.

[85] Ż. Li, et al., Noah: Reinforcement-learning-based rate limiter for microservices in large-scale E-commerce services, IEEE Trans. Neural Netw. Learn. Syst. http://dx.doi.org/10.1109/TNNLS.2023.3264038.

[86] A. Krylovskiy, M. Jahn, E. Patti, Designing a smart city internet of things platform with microservice architecture, in: Proceeding of 3rd International Conference on Future Internet of Things and Cloud, Rome, Italy, 2015, pp. 25–30.

[87] J. Dobaj, J. Iber, M. Krisper, C. Kreiner, A microservice architecture for the industrial internet-of-things, in: Proceeding of 23rd European Conference on Pattern Languages of Programs, 2018, pp. 1–15.

[88] S. Trilles, A. González-Pérez, J. Huerta, An IoT platform based on microservices and serverless paradigms for smart farming purposes, Sensors 20 (8) (2020) p2418.

[89] M. Parvizi, J. Bahrami, M. Noei, M. Yalpanian, An IoT platform based on microservices architecture for energy power management, in: Proceeding of Fifth International Conference on Technology Development in Iranian Electrical Engineering, Tehran, 2021.

[90] B. Liu, V.P. Betancourt, Y. Zhu, J. Becker, Towards an on-demand redundancy concept for autonomous vehicle functions using microservice architecture, in: Proceeding of 2020 IEEE International Symposium on Systems Engineering, ISSE, Vienna, Austria, 2020, pp. 1–5.

[91] W. Sriborrirux, P. Laortum, Healthcare center IoT edge gateway based on containerized microservices, in: Proceeding of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, 2020, pp. 24–29.

[92] L. Daming, L. Deng, Z. Cai, A. Souri, Blockchain as a service models in the internet of things management: Systematic review, Trans. Emerg. Telecommun. Technol. 33 (4) (2020) e4139.

[93] M. Driss, D. Hasan, W. Boulila, J. Ahmad, Microservices in IoT security: Current solutions, research challenges, and future directions, Procedia Comput. Sci. 192 (2021) 2385–2395.

[94] L. Roda-Sanchez, et al., Cloud–edge microservices architecture and service orchestration: An integral solution for a real-world deployment experience, Internet Things 22 (2023) p100777.

[95] E. Casalicchio, S. Iannucci, The state-of-the-art in container technologies: Application, orchestration and security, Concurr. Comput.: Prac. Exp. 32 (17) (2020) e5668.

[96] K. Fu, W. Zhang, Q. Chen, D. Zeng, M. Guo, Adaptive resource efficient microservice deployment in cloud-edge continuum, IEEE Trans. Parallel Distrib. Syst. 33 (8) (2022) 1825–1840.

[97] C.T. Joseph, K. Chandrasekaran, IntMA: Dynamic interaction-aware resource allocation for containerized microservices in cloud environments, J. Syst. Archit. 111 (2020) p101785.

[98] D. Trihinas, A. Tryfonos, M.D. Dikaiakos, G. Pallis, Devops as a service: Pushing the boundaries of microservice adoption, IEEE Internet Comput. 22 (3) (2018) 65–71.

[99] E. Pimentel, W. Pereira, P.H.M. Maia, M.I. Cortés, Self-adaptive microservice-based systems-landscape and research opportunities, in: Proceeding of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS, 2021, pp. 167–178.

[100] G. Márquez, M.M. Villegas, H. Astudillo, An empirical study of scalability frameworks in open source microservices-based systems, in: Proceeding of the 2018 37th International Conference of the Chilean Computer Science Society, SCCC, Santiago, Chile, 2018, pp. 1–8.

[101] S. Sultan, I. Ahmad, T. Dimitriou, Container security: Issues, challenges, and the road ahead, IEEE Access 7 (2019) 52976–52996.

[102] N. Singh, et al., Load balancing and service discovery using docker swarm for microservice based big data applications, J. Cloud Comput. 12 (1) (2023) 1–9.