Survey of Function Offloading and Serverless Functions in the Computing Continuum

Maciej Kozub
Vrije Universiteit Amsterdam
m.p.kozub@student.vu.nl

Matthijs Jansen
Vrije Universiteit Amsterdam
m.s.jansen@vu.nl

Daniele Bonetta
Vrije Universiteit Amsterdam
d.bonetta@vu.nl

Abstract

Serverless computing has revolutionized application development by abstracting infrastructure management from application development, enabling developers to focus on application logic. Since the inception of serverless computing with AWS Lambda, serverless technology has seen rapid adoption across major cloud platforms. Besides application development, serverless computing has had a significant impact on how applications are offloaded between cloud and edge in the computing continuum. To better understand how and why serverless computing impacts function offloading, a study of the driving factors behind serverless offloading and the impacted application domains is essential. However, existing surveys on function offloading application domains do not study the impact of serverless technology. This systematic literature survey seeks to bridge this knowledge gap by investigating and underlining serverless functions' role in reshaping the function offloading paradigm within the modern computing continuum.

1 Introduction

Serverless is a computing model that decouples infrastructure management from application code development [35]. It allows developers to focus on delivering the application logic and not managing servers, networks, and cluster configurations [34]. Function-as-a-service (FaaS) [82] or cloud functions [78] build upon serverless technology and introduce further abstraction. In the FaaS model, developers break the application into individual functions that are later executed by stateless, ephemeral workers [85]. Introduced in 2014 by the introduction of Amazon Lambda [1] and followed by Google and Microsoft in 2016, the serverless adoption rate is significant. According to Datadog's analysis [27], more than 70% of organizations using AWS and over 50% of those using Azure and Google Cloud have adopted serverless in June 2022. One year before, this was only above 50%, 35%, and 20%, respectively [26]. This trend is believed to continue, and it is predicted that "Serverless will dominate the future of cloud computing" [44].

Serverless functions have evolved beyond the cloud and are often used for function offloading because of their flexibility and scalability [49]. Function offloading is the general pattern of transferring resource-heavy tasks from local resource-constrained devices to distribute workloads [17] [54]. The primary concern of function offloading is shaped by the need to balance computational, networking, or storage demands with resource constraints. Function offloading allows for agile advancements in the computing continuum [59]. The computing continuum integrates cloud, edge computing, and IoT into a unified system that spans from centralized data centers to user devices such as mobile phones. The introduction of a serverless computing model and FaaS platforms is recognized as a significant milestone in function offloading development [81]. Serverless is a key technology for offering function offloading with better scalability, easier deployment, and lower costs. Thanks to the objectives of serverless functions new application domains arise for function offloading.

To fully understand the importance of serverless for function offloading, the exploration of application domains, and the trends and objectives driving the adoption of function offloading and FaaS is needed. By studying the application domains, we gain insight into how serverless functions extend the variety of use cases for function offloading. By exploring the trends and objectives driving those new use cases, we can better understand the current computing landscape and try to predict further development directions. While there are a significant number of surveys on task offloading [28] [86] or serverless functions [35] [82], they do not extensively cover the topic of use cases and drivers behind function offloading adoption. Moreover, none of the found surveys considers the serverless functions' impact on function offloading application domains. Instead, they often have a more narrowed focus, like task-scheduling approaches [55] [86] or offloading decision-making algorithms [86] [2].

We aim to fill this knowledge gap by studying the landscape of function offloading and how it is influenced by serverless

Survey	Use Cases	Objectives	Serverless	Non-serverless	Use cases												Objectives									
					AR/VR	Health	HPC/Scientific	IIoT	Media Processing	ML/AI	Mobile Phones	Sensor Networks	Smart City	Smart Grid	Vehicles/UAVs	VNF	Availability	Compute Power	Cost	Energy Efficiency	Event Driven	Infrastructure	Latency	Parallel Computing	Resource Utilization	Scaling
[54]	✓		✓	✓						✓	✓		✓	✓	✓											
[6]		✓	✓														✓		✓	✓	✓			✓	✓	✓
[76]	✓	✓	✓			✓	✓	✓	✓	✓	✓		✓		✓				✓	✓		✓			✓	✓
[42]		✓		✓				✓	✓		✓				✓											
[35]	✓	✓	✓						✓		✓			✓		✓			✓			✓	✓			✓
[30]		✓	✓														✓	✓	✓		✓	✓	✓			✓
[13]	✓		✓		✓	✓		✓	✓	✓		✓	✓		✓											
[58]		✓	✓														✓		✓		✓	✓		✓		✓
Us	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Overview comparison of function offloading use cases and objectives covered in related works and this survey.

functions. We capture the current state of the art in the field of application domains and factors driving the adoption of function offloading and FaaS. To achieve this, we formulate and answer the following questions: (I) *what are the primary use cases and application domains of function offloading and serverless functions?*; and (II) *what are the main objectives driving the adoption of function offloading and serverless functions?*. We organize the contributions of this survey as follows:

1. We delve into related surveys, outlining their main contributions and comparing them to this survey in Section 2. We identify gaps in prior work and design a new systematic literature survey. We present our methodology in Section 3.

2. We construct an overview of application domains for function offloading including serverless functions (Section 4) and we structure the identified domains and use cases into a taxonomy to provide a better understanding of their diversity. This provides an insight into the current state of function offloading and serverless function usage across the computing continuum.

3. We present the main trends in computing that drive the adoption of function offloading in Section 5. We showcase what processes influence the rapid growth of function offloading, including serverless functions as computation models. We also provide an insight into how the identified trends are behind the rise of particular use cases.

4. We present and group the main objectives and characteristics that drive the adoption of function offloading including serverless functions into a taxonomy in Section 6. We couple objectives with specific use cases to better understand the driving factors of function offloading. This contribution fulfills the image of function offloading and serverless functions driving factors, showing how current application domains benefit from characteristics and how those characteristics make use cases possible.

2 Related Work

Existing surveys targeting function offloading in the computing continuum rarely include a broad discussion of use cases alongside the objectives and the benefits of using function offloading. Even fewer studies correlate use cases and their characteristics to understand the objectives driving particular use cases. Instead, they often focus on more detailed aspects like task scheduling algorithms, offloading decision-making processes, or systems' performance optimization.

We present an overview of how related works cover use cases that leverage offloading and their objectives and compare it with our survey in Table 1. In the table we show whether related studies discuss use cases and objectives driving function offloading; we also indicate what the main focus of such a study is: serverless functions or function offloading to non-serverless infrastructure. The next two parts of the table indicate which particular use cases and objectives, if any, are discussed in each survey. For reference, the last row of the table refers to our survey. Surveys are ordered by their number of citations on Google Scholar. For each, we provide a more detailed summary and a comparison to our work below.

Lin et al. [54] present an in-depth analysis of the history of computation offloading and the recent shift of offloading towards edge computing. The paper discusses various aspects, such as offloading granularity, architecture, challenges, application scenarios as shown in Table 1, and future directions. It also presents serverless as a promising and emerging execution platform for computation offloading. This survey does

not discuss any objectives of offloading (Table 1).

Aslanpour et al. [6] contribute to exploring the integration of serverless computing with edge computing to address modern application demands. The survey also considers the benefits of task offloading to the cloud. As presented, the edge provides a seamless transition layer before cloud resources are available. However, function offloading is not the main focus of this survey, and it covers a limited number of drivers for adoption and use cases, as can be seen in Table 1.

Shafiei et al. [76] contribute to identifying and describing potential opportunities, application domains, and challenges of serverless functions. It also explores how the FaaS paradigm shifts the traditional cloud computing model towards a more granular, event-driven architecture. This survey, however, does not consider the objectives of cloud functions from the perspective of end-user devices.

Islam et al. [42] primarily focus on classifying the multi-access edge computing (MEC) architecture and on strategies of task offloading within the MEC. The survey delves into the benefits of offloading to MEC. Another contribution of this survey is the six groups of use cases that were identified and presented. This survey takes a more generic approach and does not focus solely on function offloading.

Hassan et al. [35] systematically review the literature on serverless computing, addressing its state-of-the-art, concepts, platforms, usage, and challenges. It identifies the benefits of serverless computing and discusses future research directions. The key contribution of this survey is the very precisely classified characteristics of serverless functions and their application areas.

Eismann et al. [30] main contribution is a comprehensive analysis of serverless computing use cases. It presents and characterizes 89 different use cases sourced from literature. Key findings highlight the dominance of AWS as a platform and its significant adoption in production environments. The analysis also reveals diverse application characteristics, workload patterns, and the motivations behind serverless adoption.

Cassel et al. [13] conduct a broad study of the adoption of FaaS and serverless computing in IoT devices. The survey comprehensively studies the related literature and provides insights into architectures, programming languages, and protocols used in the serverless IoT world. Cassel et al. contribute to identifying six groups of use cases, including further offloading of serverless functions to other layers in the computing continuum. This survey does not present the objectives of function offloading.

Manner [58] presents an analysis to clarify the concepts of serverless computing and FaaS. The key contributions of this survey are the identified characteristics and benefits of FaaS; however, use cases are not considered.

Based on the related studies presented above, we find that our work fills the existing knowledge gap of a comprehensive study of application domains and factors driving the adoption of function offloading and serverless functions as well. Existing surveys do not combine non-serverless function offloading with serverless functions, which we do. There is also a lack of comprehensive exploration of a landscape of function-offloading application domains, use cases, and key drivers of its adoption that we provide with this survey.

3 Survey Methodology

In this section, we present the approach and methodology followed to execute this systematic literature survey. Firstly, we delve into keyword selection and search for literature. Next, we present an overview of the adopted selection criteria for evaluating found publications. We conclude this section with a description of the data extraction, analysis, and taxonomy-forming process.

3.1 Literature Search

In the process of searching for relevant literature, we started with defining the initial set of keywords that are later used to build the search queries. We looked into an initial set of highly-cited papers [17] [69] and one survey [54] in the field. Those publications helped us to grasp the current state of the research as well as the vocabulary used in the area of function offloading in the computing continuum. Based on this, we concluded that the fields of computation offloading and serverless functions are seldom considered together, which is important for fully understanding ongoing advancements in function offloading. We defined the following keywords together with their synonyms: *function*, *offloading*, *computing*, *faas*, *serverless*, *cloud*, *edge*, *fog*, and *continuum*. Based on the keywords we also defined phrases that consist of more than one keyword based on the context of this survey: *funtcion offloading*, *computing offloading*, and *funtion as a service*.

In this survey, we focus on electronic databases, and for searching purposes, we use Google Scholar, as it is commonly used for conducting literature surveys in software engineering [92] and is believed to be one of the most comprehensive sources of publications [60]. An initial title search is conducted to include all papers that might be relevant for this survey. Below, we present the initial search query that was constructed and executed on the Google Scholar database.

```
allintitle:
("function offloading" OR "computing offloading" OR
faas OR "function as a service" OR serverless)
(continuum OR cloud OR edge OR fog)
```

Besides the database search, we also conduct a snowball search. This method uses the reference list of accepted publications to identify other relevant works in the field that were not found using the query search. This flexible approach to systemic search may increase the research value of this survey since it might include relevant literature with unusual titles that would not be included otherwise.

3.2 Selection Criteria

The initial query retrieved 1670 papers, so further assessment to decide whether a publication is relevant and should be in the scope of this survey was required. To realize this goal, some selection criteria have been established and employed, as follows below.

Inclusion criteria:

- *I1* - Publication date is not earlier than 2019. The rationale is that we only want to consider the most recent drivers and application domains.
- *I2* - Publication directly addresses function offloading or serverless functions in the title or abstract.
- *I3* - Publication directly considers one or many application domains or use cases and one or many function offloading objectives in its title or abstract. To evaluate this, we searched for the following key phrases: *characteristics*, *application domains*, *use cases*, *benefits*.
- *I4* - Publication is developed by either academia or practitioners.

Exclusion criteria:

- *E1* - Publication full text is not publicly accessible and cannot be accessed.
- *E2* - Publication is marked as retracted.
- *E3* - Publication is unfinished.
- *E4* - Publication is a master thesis or literature survey. Related surveys are considered separately in Section 2.
- *E5* - Publication is written in a language other than English.
- *E6* - Publication is a duplicate of an already-considered publication.

3.3 Literature Analysis and Results

After applying the selection criteria presented in subsection 3.2 that can be applied automatically in Google Scholar i.e. publication year, the number of publications was reduced to 238. After reviewing the abstracts of the remaining papers and applying the remaining selection criteria by hand, we are left with 48 papers that are accepted for this survey. This includes the publications added during the snowball search. Serverless functions are the main concern of 21 of those publications, whereas the main area of the remaining 27 publications is non-serverless function offloading.

All selected publications and their content, including all figures are systematically analyzed. This is the most challenging part of a systematic literature review that aims to extract, gather, and classify information included in the literature. This allows us to build multiple taxonomies that provide a structured decomposition of related concepts.

4 Application Domains and Use Cases

This section explores application domains and use cases for function offloading and serverless functions in the computing continuum. It provides a comprehensive overview of how function offloading is applied across various sectors, with each subsection offering a brief description of notable use cases, supported by literature references. Figures 1 and 2 provide a graphical representation of the taxonomy of identified application domains and corresponding use cases found in the literature. We present below all use cases covered in the selected literature.

4.1 IoT World of Smart Devices

We identify the wide variety of IoT devices, including: (I) smart city devices (Section 4.1.1); (II) smart grid devices (Section 4.1.2); (III) industrial IoT devices (Section 4.1.3); (IV) smart healthcare devices (Section 4.1.4); and (V) sensors (Section 4.1.5) as one of the application domains for function offloading and serverless functions. IoT devices are usually resource-constrained, offering limited computing, networking, and storage capabilities [89] [64]. Some IoT devices are also battery-powered setting additional power restrictions for applications. Use cases resulting from applications running on IoT devices have various characteristics. However, based on the selected literature, we can identify the main objectives of applications that can be met using serverless and non-serverless function offloading. Specifically, low delays, data privacy, elastic scaling, energy efficiency, and access to more resources. In the next subsections, we present the use cases found in the literature in more detail.

4.1.1 Smart City

Smart city applications can vary from infrastructure monitoring, water monitoring and management, smart buildings, smart city services, smart transportation, or public health. The primary characteristics of smart city use cases are high availability and low maintenance costs since numerous IoT devices are spread among urban areas and often process critical data. Moreover, for use cases that influence public safety, providing secure computation offloading is essential. For this reason, Alli et al. [4] propose SecOFF-FCIoT, a secure offloading to fog and cloud for smart cities. In this solution, resource-constrained and battery-powered IoT devices can offload intensive tasks to fog nodes. The workload is categorized as either sensitive or non-sensitive and then transmitted to the cloud if the fog node is unable to handle it. The SecOFF-FCIoT evaluation showed that it minimizes latencies and

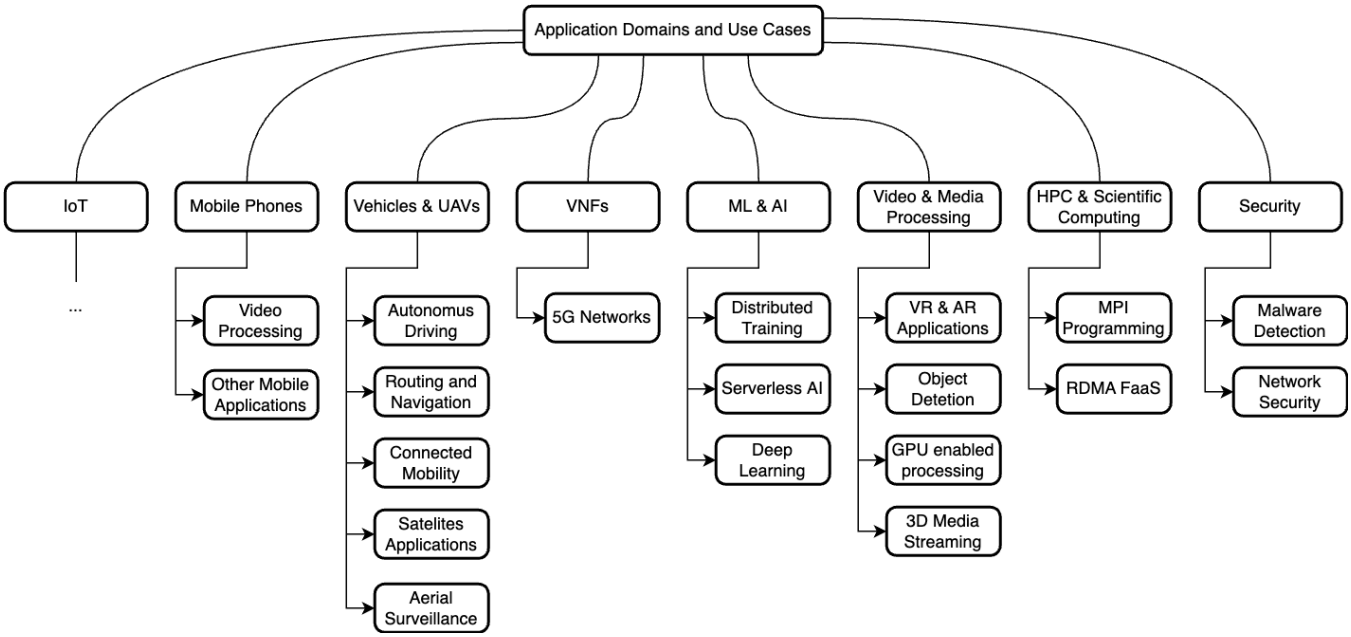

Figure 1: Taxonomy of Application Domains and Use Cases.

energy consumption. In [19], Cheng et al. propose something different. A data-centered programming model and orchestration mechanism called Fog Function. It aims to allow easier service logic modeling and provide higher flexibility than existing FaaS solutions for IoT. It does so thanks to a relaxed lifetime of functions, relaxed resource constraints on functions, and the support of mobility-aware task migration. In their paper, Cheng et al. showcase the usability of this solution by realizing the smart parking system based on the real-life scenario of Murcia City. The Fog Function system sources data from multiple parking site data providers and connected cars in the city. Based on this data, upon user request, it provides real-time parking recommendations. Since such calculations are data-centered and data-heavy, they are offloaded to edge or cloud nodes with more resources. Another use case is presented by Mete et al. [63]. They propose a serverless cloud GIS platform for real estate valuation. The novelty of this solution is that it is based on a serverless database management system. Such offloaded databases proved to respond to big queries faster and scale up better during high workloads.

4.1.2 Smart Grid

Smart grid applications vary from smart city use cases by covering wider areas, especially IoT devices in energy grids can be spread out across the whole country. Thus high elasticity and auto-scaling play a crucial role for smart grid applications. Tornado, or Toci, is a computational intelligence system for energy management presented by Huber et al. [39]. It can comprise multiple buildings, energy farms, and hundreds of sensors. Toci's main objective is to learn about typical energy use in the environment and alert managers in case of abnormalities in the operation of the energy grid. The system is designed to scale well due to the serverless architecture while preserving a low operational cost. Computations are run both at the edge and in the cloud, so more complex machine-learning tasks are offloaded to the cloud. Zhang et al. [94] also rely on serverless technology. They provide a cloud-centered architecture to ensure operation continuity regardless of local infrastructure availability for power grid emergency generation dispatch. The major advantages of this work are fault-tolerant infrastructure for grid monitoring and control abilities, even if on-premises systems lose functionality.

4.1.3 IIoT

Intelligent Industrial Internet of Things and its applications, together with the number of connected devices, including sensors, are growing extensively. This leads to an increase in the demands for low latency, resource-sensitive, and computation-intensive workloads that consume a significant amount of data. Moreover, event-driven execution and businesses' pursuit of cost reduction, e.g. reducing idle times, further distinguish IIoT use cases from smart city or smart grid use cases. Hazra et al. in [36] propose a computation offloading mechanism that utilizes the collaboration of fog and cloud nodes for IIoT applications. Their framework, called TSCO, aims to optimize energy usage and latencies by transferring offloaded tasks based on their importance. TSCO is providing good results thanks to the use of a federated fog network that can be supported by conventional cloud nodes during high traffic. CSCO is another solution for IIoT, proposed by Bebortta et

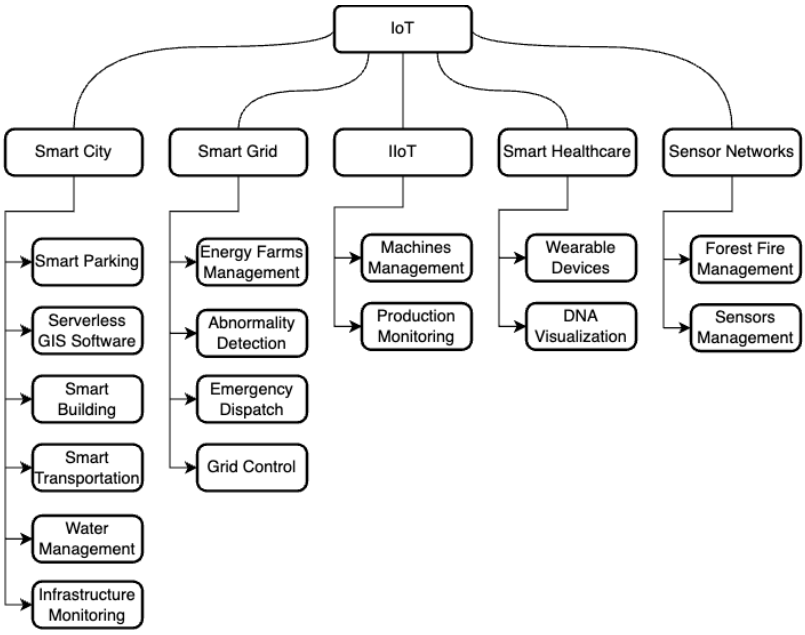

Figure 2: Taxonomy of Application Domains and Use Cases, continued.

al. [8]. It aims to provide QoS for function offloading to the edge. CSCO models industrial workloads as stochastic processes and observes their processing denial rates caused by overloaded servers. Using this data, the solution can provide adaptive performance modeling and can be used for computation offloading of time-critical tasks. On the other hand, Fan et al. [33] focus on task offloading for machine learning-based IIoT applications. The main objective of this work is to minimize the long-term average system cost, which is influenced by task offloading, resource allocation, and the accuracy of the ML model deployed on sensors, edge, and cloud nodes.

4.1.4 Smart Healthcare

Healthcare is one of the primary applications for IoT devices today, which includes various wearable devices. In contrast to IIoT or smart city devices, wearables are by design tiny, battery-powered devices, thus focusing on saving the device battery, and providing more resources for applications is a characteristic of smart healthcare use cases. Data generated by those applications needs analysis and further storage, which is not possible on resource-constrained smart devices. Moreover, these data have a private nature and need to be processed and stored securely. Meena et al. [62] address those concerns by introducing TEFLON, a trust-enforced computation offloading technique. TEFLON implements two algorithms that enable optimal offloading and trust assessments for entities in the fog environment. Another interesting use case in the domain of healthcare is DNAvisualization.org, presented by Lee et al. [51]. Their work allows the transformation of DNA sequences into easier-to-interpret visualizations. The implemented solution utilizes serverless functions to achieve cost efficiency and high performance.

4.1.5 Sensor Networks

Use cases of function offloading and serverless functions for networks of sensors have yet different characteristics. Such use cases, due to the potentially large amount of sensors require easy auto-scaling and support for parallel computation to accommodate data from all devices. An exciting use case for serverless functions is presented by Kang et al. [75]. They propose a dynamic offloading model for edge computing and showcase its usability on the network of sensors supporting forest fire management. Sensors collect data such as temperature, humidity, or wind direction in real-time, but due to limited resources, they are unable to store historical data and perform data analysis. Therefore, data is offloaded and computation is collaboratively done among edge nodes providing fire predictions, spread rate, flame height information, etc. Another example of using serverless functions in IoT is presented in [46], where a management system for a sensing system is presented. The presented solution helps in modeling the services and task offloading between IoT devices, fog, and cloud nodes. Yu et al., in their work [93], take a different look at IoT devices and the massive amount of data generated by smart sensors. They present an offloading scheme that aims to maximize the benefits of data analysis. They pinpoint the business value that can be collected out of the data processed in the IoT-Edge-Cloud ecosystem.

4.2 Mobile Phones

Another application domain we identify is mobile devices such as smartphones and tablets. Mobile phones still suffer from limited computational and storage resources like IoT devices (Section 4.1). Moreover, mobile devices are battery-powered by nature which further limits their capabilities to maintain heavy computational load for long periods. However, use cases for function offloading and serverless functions in the mobile phone application domain differ from those for IoT smart devices. The difference is dictated by the distinct nature of mobile applications compared to applications for IoT devices. Mobile applications are foremost user-centered and often focus on delivering high user-perceived QoS while preserving device battery life [77].

Some of the accepted papers focus on mobile applications. All of those publications agree that code offloading is a promising way to accelerate mobile applications and reduce the energy consumption of mobile devices by moving computations to more resourceful nodes. In [53] Lin et al. argue that existing code offloading systems suffer high delays between mobile devices and the cloud. To mitigate this issue, they propose Echo, an edge-centric decision engine that aims to efficiently offload functions among three-layer computing environments, consisting of mobile, edge, and cloud. LAMCO, the layered mobile offloading system proposed by Ballan et al. [7], aims to improve code migration from mobile Android devices. The main advantage of the LAMCO is the introduction of standardized offloading architecture and QoS components that ensure enhanced performance of the system. Other works look at the offloaded tasks in more detail. Mo et al. [64] investigate the problem of dependency-aware function offloading and service caching to propose a directed acyclic graph-based offloading mechanism that minimizes the overall service delay. Other authors focus on video processing, Appadurai et al. [5] define a weighted function of frame rate and precision to enhance video analytics. Ren et al. [71] reveal ACTOR, an adaptive offloading framework for 4K mobile AR. The ACTOR dynamically downscales the mobile video feed before sending it to a remote server to minimize user-expected latency, thus improving the user experience.

4.3 Vehicles & UAVs

Function offloading finds its use in another area of research that has received widespread attention in recent years, autonomous driving. Due to the limited battery life and computing capabilities of autonomous vehicles, many urgent and computation-intensive tasks need to be offloaded to maintain response time requirements and safety standards. A solution improving the Earliest Deadline First Algorithm for scheduling such tasks for autonomous driving is presented by Dai et al. [25]. Work in a similar area by Royuela et al. [73] proposes a testbed for evaluating cooperative, connected, and automated mobility applications. While both works highlight the benefits of edge computing and the deployment of 5G networks, Royuela et al. focus on analyzing images and making vehicle routing decisions.

For unmanned aerial vehicles (UAVs), those concerns of limited power and computing resources are valid as well. UAVs are a promising technology to support human activities such as surveillance or disaster rescue. However, many of those tasks require processing that is beyond the capabilities of the UAV computation platform. To tackle this issue, Chen et al. [16] propose iTAO, an intelligent task offloading algorithm. iTAO aims to lower the latencies and power consumption for task offloading in UAV networks using the Monte Carlo Tree Search algorithm. A different approach to UAVs is presented by Kang et al. [45]. Their work explores the use of UAVs and high-altitude platforms to collect and process functions offloaded from ground devices. Another work by Xiong et al. [90] also utilizes UAVs to support computation offloading from even more resource-constrained IoT devices on the ground.

Fan et al. in [32] focus on yet another class of devices, which are satellites. Intelligent applications on satellites, i.e., ones utilizing deep neural networks, are urgently in need of help in extracting useful information from massive surveillance data on time. Therefore, due to the limited computing abilities of satellites, Fan et al. propose to offload part of the computation to the ground station with the intermediary step of airships, helping to reduce energy usage and minimize delays.

We identify this group of devices, namely ground and aerial vehicles because they and applications for those devices further differ from IoT and mobile devices. Vehicles and UAVs often require time-sensitive computations to maintain the safety of operations, and not less often those computations involve resource-demanding image processing. In the case of vehicles and UAVs, the use of function offloading and serverless functions enables their autonomous acting, and applications utilizing offloading are not specifically user-centered as mobile applications are (Section 4.2).

4.4 Virtual Network Functions

Offloading of Virtual Network Functions (VNFs) is yet another application domain we identify. VNFs in contrast to IoT, mobile phones, and vehicle application domains do not focus on supporting tasks and applications on resource-constrained devices but on providing network services that can be entirely virtualized in software and can either support networking hardware or be an alternative to networking hardware [18]. Characteristics of serverless functions such as event-driven execution and elastic auto-scaling are crucial for the VNF application domain.

VFN offloading leverages generic servers to implement various network functions as software components instead of

purpose-specific hardware. This approach introduces a new dimension of cost savings and network function deployment flexibility and scalability by leveraging the computational power of the cloud or the proximity of edge computing. Ma et al. [56] propose a high-performance VNF and task offloading provisioning and assignment algorithm. They aim to balance the non-trivial tradeoff between the usage of computing and networking resources on edge servers. The proposed solution maximizes the number of request admissions by choosing whether to allow more VNF instances or task offloading requests in the available infrastructure. Another example is shown by Paolucci et al. [66], who offload user plane functions to P4 programmable switches in the existing edge network. They argue that VNF offloading to programmable network devices is of extreme interest for the deployment of 5G and beyond networks, where the offered network capacity will need to sustain low latencies under high traffic loads.

4.5 ML & AI

We identify a wide variety of machine learning and artificial intelligence applications as another application domain strongly present in the selected literature. ML & AI workloads leverage the high scalability and parallelism properties of serverless functions. Resource-constrained devices also benefit from powerful resources that are available in the cloud or edge and thus can run more demanding applications utilizing e.g. deep reinforcement learning.

SIREN is an asynchronous distributed machine learning framework presented by Wang et al. [87] in response to the need for scaling up machine learning. In the presence of an increase in volume and variety of data, new distributed machine learning systems based on parameter servers are developed. SIREN aims to provide a higher level of parallelism and elasticity while reducing system configuration overhead at the same time, thanks to the use of serverless functions. A similar interest in optimizing serverless AI workloads is presented in the work of Christidis et al. [20]. Their contribution focuses on providing optimization techniques for transforming generic AI codebases into serverless functions. They evaluate the proposed solution based on the intelligent transportation case study of on-demand, real-time predictions of flows of train movements across the UK rail network.

Other examples found in the literature include using deep reinforcement learning across serverless functions to reduce costs of malware detection [10], offloading machine learning tasks from sensors and IoT devices in an industrial setting [33], offloading ML tasks from satellites [32] or using deep neural networks deployed in serverless clouds to build on-demand video surveillance systems [31]. Yet another usage is presented by Yao et al. in [91]. They aim to optimize the performance of function offloading in serverless edge computing using deep reinforcement learning. This work focuses on modern ML-based intelligence applications such as face recognition, which need to be offloaded from resource-constrained IoT devices.

4.6 Video and Media Processing

We identify video and other media processing as another application domain for function offloading and serverless functions. Video processing is gaining traction due to its significance for VR/AR applications which offer users unique experiences in various domains such as entertainment, education, training, and healthcare [88]. Use cases of video and other media processing leverage the additional resources that are available to end devices by utilizing the function offloading and serverless functions.

Video processing is an enabling technology for many modern applications, such as virtual and augmented reality, interactive applications that analyze facial expressions and gestures, self-driving, accurate tracking, and surveillance or traffic monitoring, among others. However, real-time video processing is a computationally expensive process that often has to be offloaded from resource-constrained devices, as presented by Chemodanov et al. [15]. The authors of this publication propose a policy-based function offloading scheme that aims to enable real-time analysis with object and motion detection for drone video feeds. The solution is based on decomposing an existing computer vision pipeline for object and motion detection into micro-service functions that are executed on geo-distributed servers. Similarly, Salehe et al. [74] decompose video processing into modules that can be executed on any device. Their solution, called VideoPipe, allows collaborative video processing done on connected edge devices, e.g., smart home IoT devices connected through a Wi-Fi network. Salehe et al. evaluate VideoPipe by implementing a gesture-based IoT application. Another work by Risco et al. [72] delves into enabling the usage of GPU-based resources for function offloading in multimedia processing. The authors presented a case study of an automated workflow that generates subtitles based on the audio input and applies object recognition to the video frames. Those compute-intensive functions were offloaded to the server and efficiently executed using GPU resources. Appadurai et al. [5] introduce resource-aware video analytics offloading for mobile devices such as smartphones. They focus on optimizing the resource allocation when offloading the functions. Immersive 3D media streaming is discussed, and a serverless framework for streaming such media is presented by Konstantoudakis et al. [50]. The authors also showcase 5G networks as an enabling technology for 3D media streaming. In [68], Peng et al. propose a privacy-aware computation offloading method for virtual reality that enables secure offloading from mobile VR devices. Ren et al. [71] also focus on mobile VR use cases, but their work aims to provide 4K resolution with low latencies thanks to dynamic down-scaling of the mobile video feed before sending it to a remote server. Ren et al. claim that

their solution, Actor, outperforms competitive methods and provides high user-perceived video quality.

4.7 HPC & Scientific Computing

Yet another application domain, with different characteristics and requirements, that we identify is the field of high-performance and scientific computing. HPC computing usually processes large amounts of data and relies on compute-heavy algorithms and high parallelism of computations [12]. Thus, to reduce costs and simplify the HPC and scientific computations, function offloading and serverless functions can be leveraged, as they promise such characteristics as high parallelism, good resource utilization, idle times reduction, and pay-as-you-go model.

Message-passing Interface (MPI) is a very popular distributed programming paradigm that dominates the current landscape of high-performance computing (HPC) and is the leading use case for supercomputers. Despite the elasticity of the cloud, supercomputers still suffer from low resource utilization rates. To address this issue, Copik et al. [23] present a remote direct memory access (RDMA) capable function-as-a-service platform for HPC. rFaaS, as it is called, brings the benefits of serverless computing and offers low-latency remote function invocations in multi-tenant environments. Cepik et al. evaluate rFaaS and claim remote function execution with negligible performance overheads. Another solution for a flexible and efficient serverless supercomputing platform is presented by Chard et al. [14]. They present funcX, a platform dedicated to scientific high-performance computation tasks. They provide case studies of metadata extraction, machine learning interfaces, synchrotron serial crystallography, quantitative neurocartography, and X-ray photon correlation spectroscopy. Cordasco et al., in their work [24], also focus on scientific workloads in serverless infrastructure models. They present FLY, a domain-specific language that aims to streamline designing, deploying, and executing scientific applications, specifically exploiting FaaS characteristics. Similarly to previous authors, Cordasco et al. also notice the promise of serverless functions in HPC to allow extreme-scale applications through fine-grain decomposition of the application and more efficient scheduling on cloud infrastructure.

4.8 Security

We also identify the application domain of security use cases, since cybersecurity is a well-established domain of research, and security applications can be implemented among various devices, systems, and networks. Security use cases can leverage e.g. the cost-efficiency, event-driven execution, or ease of deployment that are promised by serverless functions.

Birman et al. [10] investigate cost-effective malware detection as a service over serverless functions. This proposition follows the trend of transitioning parts of operations off-premise to reduce costs and improve scalability. Such migration of security functions to the cloud also proves to provide easier management and excellent disaster recovery capabilities. Parres-Peredo et al. [67], on the other hand, look into evaluating and identifying unexpected behavior of network users. The proposed solution leverages the potential of serverless computing to integrate with other services, such as distributed caches or data stores, and execute functions in an event-driven manner. The implementation uses a TopK ranking-based method to classify captured network traffic, and all analytic functions are offloaded from on-premise hardware.

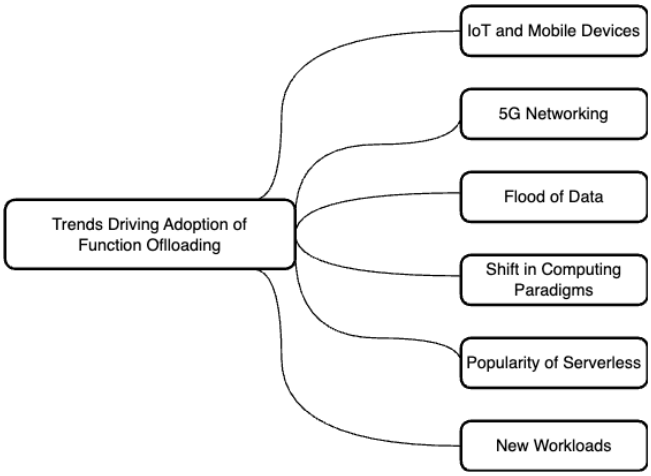

Figure 3: Taxonomy of Trends Driving the Adoption of Function Offloading.

5 Trends Driving Adoption of Function Offloading

In this section, based on the use cases and application domains presented in Section 4, we identify and present the main trends in computing and new enabling technologies that drive the adoption of function offloading in the computing continuum. We find that the reasons behind the adoption of function offloading are correlated; however, we managed to identify six main driving trends. These are (I) an increase in the number of connected IoT and mobile devices (Section 5.1); (II) the introduction and deployment of 5G networks (Section 5.2); (III) a still increasing amount of generated data needing processing (Section 5.3); (IV) a shift towards edge and fog computing (Section 5.4); (V) the rising popularity of serverless models (Section 5.5); and (VI) new, emerging computation-heavy types of workloads (Section 5.6). We depict the taxonomy of identified trends in Figure 3, and below we present each trend or technology in more detail.

5.1 IoT and Mobile Devices

The rapid and steady increase in the number of IoT and mobile devices connected to the Internet is one of the main identified factors driving and influencing the adoption of function offloading in the modern computing continuum [48]. Many of those IoT and mobile devices (Section 4.1 and Section 4.2), i.e., smartwatches, smart glasses, and environment sensors, are resource-constrained, offering limited computing and storage capabilities [89] [64]. Another group of low-cost devices, including those of industrial grade or do-it-yourself (DIY) devices, run on low-end hardware that, in many cases, cannot do more than just handle the network connection and offload the computation to external computing nodes [16] [74]. Some other devices, like VR/AR headsets, drones (Section 4.3), or wearable health sensors, are battery-powered and do not have enough energy to compute heavy tasks on their hardware [56] [4]. The solution to the above issues is the introduction of function offloading, which aims to move compute- and data-heavy tasks to other network devices. Offloading enables more applications to run on IoT and mobile hardware (Section 4.5 and Section 4.6), i.e., virtual reality [15], machine learning [33], or blockchain [52]. Thanks to offloading, IoT and mobile devices find their use cases in more areas of our lives, including healthcare [62] and industrial manufacturing [36], to name a few.

The IoT market is believed to continue its rapid growth as it is dictated by the new needs that arise from further increasing global population and urbanization levels [41]. It is estimated that there will be about 75.4 billion interconnected devices by 2025 [79]. According to a Cisco report from 2020, [21], smart home applications should have the largest share of connected devices by the end of 2023. Still, connected cars should be the fastest-growing application type in the IoT world, with 30% growth yearly. Industrial interconnected machines are not far behind, almost matching the growth of connected cars and reaching 14.7 billion connections by 2023. The number of smartphones is still believed to grow by 7% yearly. The same report states that over 70% of the population will have mobile Internet connectivity by the end of 2023 and that the total number of Internet users is continuing to grow. Having the above numbers in mind, we can assume that IoT and mobile devices will continue to push the adoption of function offloading forward in the upcoming years.

5.2 5G Networking

We find that the emergence of 5G networking is another main driving factor behind the adoption of function offloading. The emerging 5G networks bring a lot of promises, such as high speeds, ultra-low latencies, increased network connectivity and capacity, greater bandwidth, and improved energy efficiency [47]. With such targeted key performance indicators, 5G is becoming an attractive option for enabling new devices and applications in the computing continuum [38]. As established in the previous Section 5.1, the increasing rate of overall global connectivity and the advent of IoT and mobile devices require enormous networking capabilities to accommodate the number of connections and bandwidth demands. In that regard, 5G networking also seems to secure further development of IoT, mobile devices, and new applications.

Thanks to high-performance networking, more remote devices can quickly access network resources such as powerful computing nodes, storage, caches, or data providers [17]. That characteristic makes function offloading not only possible but also compelling and beneficial from the quality of service and user experience points of view [50]. For instance, drones and other UAVs (Section 4.3) can offload their trajectory computation to edge network nodes with minimal delays [16]. Available greater bandwidth enables mobile battery-powered devices to offload ML-based computations in real-time for virtual reality applications [50] (Section 4.5). 5G enables and drives other use cases as well, such as video analytics for resource-constrained devices [5] (Section 4.6), connected self-driving cars [25] [73], and efficient streaming of immersive 3D and 4K media [50]. Another application domain comes from 5G technology itself. Modern networks such as 5G need to be flexible, scalable, and efficient to sustain high data rates and extremely low latencies under heavy loads [47]. This can be achieved by virtualizing network functions (VNFs) and services traditionally run on proprietary hardware (Section 4.4). Offloading strategies for VNFs are currently of extreme interest for the deployment of 5G networks, as they reduce the need for physical infrastructure and allow more agile network service development [66]. Since 5G networking is still in its early years of adoption, with 5G devices accounting for only over 10% of global mobile device connections [21], we predict the further emergence of new types of applications and use cases driven by 5G deployment.

5.3 Flood of Data

The ongoing increase in the already massive amount of data generated every day influences the adoption of computation offloading. The collected data needs processing; however, the required processing power and storage are often beyond the capabilities of the collecting device. As we call it, a flood of data that takes place in computing is strongly correlated with the advent of IoT and mobile devices discussed in Section 5.1 and new networking possibilities delivered by 5G networks discussed in Section 5.2, as the 5G networks can transfer more data from a greater number of devices than ever before. A forecast by the International Data Corporation indicates that only IoT devices will generate 79.4 zettabytes (ZBs) of data in 2025 [29]. By the same date, the total amount of data created and consumed globally is projected to grow to more than 180 ZBs [80].

As more and more devices and applications produce and

	Computing Power	Latency	Scalability	Mobility	Architecture
Cloud Computing	High	High	High	Low	Centralized
Fog Computing	High	Medium	Medium	Medium	Distributed
Edge Computing	Medium	Low	Low	High	Distributed

Table 2: Cloud, Edge, and Fog Computing characteristics comparison.

require real-time processing of vast volumes of data, traditional computing models struggle with bandwidth, latency, and processing power limitations [9]. Computation offloading addresses these challenges by moving data processing tasks from resource-constrained devices to more powerful cloud or fog computing resources [93]. This shift enables more efficient data handling, quicker processing times, and improved battery life for devices [19]. It supports the development of advanced technologies and applications that demand high computational power and real-time data analysis. Examples include networks of sensors in industrial [36] or urban [4] environments (Section 4.1), machine learning tasks involving training models on big data sets [87] (Section 4.5), real-time video analysis [15] (Section 4.6), and distributed cache services [85], among others. With the amount of generated data believed to be still growing in the upcoming years, we think it will continue to push the adoption and applications of function offloading forward.

5.4 Shift in Computing Paradigms

Due to the tremendous amount of data generated by the continuously growing number of connected devices as described in Section 5.3 and the new networking possibilities presented in Section 5.2, the landscape of the computing continuum has changed. This change makes function offloading more suitable and efficient for modern applications and their requirements. Currently, next to cloud computing, there are a few emerging paradigms in the computing continuum that are of significant importance to the adoption of function offloading, namely edge computing, fog computing, mobile cloud computing, and mobile edge computing. Table 2 presents an overview and comparison of cloud, edge, and fog computing characteristics.

Edge computing refers to a distributed computing paradigm that brings computation and data storage closer to where they are needed to improve response times and save bandwidth. The main idea is to process data near the network's edge, where the data is being generated, instead of in a centralized data-processing cloud. This approach reduces latency and can also enhance privacy and security by keeping sensitive data within the local device or local network. Edge computing is widely used in IoT applications [89] [91] [46] (Section 4.1), autonomous vehicles [25] [73] (Section 4.3), multimedia processing [74] (Section 4.6), and other scenarios where quick, local decision-making is critical [53] [64].

Fog computing extends the concept of edge computing by creating a distributed network that includes the edge devices and the intermediate nodes between these devices and the cloud. This "fog" layer can process, analyze, and store data, offering additional flexibility and resources compared to edge computing alone. Fog computing supports a broader range of applications by enabling more processing resources over often larger geographic areas. Example use cases include healthcare applications [62], data-intensive computation [19] (Section 4.7), industrial IoT [36], smart cities [4], and sensing network applications [75].

Mobile cloud computing specifically refers to the delivery of cloud computing services to mobile devices. While cloud computing serves a broad range of devices and needs, mobile cloud computing (MCC) is specifically tailored to improve the mobile user experience by offloading processing and storage to cloud resources, thereby overcoming the limitations of mobile devices, such as limited battery life, storage, and processing power.

Mobile edge computing, or multi-access edge computing (MEC), is a subset of edge computing that specifically targets mobile networks. It involves deploying computing resources at the edge of mobile networks, such as in cell towers or base stations, to serve mobile devices directly. It has been extensively researched, and applications in many fields have emerged, such as mobile blockchain [52], video processing for VR/AR [68] [5], virtualization of network functions [56] [66] (Section 4.4), or general function offloading for IoT use cases [17] [8].

This shift in the computing continuum helps in distributing the computation among the network, lowering the delays, and minimizing core network congestion. Those performance characteristics are desired when considering function offloading; thus, we consider this shift in computing as a driving factor in the current adoption of function offloading. While cloud computing and MCC provided enormous resources for mobile devices, offloading from such devices often suffered from high latency, which made it unusable for the real-time video processing required by autonomous driving, drones, or mobile VR/AR platforms [4] [19] [52]. The introduction of edge computing, fog computing, and MEC, coupled with new high-capacity networking, mitigated those concerns and bootstrapped the adoption of function offloading in the aforementioned use cases [25] [16] [90].

5.5 Popularity of Serverless

The rising popularity of serverless computing and the function-as-a-service (FaaS) model, in particular, were identified as another main factor driving the adoption of function offloading. Serverless, despite being a relatively new technology, was already adopted by more than 70% of organizations

using the Amazon AWS platform in 2022 [27], and the high adoption rate is expected to continue. FaaS platforms are built on top of serverless ideology and are the result of the evolution of service-oriented architecture [83]. Firstly, services were decomposed into microservices, later into smaller, single-responsibility code units - functions, whose execution is event-driven. Serverless functions are one of the ways to execute offloaded tasks from client devices or other nodes in the network. For that reason, the growing popularity of serverless and FaaS is driving and shaping the direction of function offloading adoption in the current landscape of the computing continuum.

Serverless characteristics and benefits are the source of this trend of the industry going serverless. That includes cost efficiency as the result of the pay-as-you-go model, where customers pay only for actually used resources [76]. Another advantage is high scalability and elasticity, which result from the on-demand provisioning of independent functions that can run in parallel [84]. High availability and reliability are also considered as other benefits of FaaS platforms. Since the execution of the offloaded workload is not dependent on any application server state, the overall availability of the service is usually higher and depends only on the availability of the provider's serverless platform, which is often geographically distributed [58]. Employing a fine-grained payment scheme and functionality-focused approach makes the use of the FaaS platform more business-oriented, which is an advantage for many business owners as well [3]. Last but not least, thanks to the dynamic allocation of resources and de-provisioning of idle functions, providers of serverless platforms can reduce their operational costs due to better resource utilization. Presented characteristics drive the adoption of serverless functions in many modern use cases, such as networks of sensors [75] (Section 4.1), scientific high-performance computing [24] [14] (Section 4.7), multipurpose function-granularity offloading to cloud [37] [17] [91], data-intensive applications for IoT [19] and cybersecurity [10] (Section 4.8) among others. At the same time, they contribute to the adoption of function offloading in the computing continuum.

5.6 New workloads

We identify new, emerging workloads as another main factor driving the adoption of function offloading in the computing continuum. Figure 4 depicts their overview. These new workloads and applications often demand more resources that are available on the device or require specific resources that cannot be accommodated in the device form factor. In order to enable such applications, e.g., on mobile or IoT devices, developers need to offload some or all computations to external computing nodes [5]. This applies to real-time video processing for mobile augmented reality applications, where a battery-powered device such as a smartphone or VR/AR goggles does not offer enough computing power to process

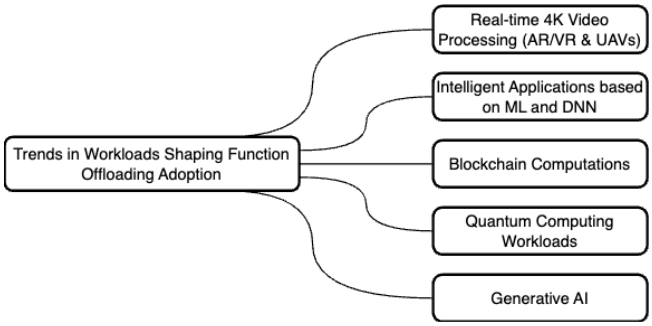

Figure 4: Taxonomy of New Emerging Workloads Influencing the Adoption of Function Offloading.

the 4K feed in real-time [71] [74] (Section 4.6). Drones and other UAVs (Section 4.3), to maintain the required quality of the user experience, also lean towards offloading the video processing [15]. Another application allows clients without powerful GPU resources to offload multimedia processing to GPU-enabled serverless functions [72]. Some more examples include autonomous driving [25], mobile blockchain applications [52], and serverless frameworks for quantum computing [65]. In the case of quantum computing, offloading quantum computations as FaaS enables more clients and applications to benefit from this novel technology on mobile platforms, which would not be currently possible otherwise.

There is also a rapidly growing group of intelligent applications that make use of machine learning and deep neural networks. That includes generative AI, whose market is believed to maintain very rapid growth in the upcoming years [70] (Section 4.5). Such ML/AI applications require significant computing resources to perform the deep learning process and massive storage resources to store the network models with all parameters. To tackle this issue and allow such applications to run on mobile platforms, a federated learning approach was proposed [40]. Federated learning aims to train the central model across decentralized devices or servers, making the training process collaborative. Some works focus on offloading such federated training computations among edge devices [43]. Other works investigate optimizations of offloading deep neural network (DNN) applications deployed in the computing continuum [31] [32].

6 Objectives Driving Adoption of Function Offloading

In this section, we present the main characteristics of function offloading found in the literature that drive its adoption in the computing continuum. We also group them into more general objectives. Those objectives are identified based on application domains and use cases discussed in Section 4, as well as trends identified and discussed in Section 5. This provides a good understanding of the function offloading domain,

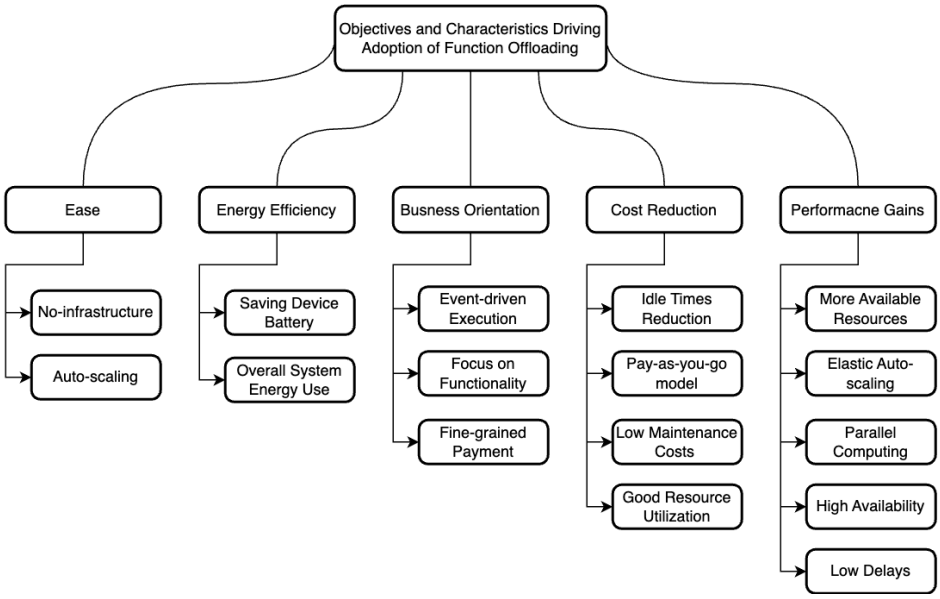

Figure 5: Taxonomy of Objectives and Characteristics Driving Adoption of Function Offloading.

including good insight into what objectives shape the current adoption of function offloading and serverless functions and what the importance of those objectives is for a particular use case. Figure 5 presents the graphical representation of the taxonomy of objectives and characteristics driving the adoption of function offloading.

6.1 Ease of Use

Ease of use is one of the objectives driving the adoption of function offloading that we identified based on the selected literature. We find that it is mostly related to the use of serverless architecture and FaaS platforms as it is showcased in numerous publications. For that reason, the characteristics we present here result from serverless model usage and are the main driving factors behind the rising popularity of serverless functions presented in Section 5.5. Many application domains and use cases discussed in Section 4 point out those characteristics and the general ease of function offloading as an important factor behind their adoption. The relationship between this objective and the use of serverless functions is bidirectional since the pursuit of easy application development, deployment, and management boosts further adoption of serverless solutions, resulting in ease. Below, we present the main characteristics contributing to this objective that we found in the literature.

No-infrastructure refers to the abstraction of servers and infrastructure management away from the developer, which is a fundamental aspect of the serverless computing model. This doesn't mean that there are no servers involved; rather, the responsibility for managing these servers and the underlying infrastructure lies with the cloud service provider [35]. Developers traditionally had to manage hardware, runtime environments, network configurations, handle load balancing, and security updates, which are time-consuming and complex tasks. Serverless functions eliminate this overhead, making the development and deployment of applications easier. Examples of application domains include meteorology [37], quantum computing [65], machine learning and AI [87] [20] (Section 4.5), web applications [51] [63], HPC [14] (Section 4.7) and smart grid applications [94] (Section 4.1).

Auto-scaling allows applications to dynamically adapt to varying workloads without any manual intervention. This functionality ensures that applications can efficiently handle any number of requests at any time by scaling compute resources up or down based on incoming request volume [34]. Developers and operations teams are freed from the complexities of planning for and managing scalability and can focus on developing features and improving application quality. In the case of serverless architecture, this also includes on-demand resource provisioning, which allows for the automatic allocation and provision of computational resources as soon as they are required without any human intervention. Publications that showcase auto-scaling benefits include works on scalable malware detection [10] (Section 4.8), implementing customized FaaS services [19] [65] or on-demand video surveillance [31] (Section 4.6). Auto-scaling characteristic is important in the era of data flood presented in Section 5.3.

We believe that the ease of use of serverless technologies will remain a key driving objective for the adoption of serverless functions as the rise in popularity of serverless and FaaS platforms does not seem to slow down (Section 5.5).

6.2 Energy Efficiency

We identify energy efficiency as another key objective driving the adoption of function offloading. It is particularly significant in contexts involving mobile devices, Internet of Things (IoT) devices, and other edge devices. In the case of this objective, there is no clear distinction between serverless and non-serverless models. Instead, there are two main separate aims: one focuses on saving the end device energy to prolong its battery life, and another tries to lower overall system energy usage, including end devices and computing nodes that execute offloaded functions.

Saving the energy of battery-powered mobile and IoT devices has the clear aim of prolonging the time that the device can remain operational and keep providing services to a customer. Such a scenario is widely discussed in the literature, and numerous application domains and use cases are provided. To name a few: smart city applications [4], drones and other UAVs [16] [90] (Section 4.3), mobile gaming [53], edge computing for multipurpose IoT devices [89] [64], VR/AR applications [68] [53], video processing and analysis for mobile applications [5] [15] (Section 4.6) and deep natural network-driven applications for mobile devices [32] (Section 4.2). All of the above examples use function offloading to move computation-heavy tasks from battery-powered devices to other computing nodes, reducing the device's workload and energy usage at the same time. The objective of energy efficiency plays a crucial role in driving the adoption of function offloading and serverless functions in the light of still increasing number of connected devices (Section 5.1) and new resource-demanding workloads such as machine learning or AR/VR video processing (Section 5.6).

The focus on overall energy usage and energy efficiency optimization in systems utilizing function offloading is also present in the literature [64] [36]. Found publications explore how to enhance the energy efficiency of the function offloading to the fog and edge layers while maintaining high performance and low delays. We believe that that kind of focus on energy efficiency will become more prominent in the computing continuum due to the rising concerns and efforts to reduce energy usage and, in consequence, the emission of greenhouse gases [57].

6.3 Business Logic Orientation

Another objective that we identify as the main driver behind the adoption of function offloading in the computing continuum is its business-oriented design. This objective applies mostly only to the serverless functions, as their invocation is event-driven and does not require any application server to be running all the time. Such event-driven execution of functions, typically in response to, e.g., HTTP requests, file uploads, and database changes, simplifies application architecture by decoupling components and services. This means that businesses can focus on innovation and core product development without worrying about hardware, infrastructure management, or network configuration. Serverless platforms, including FaaS, are the next step, after microservices, in the evolution of service-oriented architecture [83]. The trend of serverless popularity was already discussed in Section 5.5. Serverless evolution aims to make software execution resemble the business process as much as possible. Event-driven single-responsibility functions are a functionality-focused approach to computing and seem to fit in perfectly in that scenario. Moreover, this model of deployment introduces a fine-grained payment scheme, simplifies the deployment process, accelerates time to market, enhances scalability, as discussed in the previous Section 6.1, and boosts agility [22]. All of those characteristics are business-oriented. The application domains in the literature acknowledge this objective, to name some: serverless security services [10] [67] (Section 4.8), energy grid systems [94] [39] (Section 4.1), distributed cache [85], healthcare applications [51], and multimedia processing [72] (Section 4.6), among others.

We predict that software infrastructure and processes will continue to transform towards being more business logic-oriented [61]. As software continues to advance, there is an increasing emphasis on integrating business logic rules at the application level to enhance efficiency and accuracy in decision-making processes

6.4 Cost Reduction

We identify the cost reduction objective as another main factor driving the adoption of function offloading in the computing continuum. Cost reduction is a key objective of both non-serverless function offloading and serverless functions. Function offloading in a modern computing continuum, consisting of fog and edge layers as discussed in Section 5.4, introduces new opportunities for lowering deployment costs for providers. Serverless functions further this by eliminating the need for upfront, always-on server provisioning, which at the same time enables more attractive pricing for customers. Serverless model significantly lowers the financial barrier to deploying applications, making technology more accessible and budget-friendly for many businesses. For cost reduction objective, we distinguish two separate perspectives that are present in the literature, namely the provider's perspective and the customer's perspective.

Provider's perspective is focused on one characteristic: good resource utilization, which can be achieved in the multi-layer computing continuum thanks to adaptive resource allocation, dynamic utilization optimization, and reducing idle times. In non-serverless function offloading, providers can introduce dynamic and adaptive task allocation, scheduling, and migration among layers and nodes in the computing continuum. Such techniques aim to lower overall system utilization and optimize bandwidth usage. Lower system utilization means

decreased energy consumption and energy costs, while lower bandwidth usage means lower networking costs. In the case of serverless functions, next to the above-presented means, providers can also reduce idle times. Since the serverless functions are ephemeral and event-driven, the resources do not have to be provisioned upfront and kept idle for a customer [84]. This naturally allows providers to lower the overall system utilization and thus further lower energy costs. Publications that include this perspective on cost-efficiency notice the benefits for many application domains, including quantum cloud providers [65] and networking service providers [66] (Section 4.4), among others [4] [39] [93] [50].

Customer's perspective presented in the literature focuses on two main characteristics: the pay-as-you-go model and reduced maintenance costs. Both characteristics are also driving factors for the rising popularity of serverless, as discussed in Section 5.5. The **pay-as-you-go** model, which is integral to serverless functions, serves as a key strategy for cost reduction for customers. This model allows businesses to pay only for the computing resources they actually use, rather than paying for always-on resources provisioned by cloud providers [76]. Mitigating this overhead in payment for idle resources offers a direct pathway to operational efficiency and financial savings by aligning costs directly with usage. This characteristic is widely present in accepted publications; some of the use cases benefiting from it include cybersecurity services [67] [10], energy grid applications [39] [94] (Section 4.1), multimedia processing [72] (Section 4.6), healthcare applications [51], among others [85]. The characteristic of **reduced maintenance costs** refers to reducing costs associated with running applications on in-house or cloud servers. In a serverless setting, businesses are no longer required to keep the server's systems and runtimes up-to-date, so software maintenance costs are minimal. An approach like that also eliminates the cost of investment and further hardware maintenance. Examples of publications mentioning this characteristic are numerous and include serverless scientific and high-performance computing [23] [24] [14] (Section 4.7), serverless quantum computing [65], meteorology functions [37], on-demand video surveillance [31], and others [20] [63].

Cost reduction objectives will continue to drive and shape the adoption of function offloading and serverless functions as service providers and application maintainers will continue to try to keep the operational costs low despite the increasing number of interconnected devices (Section 5.1), the rising volume of data needing processing (Section 5.3) and new emerging resource-hungry workloads such as ML or VR/AR video processing (Section 5.6).

6.5 Improved Performance

Last but not least, we identify improved performance as one of the main drivers behind function offloading adoption. Function offloading enables devices to transfer heavy computational tasks to more powerful cloud or edge resources, enhancing application responsiveness and efficiency. Serverless functions contribute by allowing automatic scaling and eliminating the need for manual infrastructure management, ensuring applications can handle varying loads swiftly. This combination improves overall application performance, enabling a highly reliable user experience. Based on the examples in the literature, we identify and present below the five main characteristics contributing to this objective.

More resources than on local devices is an enabling characteristic for many modern applications and small form factor devices such as mobile (Section 4.2) or IoT devices (Section 4.1). Such devices have limited computing or storage capabilities; thus, they must rely on external computing resources to support many modern workloads, such as machine learning (Section 4.5) or real-time video processing (Section 4.6). Moreover, function offloading can give them access to more specific resources, such as quantum computers, that cannot be accommodated on their internal hardware. Examples showcasing the importance of this characteristic in the literature span across all application domains presented in Section 4. Some of the use cases are mobile applications [7] [64] including blockchain [52], function offloading from UAVs [45] [16] [32] [15] (Section 4.3), video processing [5] [15] [74] including VR/AR devices [68], healthcare IoT devices [62], and others [91] [53] [25].

Elastic auto-scaling was already presented in Section 6.1 in the context of ease that serverless functions bring to the developers. However, the disjointed group of publications considers the scaling characteristics of serverless functions as a performance gain. The scalability properties of serverless allow for handling spikes in traffic effortlessly, maintaining performance levels without the need for pre-provisioning resources. It translates to consistent, reliable user experiences even under variable workloads, enhancing overall application performance and responsiveness. Publications from this group present such use cases as serverless supercomputing [14], distributed machine learning and AI workloads [87] [20], energy grid intelligence [94] [39], and others [85] [72].

Parallel computing is crucial in several application domains where large-scale computations and data processing are required. These domains include scientific computing, big data analysis, machine learning and AI, multimedia processing, and high-performance computing [11]. In those areas, use cases like simulations based on complex models, training complex ML models, or processing large multimedia files can leverage and benefit from splitting a large task into smaller chunks that can be processed simultaneously across multiple serverless function instances. This approach allows for high levels of concurrency, and the practicality of serverless in such scenarios is noticed in the literature [24] [23] [51] [87].

High availability is another characteristic boosted by the use of serverless functions. It ensures that applications are resilient and continue to operate without significant downtime.

This is achieved through automatic replication and distribution of functions across multiple servers and data centers [58]. Serverless platforms manage these aspects, providing built-in fault tolerance and offering businesses the desired confidence that their applications are always accessible to users. Found publications showcase high availability benefits, for example, in energy grid monitoring [94] or high-performance computing for science [14] (Section 4.7).

Low delays is another key characteristic that drives the adoption of function offloading, especially when offloading takes place to the edge or fog layer, as discussed in Section 5.4. Thanks to modern low-latency networking technology, such as 5G, discussed in Section 5.2, function offloading can enhance application responsiveness and lower the computation delay seen by the user. This characteristic is mentioned in the publications, considering mobile VR/AR applications [71], serverless streaming of 3D media [50], and IoT use cases such as sensing networks [75] and industrial devices [8].

7 Conclusions

This study has thoroughly explored the evolving landscape of function offloading in the computing continuum, primarily focusing on serverless functions. Through our examination, we have identified key advantages such as cost reduction, performance gains, enhanced scalability, and high availability that these technologies offer. Our findings help in further understanding how function offloading, influenced by many trends like IoT proliferation and 5G networking, addresses modern computational needs across various sectors, including IoT, big data analytics, and AI. The pay-as-you-go model, intrinsic to serverless computing, exemplifies the shift towards more economically sustainable computing, allowing organizations to align their expenditures with actual usage. Moreover, the inherent scalability and high availability of serverless functions ensure that applications remain responsive and reliable under varying loads, contributing to a seamless user experience.

Function offloading has emerged as a strategic approach to enable new workloads and applications on resource-constrained devices, significantly enhancing energy efficiency and operational agility. Serverless computing, on the other hand, has redefined application development and deployment, enabling businesses to achieve unprecedented scalability and flexibility without the burdens of traditional infrastructure management. As we look to the future, the integration of function offloading and serverless computing will continue to play a pivotal role in shaping the next generation of digital solutions. Their ability to meet the demands of increasingly complex and data-intensive applications will be crucial in driving forward technological innovation and digital transformation.

In conclusion, serverless computing stands at the forefront of modern function offloading strategies, offering pathways to more sustainable, efficient, and scalable digital infrastructures.

References

[1] Aws re:invent 2014 | (mbl202) new launch: Getting started with aws lambda, Nov 2014.

[2] Abu-Taleb, N. A., Hasan Abdulrazzak, F., Zahary, A. T., and Al-Mqdashi, A. M. Offloading decision making in mobile edge computing: A survey, 2022.

[3] AlibabaCloud. Deploying a faas platform for content delivery, 2021.

[4] Alli, A. A., and Alam, M. M. Secoff-fciot: Machine learning based secure offloading in fog-cloud of things for smart city applications, 2019.

[5] Appadurai, J. P., Sengodan, P., Venkateswaran, N., Roseline, S. A., and Rama, B. Radial basis function networks-based resource-aware offloading video analytics in mobile edge computing, Jun 2023.

[6] Aslanpour, M. S., Toosi, A. N., Cicconetti, C., Javadi, B., Sbarski, P., Taibi, D., Assuncao, M., Gill, S. S., Gaire, R., and Dustdar, S. Serverless edge computing: Vision and challenges, 2021.

[7] Ballan, Y., Ahmed, A., and Baghaei, N. Lamco: A layered approach to mobile application computation offloading, 2020.

[8] Bebortta, S., Senapati, D., Panigrahi, C. R., and Pati, B. Adaptive performance modeling framework for qos-aware offloading in mec-based iiot systems, 2022.

[9] Berisha, B., Mëziu, E., and Shabani, I. Big data analytics in cloud computing: an overview, Aug 2022.

[10] Birman, Y., Hindi, S., Katz, G., and Shabtai, A. Cost-effective malware detection as a service over serverless cloud using deep reinforcement learning, 2020.

[11] Blaise Barney, Livermore Computing, D. F. Introduction to parallel computing tutorial, 2021.

[12] Carvalho, F. H. d., Al-Alam, W. G., and Dantas, A. B. d. O. Contextual contracts for component-oriented resource abstraction in a cloud of high performance computing services, 2021.

[13] Cassel, G. A. S., Rodrigues, V. F., da Rosa Righi, R., Bez, M. R., Nepomuceno, A. C., and André da Costa, C. Serverless computing for internet of things: A systematic literature review, 2022.

[14] Chard, R., Skluzacek, T. J., Li, Z., Babuji, Y., Woodard, A., Blaiszik, B., Tuecke, S., Foster, I., and Chard, K. Serverless supercomputing: High performance function as a service for science, 2019.

[15] Chemodanov, D., Qu, C., Opeoluwa, O., Wang, S., and Calyam, P. Policy-based function-centric computation offloading for real-time drone video analytics, 2019.

[16] Chen, J., Chen, S., Luo, S., Wang, Q., Cao, B., and Li, X. An intelligent task offloading algorithm (itoa) for uav edge computing network, 2020.

[17] Chen, X., Li, M., Zhong, H., Chen, X., Ma, Y., and Hsu, C.-H. Funoff: Offloading applications at function granularity for mobile edge computing, 2024.

[18] Chen, X., Zhou, J., and Wei, S. Sfc-ho: reliable layered service function chaining, 2022.

[19] Cheng, B., Fuerst, J., Solmaz, G., and Sanada, T. Fog function: Serverless fog computing for data intensive iot services, 2019.

[20] Christidis, A., Moschoyiannis, S., Hsu, C.-H., and Davies, R. Enabling serverless deployment of large-scale ai workloads, 2020.

[21] Cisco. Cisco annual internet report (2018–2023) white paper, March 2020.

[22] CLOUDFLARE. Why use serverless computing? | pros and cons of serverless, 2018.

[23] COPIK, M., TARANOV, K., CALOTOIU, A., AND HOEFLER, T. rfaas: Rdma-enabled faas platform for serverless high-performance computing, 2021.

[24] CORDASCO, G., D'AURIA, M., NEGRO, A., SCARANO, V., AND SPAGNUOLO, C. Fly: A domain-specific language for scientific computing on faas, 2020.

[25] DAI, H., ZENG, X., YU, Z., AND WANG, T. A scheduling algorithm for autonomous driving tasks on mobile edge computing servers, 2019.

[26] DATADOG. The state of serverless 2021, May 2021.

[27] DATADOG. The state of serverless, Jun 2022.

[28] DE SOUZA, A. B., REGO, P. A. L., CARNEIRO, T., RODRIGUES, J. D. C., FILHO, P. P. R., DE SOUZA, J. N., CHAMOLA, V., DE ALBUQUERQUE, V. H. C., AND SIKDAR, B. Computation offloading for vehicular environments: A survey, 2020.

[29] DELLTECHNOLOGIES. Internet of things and data placement, 2019.

[30] EISMANN, S., SCHEUNER, J., VAN EYK, E., SCHWINGER, M., GROHMANN, J., HERBST, N., ABAD, C. L., AND IOSUP, A. A review of serverless use cases and their characteristics, 2021.

[31] ELORDI HIDALGO, U., UNZUETA IRURTIA, L., GOENETXEA IMAZ, J., LOYO MENDIVIL, E., ARGANDA CARRERAS, I., AND OTAEGUI MADURGA, O. On-demand serverless video surveillance with optimal deployment of deep neural networks, 2021.

[32] FAN, R., LI, X., LIU, Z., ZHAN, C., AND HU, H. Optimal task offloading for deep neural network driven application in space-air-ground integrated network, 2022.

[33] FAN, W., LI, S., LIU, J., SU, Y., WU, F., AND LIU, Y. Joint task offloading and resource allocation for accuracy-aware machine-learning-based iiot applications, 2023.

[34] HADARY, O., MARSHALL, L., MENACHE, I., PAN, A., GREEFF, E. E., DION, D., DORMINEY, S., JOSHI, S., CHEN, Y., RUSSINOVICH, M., AND MOSCIBRODA, T. Protean: VM allocation service at scale, Nov. 2020.

[35] HASSAN, H. B., BARAKAT, S. A., AND SARHAN, Q. I. Survey on serverless computing, Jul 2021.

[36] HAZRA, A., DONTA, P. K., AMGOTH, T., AND DUSTDAR, S. Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial iot applications, 2023.

[37] HLUCHÝ, L., HABALA, O., BOBÁK, M., TRAN, V., AND IVICA, L. Serverless computing and faas for airport meteorology, 2022.

[38] HU, Y. C., PATEL, M., SABELLA, D., SPRECHER, N., AND YOUNG, V. Mobile edge computing—a key technology towards 5g, 2015.

[39] HUBER, F., AND MOCK, M. Toci: Computational intelligence in an energy management system, 2020.

[40] IBM. What is federated learning?, 2022.

[41] INSIGHTS, F. B. Internet of things (iot) market size, share and covid-19 impact analysis, April 2023.

[42] ISLAM, A., DEBNATH, A., GHOSE, M., AND CHAKRABORTY, S. A survey on task offloading in multi-access edge computing, 2021.

[43] JI, Z., CHEN, L., ZHAO, N., CHEN, Y., WEI, G., AND YU, F. R. Computation offloading for edge-assisted federated learning, 2021.

[44] JONAS, E., SCHLEIER-SMITH, J., SREEKANTI, V., TSAI, C.-C., KHANDELWAL, A., PU, Q., SHANKAR, V., CARREIRA, J., KRAUTH, K., YADWADKAR, N., GONZALEZ, J. E., POPA, R. A., STOICA, I., AND PATTERSON, D. A. Cloud programming simplified: A berkeley view on serverless computing, 2019.

[45] KANG, H., CHANG, X., MIŠIĆ, J., MIŠIĆ, V. B., FAN, J., AND LIU, Y. Cooperative uav resource allocation and task offloading in hierarchical aerial computing systems: A mappo-based approach, 2023.

[46] KANG, J., KIM, S., KIM, J., SUNG, N., AND YOON, Y. Dynamic offloading model for distributed collaboration in edge computing: A use case on forest fires management, 2020.

[47] KAUR, K., KUMAR, S., AND BALIYAN, A. 5g: a new era of wireless communication, Jun 2020.

[48] KHAN, S., SHAH, I., TAIRAN, N., SHAH, H., AND NADEEM, M. F. Optimal resource allocation in fog computing for healthcare applications, 2022.

[49] KO, H., PACK, S., AND LEUNG, V. C. M. Performance optimization of serverless computing for latency-guaranteed and energy-efficient task offloading in energy-harvesting industrial iot, 2023.

[50] KONSTANTOUDAKIS, K., BREITGAND, D., DOUMANOGLOU, A., ZIOULIS, N., WEIT, A., CHRISTAKI, K., DRAKOULIS, P., CHRISTAKIS, E., ZARPALAS, D., AND DARAS, P. Serverless streaming for emerging media: towards 5g network-driven cost optimization, Apr 2022.

[51] LEE, B. D., TIMONY, M. A., AND RUIZ, P. DNAvisualization.org: a serverless web tool for DNA sequence visualization, 06 2019.

[52] LI, L., LI, Y., AND LI, R. Double auction-based two-level resource allocation mechanism for computation offloading in mobile blockchain application, Jan 2021.

[53] LIN, L., LI, P., LIAO, X., JIN, H., AND ZHANG, Y. Echo: An edge-centric code offloading system with quality of service guarantee, 2019.

[54] LIN, L., LIAO, X., JIN, H., AND LI, P. Computation offloading toward edge computing, 2019.

[55] LUO, Q., HU, S., LI, C., LI, G., AND SHI, W. Resource scheduling in edge computing: A survey, 2021.

[56] MA, Y., LIANG, W., HUANG, M., LIU, Y., AND GUO, S. Virtual network function service provisioning for offloading tasks in mec by trading off computing and communication resource usages, 2019.

[57] MALMODIN, J., LÖVEHAGEN, N., BERGMARK, P., AND LUNDÉN, D. Ict sector electricity consumption and greenhouse gas emissions – 2020 outcome, 2024.

[58] MANNER, J. A structured literature review approach to define serverless computing and function as a service, 2023.

[59] MAO, Y., YOU, C., ZHANG, J., HUANG, K., AND LETAIEF, K. B. A survey on mobile edge computing: the communication perspective, 2017.

[60] MARTÍN-MARTÍN, A., THELWALL, M., ORDUNA-MALEA, E., AND DELGADO LÓPEZ-CÓZAR, E. Google scholar, microsoft academic, scopus, dimensions, web of science, and opencitations' coci: a multidisciplinary comparison of coverage via citations, Jan 2021.

[61] MAZILESCU, V., AND MICU, A. "technologies that through synergic development can support the intelligent automation of business processes", 2019.

[62] MEENA, V., GORRIPATTI, M., AND SURIYA PRABA, T. Trust enforced computational offloading for health care applications in fog computing, Jul 2021.

[63] METE, M. O., AND YOMRALIOGLU, T. Implementation of serverless cloud gis platform for land valuation, 2021.

[64] MO, J., LIU, J., AND ZHAO, Z. Exploiting function-level dependencies for task offloading in edge computing, 2022.

[65] NGUYEN, H. T., USMAN, M., AND BUYYA, R. Qfaas: A serverless function-as-a-service framework for quantum computing, 2024.

[66] PAOLUCCI, F., SCANO, D., CUGINI, F., SGAMBELLURI, A., VALCARENGHI, L., CAVAZZONI, C., FERRARIS, G., AND CASTOLDI, P. User plane function offloading in p4 switches for enhanced 5g mobile edge computing, 2021.

[67] PARRES-PEREDO, A., PIZA-DAVILA, I., AND CERVANTES, F. Building and evaluating user network profiles for cybersecurity using serverless architecture, 2019.

[68] PENG, K., LIU, P., AND HUANG, T. A privacy-aware computation offloading method for virtual reality application., 2021.

[69] PROIETTI MATTIA, G., AND BERALDI, R. P2pfaas: A framework for faas peer-to-peer scheduling and load balancing in fog and edge computing, 2023.

[70] QUALCOMM. The generative ai economy: Worth up to 7.9, 2023.

[71] REN, J., GAO, L., WANG, X., MA, M., QIU, G., WANG, H., ZHENG, J., AND WANG, Z. Adaptive computation offloading for mobile augmented reality, Dec 2022.

[72] RISCO, S., AND MOLTÓ, G. Gpu-enabled serverless workflows for efficient multimedia processing, 2021.

[73] ROYUELA, I., AGUADO, J. C., DE MIGUEL, I., MERAYO, N., DURÁN BARROSO, R. J., HORTELANO, D., RUIZ, L., FERNÁNDEZ, P., LORENZO, R. M., AND ABRIL, E. J. A testbed for ccam services supported by edge computing, and use case of computation offloading, 2022.

[74] SALEHE, M., HU, Z., MORTAZAVI, S. H., MOHOMED, I., AND CAPES, T. Videopipe: Building video stream processing pipelines at the edge, 2019.

[75] SARKAR, S., WANKAR, R., SRIRAMA, S. N., AND SURYADEVARA, N. K. Serverless management of sensing systems for fog computing framework, 2020.

[76] SHAFIEI, H., KHONSARI, A., AND MOUSAVI, P. Serverless computing: A survey of opportunities, challenges, and applications, nov 2022.

[77] SINGLA, C., AND KAUSHAL, S. Offloading for application optimization using mobile cloud computing, 2016.

[78] SPILLNER, J. Serverless computing and cloud function-based applications, 2019.

[79] STATISTA. Internet of things (iot) connected devices installed base worldwide from 2015 to 2025, November 2016.

[80] STATISTA. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025, November 2023.

[81] STEINBACH, M., JINDAL, A., CHADHA, M., GERNDT, M., AND BENEDICT, S. Tppfaas: modeling serverless functions invocations via temporal point processes, 2022.

[82] TAIBI, D., EL IOINI, N., PAHL, C., AND NIEDERKOFLER, J. Patterns for serverless functions (function-as-a-service): A multivocal literature review, 05 2020.

[83] VAN EYK, E., GROHMANN, J., EISMANN, S., BAUER, A., VERSLUIS, L., TOADER, L., SCHMITT, N., HERBST, N., ABAD, C. L., AND IOSUP, A. The spec-rg reference architecture for faas: From microservices and containers to serverless platforms, 2019.

[84] VARGHESE, B., AND BUYYA, R. Next generation cloud computing: New trends and research directions, 2018.

[85] WANG, A., ZHANG, J., MA, X., ANWAR, A., RUPPRECHT, L., SKOURTIS, D., TARASOV, V., YAN, F., AND CHENG, Y. InfiniCache: Exploiting ephemeral serverless functions to build a Cost-Effective memory cache, Feb. 2020.

[86] WANG, B., WANG, C., HUANG, W., SONG, Y., AND QIN, X. A survey and taxonomy on task offloading for edge-cloud computing, 2020.

[87] WANG, H., NIU, D., AND LI, B. Distributed machine learning with a serverless architecture, 2019.

[88] WANG, K., YAP, L. W., GONG, S., REN, W., WANG, S. J., AND CHENG, W. Nanowire-based soft wearable human–machine interfaces for future virtual and augmented reality applications, 2021.

[89] XIAO, H., XU, C., MA, Y., YANG, S., ZHONG, L., AND MUNTEAN, G.-M. Edge intelligence: A computational task offloading scheme for dependent iot application, 2022.

[90] XIONG, J., GUO, H., AND LIU, J. Task offloading in uav-aided edge computing: Bit allocation and trajectory optimization, 2019.

[91] YAO, X., CHEN, N., YUAN, X., AND OU, P. Performance optimization in serverless edge computing environment using drl-based function offloading, 2022.

[92] YASIN, A., FATIMA, R., WEN, L., AFZAL, W., AZHAR, M., AND TORKAR, R. On using grey literature and google scholar in systematic literature reviews in software engineering, 2020.

[93] YU, M., LIU, A., XIONG, N. N., AND WANG, T. An intelligent game-based offloading scheme for maximizing benefits of iot-edge-cloud ecosystems, 2022.

[94] ZHANG, S., LUO, X., AND LITVINOV, E. Serverless computing for cloud-based power grid emergency generation dispatch, 2021.