


Article

# Exploring the Potential of Distributed Computing Continuum Systems

Praveen Kumar Donta , Ilir Murturi , Victor Casamayor Pujol , Boris Sedlak  and Schahram Dustdar 

Distributed Systems Group, Technische Universitat Wien (TU Wien), 1040 Vienna, Austria; v.casamayor@dsg.tuwien.ac.at (V.C.P.); b.sedlak@dsg.tuwien.ac.at (B.S.); dustdar@dsg.tuwien.ac.at (S.D.)

\* Correspondence: pdonta@dsg.tuwien.ac.at (P.K.D.); imurturi@dsg.tuwien.ac.at (I.M.); Tel.: +43-1-58801-18432 (P.K.D.); +43-1-58801-58413 (I.M.)

**Abstract:** Computing paradigms have evolved significantly in recent decades, moving from large room-sized resources (processors and memory) to incredibly small computing nodes. Recently, the power of computing has attracted almost all current application fields. Currently, distributed computing continuum systems (DCCSs) are unleashing the era of a computing paradigm that unifies various computing resources, including cloud, fog/edge computing, the Internet of Things (IoT), and mobile devices into a seamless and integrated continuum. Its seamless infrastructure efficiently manages diverse processing loads and ensures a consistent user experience. Furthermore, it provides a holistic solution to meet modern computing needs. In this context, this paper presents a deeper understanding of DCCSs' potential in today's computing environment. First, we discuss the evolution of computing paradigms up to DCCS. The general architectures, components, and various computing devices are discussed, and the benefits and limitations of each computing paradigm are analyzed. After that, our discussion continues into various computing devices that constitute part of DCCS to achieve computational goals in current and futuristic applications. In addition, we delve into the key features and benefits of DCCS from the perspective of current computing needs. Furthermore, we provide a comprehensive overview of emerging applications (with a case study analysis) that desperately need DCCS architectures to perform their tasks. Finally, we describe the open challenges and possible developments that need to be made to DCCS to unleash its widespread potential for the majority of applications.



**Citation:** Donta, P.K.; Murturi, I.; Casamayor Pujol, V.; Sedlak, B.; Dustdar, S. Exploring the Potential of Distributed Computing Continuum Systems. *Computers* **2023**, *12*, 198. <https://doi.org/10.3390/computers12100198>

Academic Editor: Isidro Calvo

Received: 4 September 2023

Revised: 26 September 2023

Accepted: 28 September 2023

Published: 2 October 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** distributed computing continuum systems; Internet of Things; edge computing; edge intelligence; artificial intelligence; machine learning

## 1. Introduction

Several decades ago, computing began to rapidly solve complex calculations but was prone to errors; however, now it embodies the transformative power of technology [1]. In the beginning, computers occupied large rooms (huge in size) and had limited processing power, but now, they are portable and perform high-speed processing with extensive memory capacities. Developments in hardware, software (through the evolution of programming languages), and the Internet have helped modern computation encompassing large computations in a parallel and distributed manner [2,3]. In this regard, mainframe-based computing was the first Internet-based distributed computing architecture developed in the early 1960s. Furthermore, the computing paradigm extended to grid computing, cluster computing, cloud computing, and fog/edge computing [4–7]. The accessibility and ubiquity of computing have transformed almost every field, including education, healthcare, industries, entertainment, and space research [8].

Distributed Computing Continuum Systems (DCCSs) is a revolutionary transformation in the computing realm, which is ushering in an era where boundaries between disparate computing environments dissolve, and a seamless continuum emerges [9]. DCCSs

are based on the idea of a continuum of computing resources, where each resource has different capabilities and characteristics. As cloud computing, edge computing, mobile devices, and other computing devices become a unified ecosystem, there are numerous opportunities and challenges [10,11]. This creates a cohesive ecosystem in which resources are dynamically allocated across different tiers in accordance with the specific needs of the task at hand.

DCCSs are more efficient compared with the traditional computing paradigm in several ways. For example, the cloud can be used for heavy computation tasks, the edge for real-time processing, and the fog for local data aggregation [12]. These can be utilized seamlessly while shifting resources as necessary depending on application needs. With this adaptability, resources are more appropriately utilized, and the overall performance is improved. The adaptability of the system not only limits efficient resource usage but also improves the performance in terms of different metrics such as optimized processing power, latency, energy efficiency, and overall system cost. Currently, the majority of applications need high computational power such as machine learning (ML) [13] and artificial intelligence (AI); DCCSs can efficiently perform them using available resources.

Since DCCSs are a growing technology, its features motivate using these technologies in multiple applications. The aim of this paper was to give an overview of DCCSs and compare their efficiency with traditional computing paradigms. First, we discuss the development of various distributed computing paradigms from 1960 to the present day. We discuss each computing paradigm, its working model, components, benefits, and limitations. Then, we analyze DCCSs and their architecture and how it differs from traditional computing paradigms. We analyze the advantages and limitations of DCCSs in various aspects. We identify several applications that desperately need DCCSs and explain them through an illustrative example of DCCS usage over traditional computing models. DCCSs are a growing technology that requires further research to make it more efficient, dynamic, and adaptable. In the interest of simplification, the primary contributions of this paper are summarized as follows:

- Initially, we analyze the evolution of the computing paradigm from the 1960s to current computing trends. We discuss computing paradigm benefits and limitations.
- Next, the potential for DCCSs using various computing devices is discussed, along with their advantages and limitations. In addition, DCCSs' overall benefits and limitations are analyzed.
- Furthermore, we provide various applications and real-time example scenarios wherein computing paradigms are highly needed. We highlight how these use cases benefit from DCCSs.
- Finally, we discuss several open research challenges and possible solutions for future enhancements to DCCSs to make these more efficient.

The remaining sections of this paper are organized as follows. The acronyms used in this paper are listed after Section 6. The year-wise evolution of distributed computing continuum systems is discussed in Section 2. The potential and challenges of DCCSs are discussed in Section 3. In Section 4, various real-time applications that benefit from DCCSs are discussed. A wide range of open challenges and further possible research scopes are discussed in Section 5. Finally, we conclude our paper in Section 6.

## 2. Evolution of Distributed Computing Continuum

This section explains the evolution of the computing paradigm from mainframe-based computing to DCCSs, which are summarized in Figure 1. The rise of networking and the Internet in the late 1960s brought the opportunity for computer systems to work together and simultaneously.

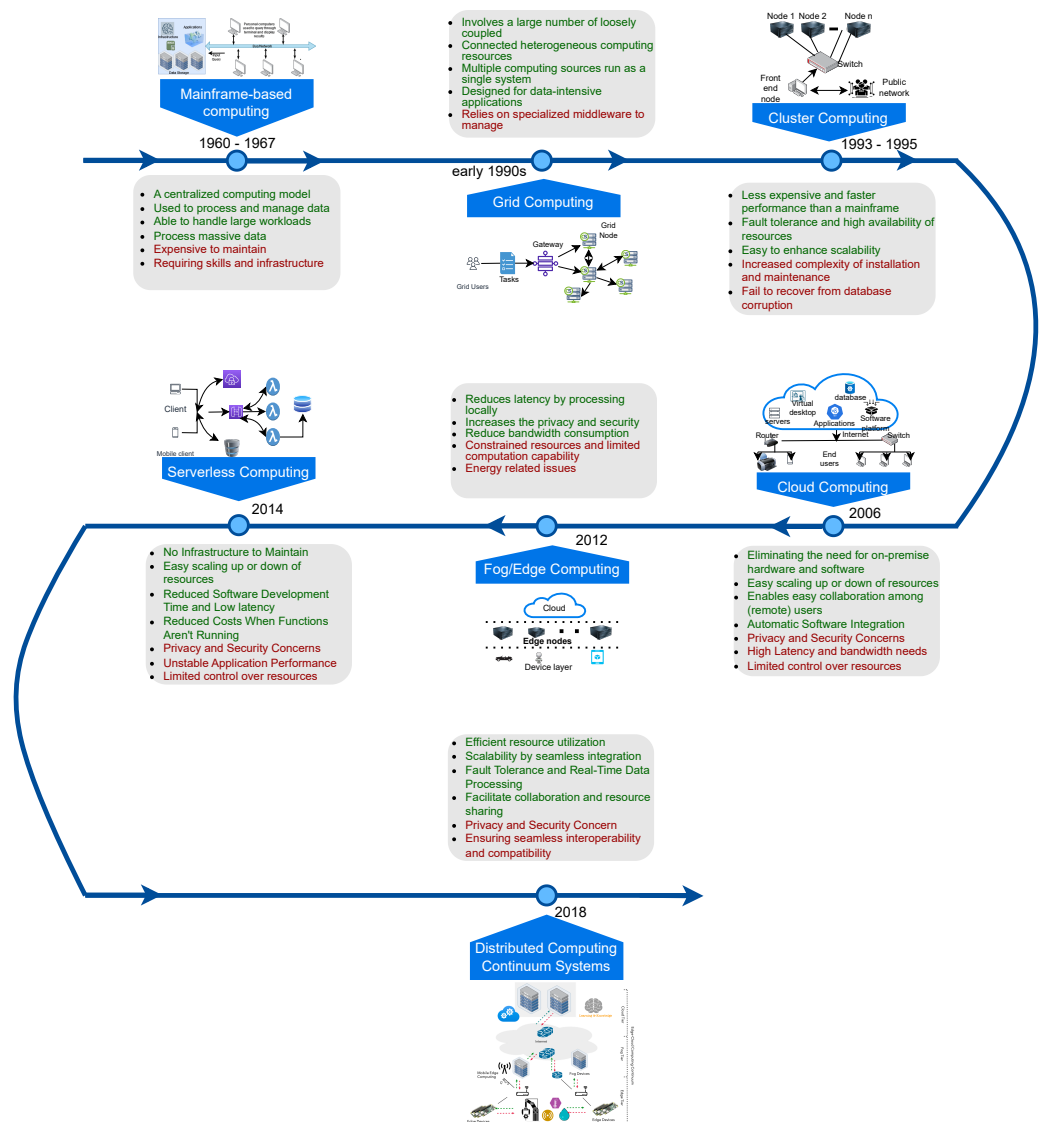


Figure 1. Evolution of distributed computing continuum systems.

### 2.1. Mainframe-Based Computing

Mainframe computers are high-performance computing platforms, performing real-time computations using huge amounts of storage (private databases) and processors (transactional servers). In addition, these use various applications to perform various tasks, such as processing banking data and providing high resilience, agility, and security [14]. A general mainframe-based computing model is replicated using Figure 2.

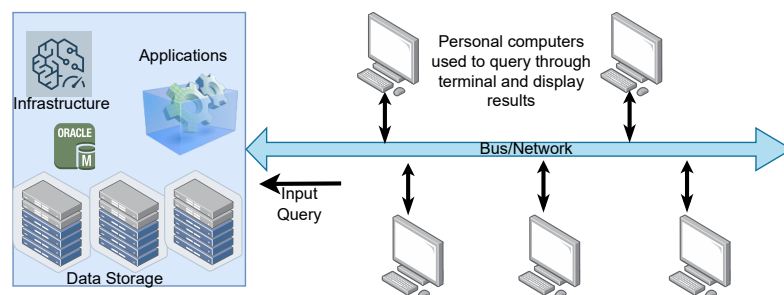
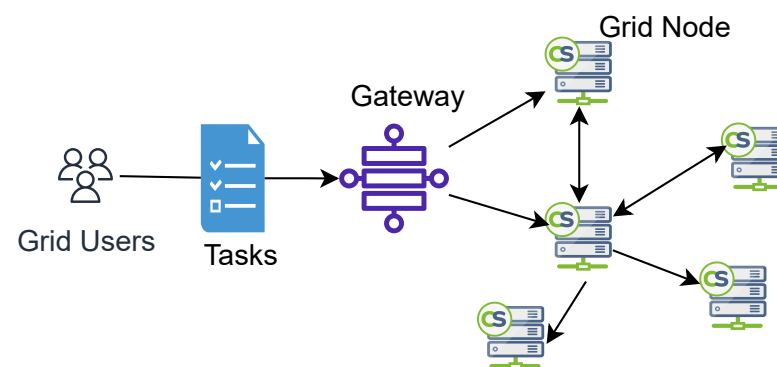


Figure 2. Architecture of mainframe-based computation.

Mainframe computing is designed with a huge amount of resources, so it performs the computations centrally and rapidly. On the other side, the mainframe computing model has several limitations. These are designed for specific tasks and may not be as flexible as other computing models. Thus, they are not suitable for adapting or customizing when the requirements of the business change. They are expensive in terms of hardware and software, and require a specialized person to operate and maintain due to their complexity. Mainframes are large and require dedicated space and infrastructure, so they are not feasible for small organizations. Due to the vendor lock-in model, it is very hard to switch from the current platform to others [15].

## 2.2. Grid Computing

Grid computing involves connecting multiple computers through a network to accomplish common computationally intensive tasks that cannot be performed on a single machine. A network of computers works together as one virtual supercomputer to perform these tasks. In this way, users and applications can seamlessly access IT capabilities by creating a single-system image [16,17]. Figure 3 explains the general working model for grid computing, where the major components are users, grid nodes, and a central server. A user represents a computer or application that requests resources from the grid for further computations. Grid networks are managed by a central control node, typically a server or set of servers. A central node maintains and controls the resources and computational assignments of the network pool. A grid node contributes resources to a network pool (such as memory, processors, and storage). The nodes in this network actively participate in the execution of computations in the distributed grid [18].



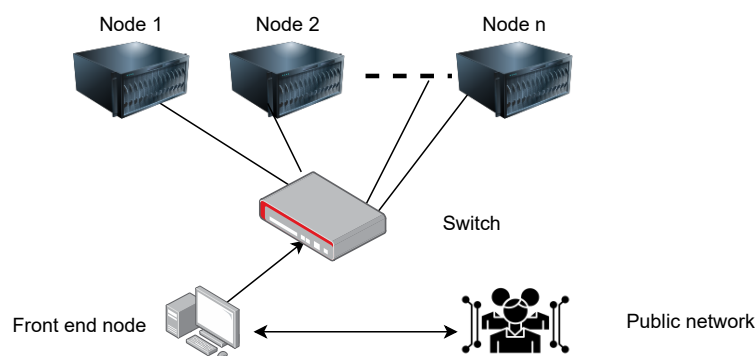
**Figure 3.** General working model for grid computing.

The grid computing model is highly efficient for large computations in a short amount of time. Multiple organizations can collaborate to exchange computation resources with this model. However, there are several limitations associated with grid computing. Since a central server controls the entire grid network, the failure of the central server causes a hotspot with the network pool. However, failing a grid node does not cause more damage because another grid node fulfills the requirements. This model is also not appropriate for small tasks. In order to exchange data between grid nodes, this model also needs a high bandwidth. This model does not support adaptability and interoperability.

## 2.3. Cluster Computing

Cluster computing is similar to grid computing in that multiple computers (called nodes) are grouped together to complete a large task at once. In addition, it provides additional benefits, such as scalability, high availability, and load balancing. A general cluster computing model is depicted using Figure 4. The components of this model include computing nodes (cluster nodes), management nodes, the connecting bus, and shared redundant storage [19]. Computing nodes are individual computers connected to the network and performing computations. Each node in this network uses its operating

system. In a cluster, management nodes monitor the hardware and software as well as reconfigure them as needed. Communications between cluster nodes are made easier with private networks. Switches maintain a connection between cluster nodes [20].



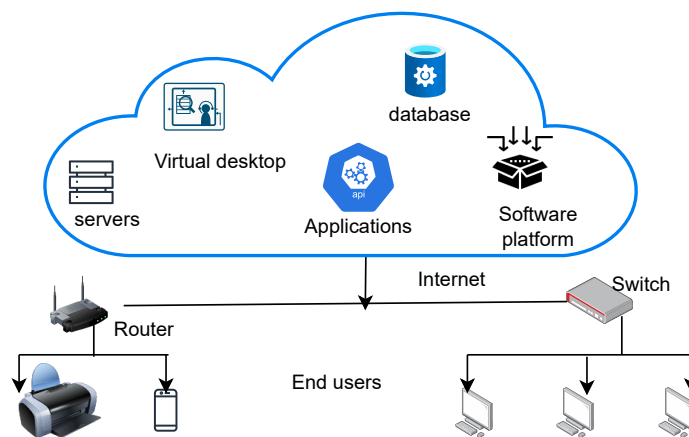
**Figure 4.** High-level architecture of cluster computing.

In cluster computing, complex computational tasks are divided into smaller subtasks distributed among multiple interconnected computing nodes. These nodes perform their assigned subtasks simultaneously, communicating through a network switch and sharing data and intermediate results as required. Once all nodes have completed their tasks, the results are aggregated and delivered to the front-end node [21].

Cluster computing enables parallel processing, partitioning complex tasks into smaller subtasks that can be worked on simultaneously at different cluster nodes, drastically minimizing computation time. Cluster systems are highly scalable, allowing additional nodes to be added as computing demands increase and vice versa. Despite a node failure, fault tolerance mechanisms ensure uninterrupted operation by assigning the computational subtask to an available node in the cluster. Data-intensive tasks are handled more efficiently with clusters because they can share data and communicate efficiently. However, cluster computing is highly scalable and can perform computationally intensive tasks, but it also comes with limitations. Some computations may not lend themselves to easy parallelization due to dependencies, making it difficult to efficiently divide tasks into parallelizable subtasks. Additionally, synchronizing the results in a particular order is also challenging. Setting up and maintaining a cluster, including the hardware, network, and software configurations, can be challenging. Adding numerous nodes will be more difficult due to communication and coordination overhead. In addition to ensuring load balance across nodes, fault tolerance and reliability can also be complex issues to address.

#### 2.4. Cloud Computing

In recent decades, cloud computing has emerged as a significant technological advancement. It is widely recognized as a crucial computing environment for facilitating the growth, implementation, and operation of IoT platforms. In this context, businesses can transfer their control, computing capabilities, and accumulated data to a platform with nearly limitless resources [22]. Today, cloud computing continues to stand out as a widely accepted solution for deploying resource-intensive computational tasks, particularly for processing vast volumes of data generated from IoT devices spread across various geographical locations (i.e., including sensors, smartphones, laptops, and vehicles). The cloud computing paradigm offers extensive resources, enabling the deployment of diverse platforms, virtual machines, multiple databases, and various applications. With its scalability and flexibility, the cloud empowers users to harness a vast pool of computing power and storage, allowing them to create and manage these computing environments tailored to their specific needs (see Figure 5).



**Figure 5.** High-level architecture cloud computing.

Nonetheless, the cloud computing paradigm encounters several challenges in effectively addressing the rigorous demands of emerging IoT applications. For instance, continuously transmitting sensory data to the cloud leads to higher latency than expected in IoT system responses. From one perspective, transferring vast and intensive data volumes to a centralized cloud through wide-area networks (WANs) gives rise to latency issues. On the other side, privacy aspects, connectivity concerns, or planned system maintenance (specifically, on the cloud side) can lead to service unavailability, presenting a potential hurdle for essential IoT systems throughout their operational duration.

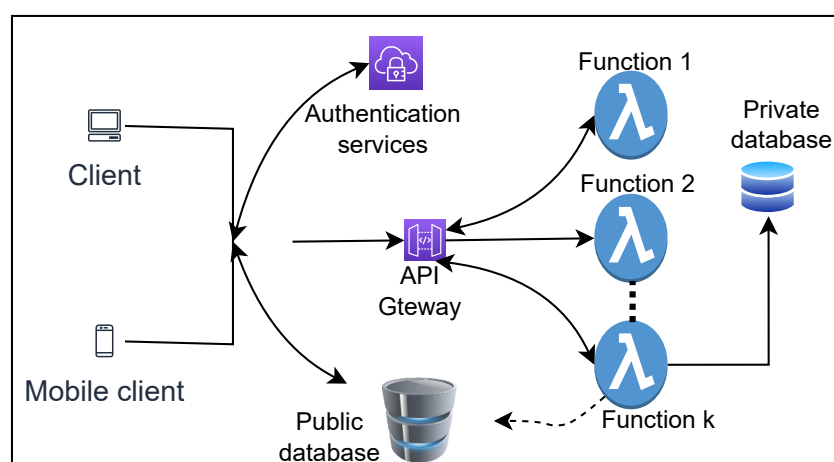
### 2.5. Fog and Edge Computing

Fog computing, a concept pioneered by Cisco [23,24], aims to enhance cloud capabilities by bringing them closer to edge networks. Various interpretations are currently available in the literature, but the most relevant ones are outlined in [25–27]. Fog computing is defined as geographically distributed computing devices that connect numerous heterogeneous devices at the network’s edge; however, it does not solely rely on seamless integration with cloud services. In essence, fog is a vital link between the network’s edge and the cloud. The fog devices in proximity enable the facilitation of the implementation of emerging IoT applications with low latency and stringent requirements [28,29]. As introduced in [30], edge computing represents a novel paradigm that aims to provide storage and computational capabilities in proximity to end-users and the IoT domain. The term “edge” encompasses all computational and networking resources along the trajectory between the primary data source and the ultimate data repository (including fog nodes and cloud data centers) [31,32].

As can be observed, the edge and fog paradigms offer nearly indistinguishable attributes. Both concepts anticipate enhancing computational capabilities close to end-users and within the IoT domain. However, the primary contrast between these two paradigms lies in administrative distinctions and responsibilities [33]. Additionally, fog nodes (e.g., those stationed in base stations) might extend their services across more expansive geographical regions. For example, intelligent transportation systems could gain advantages by linking and analyzing vehicle data within the fog infrastructure. Nevertheless, both paradigms furnish low-latency services due to end-devices’ proximity to the data source, facilitating production and consumption. Furthermore, edge and fog devices typically have limited computational resources, storage capacity, and power compared to centralized cloud servers. This limitation can impact various application types that can be efficiently run at the edge, and resource-intensive tasks might still need to be offloaded to a more powerful cloud infrastructure.

## 2.6. Serverless Computing

Serverless computing is a cloud-native development model that allows developers to build and run applications without managing servers [34,35]. It is an approach to software design that abstracts away the underlying infrastructure, allowing developers to focus on writing code and deploying applications [36]. Serverless computing works as follows. Developers write their code and package it in containers or functions, deploying them to a serverless platform provided by a cloud vendor [37,38]. Since these functions are event-driven, the serverless platform automatically scales up the necessary resources to execute the function when an event occurs (such as a request or query being received). The cloud provider dynamically assigns the necessary computing resources and storage to run the task. All routine infrastructure management tasks, including provisioning, scaling, and maintaining the server infrastructure, can be managed by the cloud provider by default [39]. A high-level serverless computing model is described in Figure 6.



**Figure 6.** A high-level working model for serverless computing.

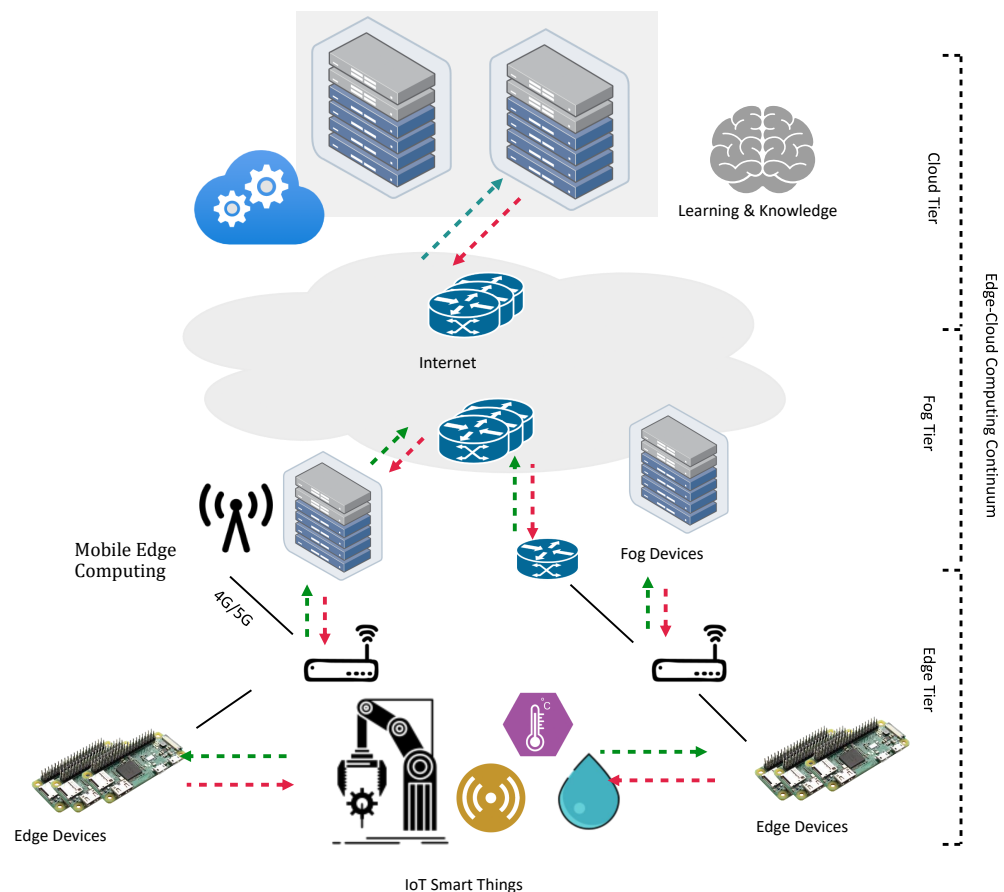
In Figure 6, the primary components of serverless computing can be seen as the end-user, content delivery network (CDN), API gateway, serverless functions, and database sources. The end-user interacts with the application through APIs. CDNs are global networks of distributed servers that cache and deliver content to end-users. The API gateway manages and routes end-user requests to specific serverless functions and acts as middleware between users and serverless functions. This function is a piece of code that runs in response to an event or request generated by an end-user. It is executed on a serverless platform and automatically scales up or down based on resource demand. A data source (private or public database) provides data to serverless functions [39].

Serverless computing offers several benefits over cloud and edge computing, such as low cost, scalability, and no need to maintain infrastructure [40]. It is more cost-effective than traditional clouds as developers do not need high computational power or space. For small organizations, the serverless computing paradigm is the most appropriate solution. Since all infrastructure maintenance is taken care of by cloud vendors, infrastructure maintenance is not a burden in serverless computing. Scalability is the primary advantage of cloud computing since it can automatically scale up or down based on the requirements of the customer [41]. Furthermore, serverless computing has multiple limitations. Serverless computing can lead to vendor lock-in, as developers may become dependent on a specific cloud provider's serverless platform and APIs. It can be difficult to optimize performance or troubleshoot problems in this environment because very little control is available over the underlying infrastructure and execution environment. While it is cost-effective for small organizations, it has become more expensive than traditional cloud computing for applications with high traffic and larger workloads. Serverless functions can experience

cold start latency, which is the time it takes to start up serverless functions and execute the first request, sometimes causing high latency.

### 2.7. Distributed Computing Continuum Systems

DCCSs are systems built of a large variety of heterogeneous networked computing devices, which are used to process data generated by devices such as sensors, mobile devices, and IoT devices. Through its integration of cloud, edge, and IoT resources, it enables efficient, real-time, and dynamic computations to meet the needs of today's diverse applications [42,43]. It performs computations by distributing the workload across multiple devices in the system. Each device performs a portion of the computation, and the results are combined to produce the final output. This allows for faster processing times and increased scalability. With DCCSs, computations are accomplished efficiently while adapting to changing demands and optimizing resource utilization outside traditional boundaries [9]. This resource allocation is based on factors like resource proximity, computational capability, and prioritizing time-sensitive tasks. Depending on the task, real-time responses may be offloaded to edge devices, while complex analytics may be conducted in the cloud by default. This dynamic distribution of tasks enhances system performance and processing efficiency whilst reducing latency. The general architecture of DCCS is illustrated through Figure 7.



**Figure 7.** General architecture for distributed computing continuum systems.

**Cloud Computing vs. DCCSs:** In cloud computing, users access virtualized servers, storage, and applications hosted by a cloud provider over the Internet [44]. In DCCSs, a wide array of resources are incorporated, including edge devices, IoT sensors, mobile devices, and even cloud servers. However, computations are distributed from localized processing to centralized cloud analytics as needed. In contrast, DCCSs dynamically assign tasks to



the most appropriate resource based on factors such as proximity, processing capability, and data sensitivity, minimizing latency and maximizing resource utilization. Cloud infrastructure involves provisioning resources according to predetermined configurations and subscription plans. User capacity can be adjusted based on their needs until vendor lock-in occurs with scalable cloud computing. Moreover, DCCS functionality can evolve naturally based on resources and demand, enabling flexibility and effective resource utilization. It is possible to process data on edge devices instead of cloud servers when a task requires an immediate response or sensitive data that are too latency-sensitive to be processed centrally.

**Edge Computing vs. DCCS:** Using edge computing, data are processed near its source, reducing latency and conserving bandwidth. Low-power devices, such as sensors and gateways, are usually treated as edge servers [45]. In contrast, DCCSs integrate not only edge devices but also cloud servers and various computing resources, which allow it to dynamically allocate tasks across available devices, optimizing resource utilization, enhancing responsiveness, and enabling real-time processing. In contrast to edge computing, DCCSs are capable of adaptive and efficient computation beyond the capabilities of individual devices by harnessing the power of an extensive array of resources. DCCSs support fault tolerance, whereas edge computing does not. A device failure does not interrupt computation, and the task is moved to another edge server or cloud server in DCCSs.

**Serverless Computing vs. DCCS:** In serverless computing, resources are automatically scaled based on demand, and users are billed just as they use them. Using DCCSs, tasks are routinely distributed according to proximity, capacity, and urgency. In contrast to serverless computing, DCCSs primarily focus on resource efficiency, integrating a wide range of devices, and enabling real-time processing across the continuum (edge-to-cloud).

### 3. Potential of Distributed Computing Continuum

DCCSs fundamentally differ from traditional computing models because they seamlessly integrate diverse computing resources. Integrating powerful cloud servers with agile edge devices and ubiquitous mobile endpoints is part of this process. By leveraging the strengths of each component, a holistic ecosystem is created to meet the diverse needs of modern computing. In this context, classes of computing devices used in DCCSs are initially explained. DCCSs offer a path towards a future where computing seamlessly adapts to application needs. This section answers how DCCSs can deliver new levels of efficiency and responsiveness by exploring the core benefits [46] and challenges [42].

#### 3.1. Classes of Computing Devices Used in DCCSs

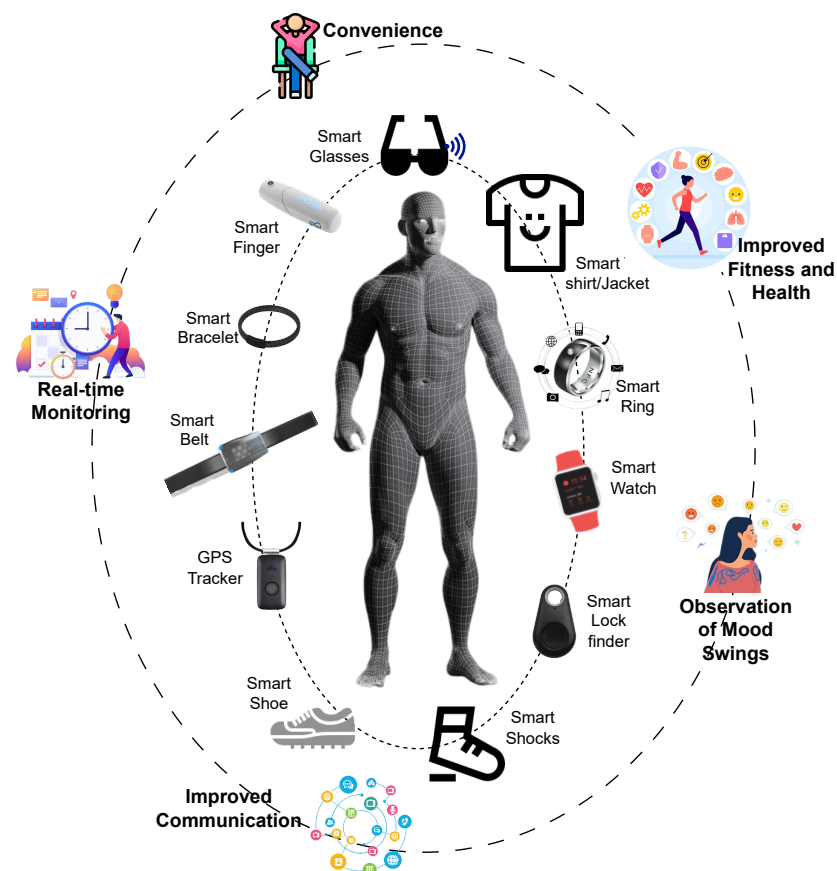
There are classes of computing devices working together in DCCSs to enable seamless data generation, processing, analysis, and communication across the system. In our paper, we categorize these devices into five groups: embedded computers, IoT, mobile devices, desktop computers, and servers.

##### 3.1.1. Embedded Computers

Embedded computing systems are electronic or mechanical devices equipped with computing hardware (such as processors—either a microprocessor or microcontroller, power supplies, sensors, actuators, communication mechanisms, and memory) and software components to perform a specific task [47–49]. In recent years, several devices have been equipped with embedded computers, such as diagnostic and patient monitoring tools in healthcare, automated teller machines (ATMs), engineering calculators, digital cameras, household appliances (digital door locks, automatic washing machines, or microwaves) autonomous vehicles, and many more. Since smart and remote applications have become increasingly popular, all these devices are connected to the Internet and operate remotely. An embedded computer can be classified according to its application or device: small-scale (using 8-bit microcontrollers), medium-scale (using 16-bit or 32-bit microcontrollers, or multiple small-scale embedded computers), or sophisticated-scale (with complex hardware and software). With this hardware and software, they can make decisions autonomously

and operate accordingly. Recently, these devices were connected over the Internet, and their data are transmitted and analyzed in the cloud to monitor and predict the devices' condition and efficiency. Gateways and routers used in networking are also embedded computing devices. Gateways translate a message from one protocol to another, which is programmed into its chip. It acts as middleware between physical and network layers. Routers can maintain routing tables and transmit the data packets from one network interface to another. Depending on their configurations, these two devices can also perform additional preprogrammed tasks.

These days, wearable devices are commonly used as embedded devices that users wear and seamlessly integrate into their daily routines. These devices range from smartwatches and fitness trackers to augmented reality glasses. They are equipped with sensors, processors, and connectivity features that enable them to capture data and interact with users and other devices [50]. Wearables often monitor health metrics, track physical activity, deliver notifications, and provide quick access to information. They act as intermediaries between users and the broader network of devices due to their portability and constant presence. An illustrative example of wearable devices and their benefits is depicted in Figure 8. The benefits of these devices are wide, and include convenience, monitoring fitness and health, mood swing monitoring, communication with the environment, people or other devices remotely, real-time analytics and monitoring, and safety [51–53]. In 2020, the wearable technology market was valued at USD 59 billion, and by 2024, it should be valued at USD 156 billion, with an annual growth rate of 24.6% according to the GlobalData (Available online: <https://www.medicaldevice-network.com/comment/wearable-technology-iot>, last accessed 3 September 2023). However, most of these devices have very limited computing capacity, and their analytics depend on other computing devices in the continuum.



**Figure 8.** Wearable embedded computing devices for the human body and some of their benefits.

### 3.1.2. Internet of Things

IoT devices are also a category of embedded computing devices that differ in terms of connectivity, communications, and functionalities. As discussed in the previous subsection, embedded computing systems are standalone systems designed for specific tasks, typically operating in isolation from other systems. While some systems can be part of an IoT network, not all embedded systems are inherently connected to the Internet. However, IoT is an interconnected set of physical devices that communicate and exchange data over a communication medium [54]. Often, these devices collect data from their environment [55], transmit these over the Internet, and interact with other devices [56]. A variety of communication protocols are frequently used to transmit data and receive commands from remote locations, including Wi-Fi, cellular networks, Bluetooth, Zigbee, or LoRa [57,58]. In general, embedded computing systems are designed with a specific purpose in mind and have limited reconfigurability and remote update capabilities. As IoT devices become more versatile, they can often be configured or remotely updated to accommodate changes to their requirements or capabilities. IoT can be used for multiple purposes instead of using them for specific tasks. Some IoT devices have their own operating systems (Raspberry Pi), which can act in the same way as personal desktop computers.

It is estimated that 14.3 billion IoT devices will be active around the world by the end of 2022. In 2032, this market will have grown to 34.4 billion, a trebling courtesy of an annual compound growth rate of 10% (Available online: <https://www.spiceworks.com/tech/iot/guest-article/iot-key-trends-over-next-decade/>, last accessed 3 September 2023). The IoT healthcare market is expected to be worth more than a trillion dollars by 2032, with 86% of healthcare organizations already utilizing IoT services (Available online: <https://www.linkedin.com/pulse/iot-landscape-next-decade-internet-things-rp-international/>, last accessed 3 September 2023). Transport systems are not exceptional due to growing autonomous vehicle research. Several other industries are expected to adopt IoT in the coming days, such as building automation and security, consumer Internet and media devices, inventory management and marketing, asset tracking and monitoring, agriculture [59], and smart grids [60,61]. Through 4G and 5G, the IoT market is growing steadily, and when 6G is adopted, it might increase even faster [62–64]. In 2032, the amount of data from these ever-growing devices may reach 150 Zettabytes, and edge devices may not be able to handle it. So, computing continuum systems are a solution to use all computing devices in a continuum to quickly complete tasks and make decisions [65].

### 3.1.3. Mobile Devices

The evolution of mobile devices has been remarkable over the past two decades, from basic cell phones to powerful smartphones and tablets. Following the advent of voice and text communication, smartphones, including Apple's iPhone and Google's Android, introduced touchscreens, app ecosystems, and enhanced capabilities. Tablets filled the gap between smartphones and laptops. Fitness trackers and smartwatches became popular for monitoring health and receiving notifications on mobile devices. Mobile devices have become indispensable tools for communication, entertainment, and productivity as a result of the transition from 2G to 5G networks, the proliferation of apps, and the convergence of technologies such as augmented reality (AR) and virtual reality (VR) [66,67]. Processors in smartphones are sometimes categorized as desktop computers due to their hardware similarities. However, they're distinct due to their capacity to execute externally developed software. For storage purposes, Flash memory is used instead of disks due to its energy and size requirements. A device's energy efficiency is influenced by battery power and heat dissipation. In such cases, memory size is crucial since it accounts for a significant portion of the system cost [68].

Mobile devices serve as capable edge computing devices by performing tasks like data preprocessing, local analytics, and content caching, resulting in reduced latency and improved system efficiency [69,70]. In addition to augmented reality, video streaming, and IoT sensor data analysis, these devices leverage their computational resources to process

data in real-time [71]. The offloading of computation to smartphones at the edge can reduce network congestion, resulting in quicker responses and better user experiences [72]. By 2028, the smartphone market size is forecast to reach 1.78 billion units, at a CAGR of 4.10%, from 1.45 billion units in 2023 (available online: <https://www.mordorintelligence.com/industry-reports/smartphones-market>, last accessed 3 September 2023). In this way, smartphones facilitate seamless integration between edge, fog, and cloud computing within DCCSs by operating as mobile edge devices. They will participate in real-time data processing with low latency and optimized bandwidth.

Recently developed large language models (LLMs) are becoming popular due to their benefits, such as efficiency, customization, understanding different languages, and automating tasks. But LLMs require large computing resources; hence, all the data move to the cloud for computation [73–76]. Currently developed hybrid AI architectures can recommend different offload options based on factors like model complexity and query size to distribute processing among the cloud and other computing devices [77,78]. However, current demands such as cost, energy, reliability, performance, latency, privacy, and security require local computations such as the edge or smartphones. Since over 10 billion searches are conducted every day, with mobiles accounting for over 63% of searches (available online: <https://www.statista.com/statistics/297137/mobile-share-of-us-organic-search-engine-visits/>, last accessed 3 September 2023), the adoption of generative AI will lead to a substantial increase in computing capacity, especially from queries made on mobile devices. This study shows the demand for on-device AI computations [79,80]. It has been announced that Qualcomm is developing a mobile chipset for smartphones called ‘Snapdragon 8 Gen 2’ that will run AI on-device (available online: <https://www.cnet.com/tech/mobile/generative-ai-is-coming-to-phones-next-year-thanks-to-qualcomm-chips/>, last accessed 3 September 2023). This further enhances smartphones’ performance and helps process personal data within the computing continuum.

#### 3.1.4. Desktop Computers

These are the primary computing devices for daily tasks, professional work, and gaming. Desktop and laptop computers have played an increasingly important role in recent decades. These devices have a high level of computational power, energy, and memory compared with smartphones or tablets [81,82]. In addition to content creation, scientific computation, and software development, GPUs can also run AI/ML applications. Laptops are also indispensable for remote work, education, and collaboration. While these devices provide secure local data storage, their adaptability keeps them relevant in an ever-changing technological landscape. Currently, available cloud storage options can provide storage based on the needs of the user. Over 68 million desktops are expected to be shipped worldwide in 2023, down from over 76 million in 2022. It is estimated that desktop shipments will reach 69 million in 2027. Compared to 207 million units in 2022, notebook shipments are projected to reach more than 214 million units in 2027. A notebook is a type of laptop that is smaller and lighter than a laptop, so they are gaining more attention than desktop or laptop computers (available online: <https://www.statista.com/statistics/272595/global-shipments-forecast-for-tablets-laptops-and-desktop-pcs/>, last accessed 3 September 2023).

Desktop computers and laptops are key components of grid and cluster computing, where multiple devices work together to tackle complex tasks. They serve as individual computation nodes, contributing their processing power and capabilities to the overall computing capacity of the system. Additionally, this feature can be used for large workloads at the edge of the computing continuum. Furthermore, these devices manage and store data, host middleware for communication and coordination, and provide monitoring and control interfaces. Due to their role as edge devices, they reduce latency by processing data locally with load balancing and security measures. Desktop and laptop computers are essential to DCCS operations, often forming hybrid architectures, making them indispensable to complex computing environments.

### 3.1.5. Servers and Supercomputers

A server is a computer or software system that operates on a network, processing incoming requests and providing specific services and resources to clients. In turn, it processes client requests according to its function (e.g., web, email, file, or database server), efficiently manages resources such as CPU and memory, and communicates responses back to clients. A supercomputer is a high-performance, parallel processing machine that is highly powerful and specialized. It is ideal for scientific research, engineering simulations, and complex computations. With immense processing speed and customized hardware, supercomputers can solve large-scale problems quickly [83–85]. Cloud computers combine both of these characteristics. Moreover, they provide security features to protect data and systems, store data on storage devices, log activities for monitoring and troubleshooting, and can scale up or down as needed. Due to their computing and memory capabilities, they can perform complex computations and store huge volumes of data. There are certain limitations, including dynamic resource allocation, struggling to efficiently handle sudden surges in demand, latency, and energy. In 2022, the global supercomputer market was valued at USD 8.8 billion. By 2032, it is expected to reach a market value of USD 24.87 billion and grow at a CAGR of 11% (available online: <https://www.statista.com/statistics/568431/hpc-server-revenue-worldwide/>, last accessed 3 September 2023). With their computational power, speed, and capabilities, they will play a pivotal role in DCCSs.

### 3.2. Benefits

DCCSs' architectural agility provides several benefits, including bandwidth optimization, scalability, low latency, efficient resource usage and load balancing, resilience, flexibility, and reliability. Since the list is small, depending on the application and nature, DCCSs can offer a wide range of benefits. We provide an example for a better understanding of each benefit.

**Optimize bandwidth:** In DCCSs, computation tasks are intelligently distributed between edge devices and centralized cloud resources. This distribution minimizes the need to continuously transfer high-bandwidth data, since only essential data (when local device resources are insufficient) or insights are transmitted to the cloud. DCCSs prioritize local processing at the edge, reducing bandwidth demands and enhancing response times compared with cloud computing, which often sends data back and forth between devices. Additionally, it reduces the need for extensive data transfers by utilizing localized caching and processing. DCCSs are particularly well suited to scenarios with limited or unreliable connectivity due to its dynamic approach that conserves bandwidth and accelerates decision making.

**Scalability:** DCCSs demonstrate scalability by dynamically distributing computation tasks across diverse resources. Consider a scenario for a better understanding of the scalability feature in DCCSs. Suppose a smart city uses DCCSs for traffic management. The system may use edge devices and local servers to process real-time traffic data during regular traffic hours. Suppose the system detects an increase in traffic (such as during morning or evening hours) or unexpected traffic surges. In that case, additional resources can be integrated (such as the cloud) to handle the increased load without compromising performance. Scalability is especially advantageous when workloads fluctuate or demand spikes suddenly, since DCCSs effectively utilize available resources without overwhelming any one component. Due to this architectural agility, DCCS can easily accommodate the growing computational needs of modern applications and services.

**Low latency:** DCCS achieves low latency because it processes tasks close to the data source, rather than sending data long distances as cloud environments do. On the contrary, cloud-based models require sending data to a remote cloud server for processing, introducing network latency that can significantly delay the response. For instance, in smart city applications where traffic management plays a pivotal role in ensuring efficient real-time responses, low latency is extremely important. Consider the scenario of an accident

that causes traffic congestion on a busy road. Sensors deployed across the roadway can detect/predict this congestion and immediately notify nearby edge servers. With their processing capabilities, these edge servers can analyze information instantly and make decisions in a timely manner. For instance, the decision-making system can adjust traffic signals in real-time, reroute traffic from a congested route, or instantly dispatch emergency services. DCCSs' localized processing effectively minimizes latency by allowing immediate analysis on local computing devices (such as edge servers), so that appropriate action is taken quickly.

**Optimized resource utilization and load balancing:** DCCSs ensure optimal resource utilization through efficient and dynamic resource allocation across the continuum. For example, consider a manufacturing facility that uses DCCSs to control quality in real time during production. A variety of sensors or cameras are integrated into the production line in order to capture product parameters, which need to be further analyzed. Depending on resource availability or computation intensiveness, DCCSs can dynamically allocate these data to edge nodes or the cloud. Basic data preprocessing and initial analysis can be carried out at the edge, where complex analyses (such as image or video analytics or AI/ML tasks) can be transferred to the cloud. Additionally, DCCSs can federate tasks among edge servers depending on computational needs and resource availability, which minimizes bandwidth usage and latency even further.

**Resilience, flexibility, and reliability:** By distributing tasks across a diverse set of resources, DCCSs guarantees resilience, flexibility, and reliability. Distributing tasks across resources makes it possible to keep the system running even if one part of it is compromised. Consider the case of a disaster (such as a hurricane) that requires an emergency management system for a smart city. A network of sensors is deployed throughout the city to read weather conditions, water levels, and structural integrity. Data from these sensors are transmitted to local servers or the nearest edge server for initial analysis, which helps identify potential hazards. Unfortunately, if these servers fail to respond due to power outages, damages due to disaster, or connection issues, the DCCS can immediately transfer to another working edge server. The data can be sent to the cloud if there are no active edge servers or local servers in the city. The emergency management system becomes more resilient, flexible, and reliable, allowing disasters to be handled effectively even under adverse conditions when DCCSs are used.

### 3.3. Challenges

DCCSs offer many benefits and have the potential to transform modern computing, but they are not without challenges. In this section, we provide key limitations worth considering.

**Interoperability:** DCCSs are multi-proprietary. This means that the infrastructure resources and their associated middle-ware layers belong to different organizations. One can imagine an application running some services in-house, some services with high computational needs in the Cloud, some latency-sensitive services in fog nodes next to the networking stations, and finally, some other services at the edge to enhance responsiveness and reduce overall bandwidth requirements. Interestingly, each set of nodes might be owned by a different organization. Hence, each has different semantics. Therefore, the application (based on all these services) needs to tackle the usage of very different devices, which, on top, have different owners with, perhaps, different priorities when designing their systems.

**Complexity of Governance:** Currently, Internet-based systems are governed through the application logic and only residually at the infrastructure level by cloud orchestrators, which can basically run more copies of an existing job or schedule new jobs. Also, these are typically centralized entities, which clearly do not fit with the requirements for DCCSs. Another interesting aspect of current Internet-based systems is their usage of service-level objectives (SLOs) to set the minimal performance indicators for these systems. Unfortunately, current SLOs are only low-level metrics (such as CPU usage) or time-related metrics

(such as end-to-end response time). Using SLOs for DCCSs seems appropriate. However, we identify two key aspects that need to be improved:

1. They would need to be able to cover all aspects/components of the system so that the governance strategies are aligned regardless of what is being controlled.
2. Their granularity is adequate to perform surgical interventions. Simply put, if the SLO is on end-to-end response time and it is violated, discovering which is the specific service/device/component/aspect that is producing the delay can be an overwhelming task, which cannot comply with time-constraint requirements.

**Data synchronization:** In the DCCSs, data are constantly generated, updated, moved, and accessed across a wide range of distributed devices, and it is necessary to ensure consistency (through proper synchronization mechanisms [86]) across the continuum. Maintaining data integrity, coherence, and consistency becomes increasingly difficult as data are processed and modified at different locations and speeds. Sometimes, end-to-end delays, network issues, and varying computational speeds (due to resource availability or constraints) can lead to inconsistencies or conflicts between data versions. Furthermore, data synchronization across hybrid setups involving diverse computational resources (cloud, edge, constrained IoT, or sensor nodes) presents additional challenges due to varying processing capabilities and connectivity limitations [87]. In DCCSs, sophisticated synchronization mechanisms are required to ensure that all components can access up-to-date and accurate data.

**Sustainability and energy efficiency:** In terms of sustainability, there are *two* key aspects to consider:

1. The vast amount of computing devices and connections;
2. Their energy sources.

Regarding the first consideration, the computational infrastructure will keep increasing in the coming years. However, it is important that we understand the need to reuse existing infrastructure to limit the need to add new resources. Unfortunately, this challenges previous topics such as governance, interoperability, and others, as dedicated resources are always easier to incorporate into the system than older ones with, perhaps, a different initial purpose. The second sustainability consideration relates to the energy sources that are used in computing systems. It is clear that AI-based systems require high amounts of energy. Hence, being able to harvest this energy from renewable sources is of great interest. Unfortunately, solutions that can do that also require control over the energy grid, which is usually not the case.

Energy efficiency relates to sustainability with the idea of using the minimum energy required for any job. This translates to choosing the right algorithm/service/device/platform for each case, which requires solving very complex multi-variate optimization problems.

Additionally, energy efficiency is key for energy-constrained devices, such as all those devices that are not permanently linked to the energy infrastructure. These require that their usage is compatible with their energy-loading/unloading cycles so that they are always available when needed.

**Privacy and security:** In DCCSs, privacy and security are inherent problems because of their complex structure built on resources, edge devices, cloud platforms, and data transmissions. A majority of privacy and security issues arise from sharing data, communicating over networks, and sharing resources across a continuum. Maintaining consistent security measures becomes more difficult due to dynamic scaling and resource sharing. To ensure the privacy and security of data, resources, and communication across the continuum, encryption, access controls, monitoring, and compliance are required.

#### 4. Applications

DCCSs are capable of seamlessly integrating a wide variety of computing resources, allowing them to perform applications across various domains. In this section, we discuss

a few applications (industry automation, transportation systems [88], smart cities [89,90], and healthcare [91]) with a use case example to show the difference between current technologies with a computing continuum. As a result of adopting DCCS features, these applications can benefit from better resource utilization, faster decision making, and several other benefits, depending on the application requirements.

#### 4.1. Industry Automation

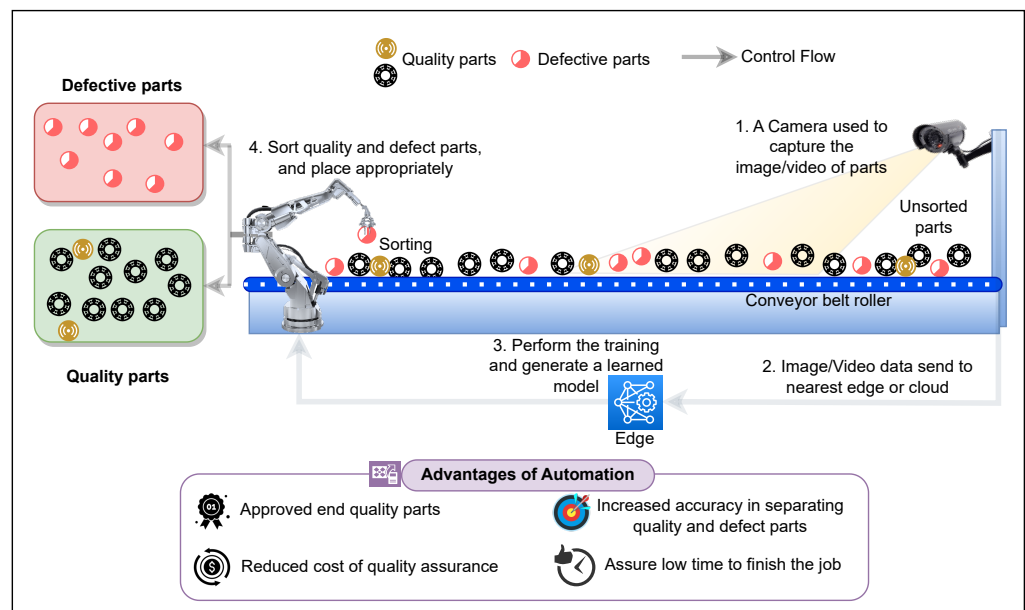
Industrial applications encompass a wide range of sectors and industries where technologies are utilized to enhance processes and operations. There are several applications that fall under industrial applications, such as manufacturing automation, smart grids, food and beverage packaging and quality assurance checks, environmental monitoring, and process control (for example, oil refining or pharmaceuticals). Most of these applications are automated through machine technology, improving efficiency, productivity, and safety. When machines malfunction, their efficiency is reduced, safety hazards are present, and maintenance costs are increased. In order to minimize machine failure and increase efficiency and productivity, preventive measures, such as regular maintenance, real-time monitoring, and predictive maintenance, should be implemented. In this context, IoT is widely used in industrial applications to collect data from machines and then send it to the cloud/edge for further analysis. DCCSs are further advanced in industrial applications and provide efficient and continuous monitoring through real-time analysis (limited interactions with central servers).

**Use case:** Separating the quality and defective parts in a manufacturing industry.

In manufacturing industries such as mobile assembly, food packing, or robot manufacturing, identifying quality parts before assembling or packing is a very time-consuming and tedious task. There is a huge chance for manual errors to lead to overall quality control tasks becoming hectic. Thus, most manufacturing industries turn to automation and perform defect parts separation using machinery. A basic quality and effective part separation system in manufacturing automation industries is discussed in Figure 9. We consider a rotating conveyor belt that moves unsorted parts (both quality and defective). In order to monitor all the objects moving on the belt, a camera was installed to capture images/videos and send them to the nearest computing device. It performs the analysis and sends the trained model to the robot that is armed. In accordance with the provided training model, the robot can determine whether or not the object it receives is a quality part or a defective part and can place it accordingly. The accuracy of the training model determines the overall performance and quality of the separated parts. Accuracy and the timely generation of robot training models determine the overall performance of end-product quality.

DCCSs are capable of analyzing images or videos captured by cameras on conveyor belts in real time. It distributes the processing load among edge devices and cloud resources according to resource availability, which always produces faster. Through real-time analysis, parts can be classified immediately, reducing the decision-making time and enhancing separation speed. DCCSs facilitate seamless coordination and communication among connected devices. With DCCSs, models are continuously refined, resulting in improved accuracy in identifying quality and defective parts. The distributed nature of DCCSs also contributes to fault tolerance. In the case of a failure in one computing system, processing can be seamlessly shifted to alternative resources, minimizing downtime and disruptions in the manufacturing process. In spite of upgrading the entire automation system, DCCSs can easily adapt and produce results according to upgraded requirements. In this context, DCCSs deliver scalability without compromising performance to adapt to changing production requirements.





**Figure 9.** Representation of separating high-quality and defective parts in a manufacturing industry automation.

#### 4.2. Transportation Systems

Transportation systems are organized networks and infrastructure that facilitate the movement of people or goods (through various modes of transportation such as roads, railways, airways, or waterways) between different locations [92–94]. DCCSs play a transformative role in modern transportation systems by integrating real-time data processing with the computation continuum. It processes data gathered through sensors equipped in vehicles and infrastructure to enhance traffic flow, reduce congestion, and ensure safe navigation [95,96]. Additionally, they support emergency response, infrastructure maintenance, and efficient mobility services.

**Use case:** Ensure safety alert when car drivers use their phones while driving.

A set of cameras were installed on the roadside to monitor traffic and vehicle conditions (for example, number plate, speed, following the rules or not) [97]. By the instant analysis (through AI inference) of these recordings, it can detect driver activity. Cameras would need to capture driver activity, radars would measure car speed, ground sensors would count cars, and a light system would allow recording in the dark. Additionally, the sensors must be connected to small processing units (such as IoT/edge processors) to compute data, and the cameras will be near AI inference boards. In addition to a large server on which to process video data captured by cameras, the application will have a large storage and processing capacity (i.e., cloud). Once the results are analyzed and any abnormal events are observed, they will be immediately reported to the driver and traffic inspectors. The detailed discussion of this problem and possible solution in DCCSs through SLOs is discussed by Dustdar et al. in [9].

Through the seamless integration of edge computing, cloud resources, and real-time analytics, DCCSs significantly enhance the efficiency of the above-described use case. DCCSs allows for the instant analysis of camera recordings, the detection of driver activity via AI inference, and processing of diverse sensor data at the source (e.g., the edge) with improved accuracy. It minimizes latency in detecting unsafe behaviors, instantly alerts drivers, and notifies traffic inspectors. DCCSs also achieve low latency due to their parallel processing capabilities. Furthermore, DCCS can adjust sensitivity levels dynamically based on lighting conditions and traffic flow, enabling the system to adapt to changing conditions. With DCCSs, we can accurately distribute data-intensive tasks and process and analyze

them in an efficient and timely manner. Ultimately, DCCSs increase road safety by rapidly and accurately detecting safety violations.

#### 4.3. Mobile Robots

Robots are autonomous machines equipped with sensors, actuators, and navigation systems to interact with environments. These robots have a broad range of applications across various industries, such as automated guided vehicles (AGVs), unmanned ground vehicles (UGVs), aerial drones, autonomous underwater vehicles (AUVs), search-and-rescue robots (SARs) and wearable mobile robots (WMRs). Mobile robots continue to evolve, benefiting from advances in artificial intelligence, machine learning, and connectivity. Hence, they will also be part of DCCSs once several research challenges are solved [98]. Their applications range from industrial automation to healthcare, agriculture, and beyond, and they enhance efficiency and reduce costs [99,100]. As a result of their ability to adapt to changing environments and obstacles through the use of sensors and sophisticated navigation algorithms, they become indispensable in the applications mentioned above.

**Use case:** search and rescue in large-scale disasters using mobile robots.

Climate changes in recent years have increased natural disasters and death rates. The first 72 h following a natural disaster or human disaster are crucial for locating and rescuing those affected [101]. It is unfortunate that these disasters do not just occur on land or underground [102], on water or underwater surfaces, or even in the air. In an emergency, rescue teams and first responders still suffer from situational awareness. But, in some cases, it is difficult to reach the rescue team in a timely manner, or sometimes it is not possible for a human to reach those locations. Identifying victims in such hostile environments is sometimes difficult for the rescue team [103]. This is despite researchers and industry investigating advanced technological solutions for SAR operations. In disaster-stricken areas, mobile robotic units, such as drones and underwater systems, serve as vital frontline assets [104]. As discussed earlier, these robots are equipped with sensors, cameras, and autonomous navigation capabilities, enabling them to detect victims and hazards in their surroundings. With edge AI, navigation and obstacle avoidance are enhanced due to rapid, informed decisions. Additionally, they provide reassurance to victims and relay critical information to the command center through vital communication links. By combining these technologies, natural disaster rescue operations are more efficient, safe, and effective, reducing fatalities and mitigating their effects [105]. With DCCSs, disaster response and rescue operations are enhanced by dynamically allocating computing resources, reducing latency, filtering and prioritizing data, ensuring redundancy and fault tolerance, maximizing communication, enabling scalable edge AI, conserving energy for mobile robots, and adapting in real time. Integrated data from IoT sensors and mobile robots streamlines decision making, ensures reliability, and leads to more efficient rescue efforts, saving lives, and mitigates natural disaster impacts.

#### 4.4. Smart Cities

A smart city uses a network of sensors and devices to collect real-time information about transportation, energy consumption, waste management, and public services [106]. Data from these sources can be analyzed and used for decision making as a means of increasing convenience, improving public services, and improving the quality of life for citizens [107–111]. Since DCCSs have inherent scalability, it can dynamically scale up or down in response to changes in the smart city ecosystem. DCCSs intelligently distribute processing tasks over this continuum to efficiently capture, analyze, and act on real-time data generated by IoT devices through interactions with urban infrastructure and citizens. The adaptive nature of DCCS also solves the unseen challenges of smart city applications.

**Use case:** Waste bin management in smart cities to reduce unnecessary collection trips during waste disposal.

In metropolitan cities across the world, municipal solid waste management has become a critical issue due to urbanization, growth, and lifestyle changes [112,113]. This issue

affects many aspects of life beyond developing nations, including health, the environment, recycling efforts, and multiple industries. It is possible to solve this problem using currently available technologies by adopting smart waste management strategies. Using this strategy, stakeholders will be notified of the type and quantity of waste generated and how smart waste collection will be implemented [114,115]. In this context, a cloud-based smart waste management mechanism was implemented by Aazam et al. in [116]. A sensor-equipped waste bin was integrated into the system, which sends information to the cloud about the level of waste in a bin. Hence, all waste bin information reaches the cloud, enabling stakeholders to access real-time waste status data and facilitating informed decision making, such as an optimized waste collection route. In addition to improving the fuel and time efficiency, intelligent route planning contributes to a more sustainable waste management system. There are several approaches available in the literature that use IoT and deep learning to improve smart waste management [117,118]. However, in the currently growing population and cities, more efficient and faster solutions are needed, and DCCSs can fill this gap.

The expansion of the city's proximity and population simultaneously increases the number of waste bins. This further generates more sensory data in greater proximity to the city. Since the cloud is located far away, the transmission delay and computational latency increase the delay in decision making. DCCSs integrate diverse technologies and resources to enable real-time data processing, analysis, and decision making, providing a great advantage for faster decision making. In addition to ensuring ongoing effectiveness, DCCSs' scalability guarantees their resilience in the face of urban growth without interrupting the current system. Since AI/ML achieves greater use in many applications in the current scenarios, their advantages are also grasped through DCCSs to predict before damage happens.

#### 4.5. Healthcare

Healthcare encompasses a variety of medical services, technologies, and systems designed to prevent, diagnose, treat, and manage diseases and health conditions. Several medical devices have evolved in recent years, from wearable sensors to high-end machines (placed in hospitals and healthcare centers) used to collect patient data and process it via smartphones (edge devices) or in the cloud [119,120]. Healthcare industries require accurate and quick analytical results from computing devices. It is sometimes necessary to analyze intensive tasks such as medical images (X-rays or CT scans) or genomic sequencing, but the result is expected to be available within a short period of time [121]. Sometimes, it is necessary to use AI or ML to predict the patient's condition, which needs more computational resources. In addition to optimizing healthcare efficiency, accessibility, and outcomes, DCCSs ensure seamless data flows from point-of-care devices to the edge-to-cloud continuum for further analytics.

**Use case:** monitoring intensive care unit (ICU) patient remotely.

Most hospitals in the world have limited caretakers, and it is very challenging to continuously monitor each patient [122–124]. In some cases, the negligence of the caretaker of ICU patients might risk their lives. Thus, a remote and automatic alert system might be useful to continuously monitor patients and record their health status for further analysis [125]. A number of wearable sensors and medical IoT devices gather real-time patient data, such as vital signs and health metrics (depending on the patient's diagnosis), and send these to a central server (private cloud) for processing [126]. In cloud-based computations, advanced algorithms and/or machine learning are used to analyze and compare the collected data with the normalized health metrics. It is also possible to visualize these metrics for quick assessment [127]. This approach allows healthcare professionals to monitor patients' conditions remotely, detect deviations, and make well-informed healthcare decisions. Some advanced analytics and pre-trained decision-making systems can also recommend prescriptions according to their assessment. As a result of this system, patients

receive better care by facilitating continuous monitoring and early intervention based on real-time insights.

As DCCSs allow computation across the continuum, optimized processing, and resource allocation between point-of-care devices and edge-to-cloud, DCCSs can handle large computations in a limited time. The scalability of DCCSs allows the system to support an increasing number of patients and data streams without affecting performance and existing patients. Using DCCSs, healthcare providers and patients can communicate in real-time, and critical changes in health can be detected and notified immediately. Furthermore, DCCSs can provide proactive patient management by identifying potential trends and risk factors based on the analysis of historical patient data (i.e., predictive analysis).

## 5. Scope for Further Research

Exploring DCCSs opens new possibilities, inspiring further investigation into its wide range of dimensions. This section aims to uncover new insights into DCCSs and provide possible opportunities for further research on the challenges discussed in previous sections.

### 5.1. Learning Models for DCCS

The rapid evolution of learning models has led to advancements across diverse fields and enabled innovative applications in a wide range of areas. In addition, DCCSs generate a large amount of data from its devices, which need to be further processed due to a variety of factors. In this aspect, learning models are more appropriate for real-time data analysis and decision making. There are several algorithms in the literature [128–130], and novel approaches are rapidly evolving. For example, deep learning models are being advanced as generative adversarial networks (GANs) that drive breakthroughs in many fields [131]. An ensemble learning model is developed by combining two or more learning algorithms to achieve a benefit in an application [132]. Furthermore, several experimental learning models acquire knowledge by interacting with environments, like human learning. However, these algorithms need a structured and huge amount of data and resources to achieve greater efficiency. Models can learn from small datasets or transfer knowledge between tasks using transfer and few-shot learning strategies. This strategy is more efficient when the computing is limited to edge devices.

DCCS challenges and features are distinct in terms of heterogeneity in devices, complexity of data generation, and communications. Thus, developing new learning strategies will help DCCSs maximize its potential while reducing its inherent challenges. When developing these algorithms, it is necessary to consider various constraints, including energy efficiency, resource and data limitations, available data sizes, real-time decision making with local and global maxima, cost-effective, easy to adapt, and scalable. It is also necessary to estimate the real-time (or run time) accuracy of these newly designed learning algorithms. Adopting incremental learning features can improve the performance of learned models with limited data availability as well as learned models from historical data [133,134]. Domain generalization in DCCSs will allow models to provide accurate predictions or decisions across various devices, locations, and conditions, which will enhance its efficiency and effectiveness [135]. Utilizing these models will enable DCCSs to maximize its potential while meeting the challenges posed by a dynamic environment and increased user experience.

### 5.2. Need for Intelligent Protocols

Currently, most data protocols used in the cloud, edge/fog computing, IoT, or DCCSs use deterministic protocols where the rules are predefined. These rules pose several challenges, including in terms of scalability, reliability, and interoperability, and are highly complex to adapt to the dynamic conditions of systems. With the increasing number of computing devices in DCCSs, data communication through data protocols is an emerging research area. In addition, advances in machine learning and AI are ever-growing, and integrating DCCSs and AI opens up an exciting land for research and development, i.e., in-

telligent data protocols. This field not only extends the existing data protocols (MQTT, AMQP, DDS, or CoAP) with intelligence but simultaneously opens the scope for developing new protocols by adding several features such as adaptability, scalability, dynamic decision making according to real-time system conditions and/or energy efficiency [136]. Adding intelligence to data protocols not only makes them smart but also mitigates several challenges, including static message expiry or priority, message filtering, and congestion issues. Furthermore, intelligent protocols are built according to the available resources, such as the queue size, the current status of the receivers, and the amount of energy available. However, it is necessary to take into account the complexity of AI/ML used in constrained devices because of the data requirements to train and resources needed to store and compute training models [137]. This challenge further opens research focus on developing lightweight learning models that can run on tiny or constrained devices with limited data availability, resulting in high real-time accuracy [138].

### 5.3. Use of Causality

Generally, causation refers to the relationship between a cause and effect, where a cause results in an effect [139]. In DCCSs, it plays a crucial role in understanding how actions, events, and decisions influence outcomes. This allows us to understand the behavior of a system, predict the behavior of the system, filter observations, and ensure consistency. Through this understanding and a priori analysis, it is easy to minimize uncertainty effects. Using causality in DCCSs offers several benefits, such as maintaining system integrity, identifying the order of execution during parallel computation across multiple computing devices, i.e., intuitive ordering, predicting the availability of resources for future usages so that latency will be avoided, and fault or failure predictions. For example, Chen et al. in [140] used causality to model fault propagation paths and infer the root causes (real culprits responsible for performance issues) of performance problems in distributed systems. In the literature, there are several algorithms for identifying the most appropriate causal relationship with limited analysis, such as graph knowledge representation (GKR) through representation learning [11].

### 5.4. Continuous Diagnostics and Mitigation

In a zero-trust architecture (ZTA), continuous diagnostics and mitigation (CDM) continually assesses the security posture of the network (available online: [https://csrc.nist.gov/glossary/term/continuous\\_diagnostics\\_and\\_mitigation](https://csrc.nist.gov/glossary/term/continuous_diagnostics_and_mitigation), accessed on 3 September 2023). It helps detect anomalous behavior or unauthorized access attempts, enabling organizations to respond promptly. Moreover, enabling such a CDM mechanism in DCCSs will enhance the governance of computing entities. Through an efficient CDM program, it is easy to minimize downtime and ensure sustainability. Due to the vast amount of features and challenges, CDM programs can also be embedded with multiple features such as autonomous configuration management, monitoring the up and downtime of each device, continuous health monitoring of devices, and its effect on environmental factors. The CDM program also serves to check the feasibility of a newly added device in the system. The design of the CDM program opens a wider research scope due to challenges associated with device data. These challenges include high nonlinearity, sparsity, difficulty in breaking structures, and complexity in extracting explanatory factors across systems' data [42]. The causal analysis discussed in the previous section helps to monitor and predict device conditions in an efficient manner [141]. With the integration of zero-touch provisioning into the CDM program, DCCS can manage configurations more efficiently and remotely [142].

### 5.5. Data Fragmentation and Clustered Edge Intelligence

Data fragmentation refers to the splitting of a large amount of data into a specific number of small partitions. Fragmented portions can be efficiently transmitted through limited bandwidth and quickly computed, separately, and in parallel [143]. DCCS data

might be generated at various locations of the continuum and could be difficult to analyze within one edge device. In such cases, data fragmentation optimizes the use of resources (memory, CPU, and bandwidth), reduces the processing times (due to processing a chunk of data), enhances scalability, and supports parallel processing. Since data fragments are processed across a continuum, fault tolerance is also supported. This means a device failure does not affect the entire processing and makes it easy to distribute the load from one failed device to another. Additionally, data fragmentation presents limitations in DCCSs, such as data dependencies, communication overhead, and load imbalance. Since DCCSs are connected to a wider area and with a large number of computing nodes, scaling the fragments for computation becomes a challenging task [144]. As an alternative to overcome these challenges, clustered edge intelligence (CEI) groups edge devices into clusters wisely based on fragmentation and resource needs [145]. Furthermore, CEI should be based on performance metrics such as latency, bandwidth, CPU, and/or memory availability. Since the CEI is still incompletely evolved, there is huge scope for further research.

### *5.6. Energy-Efficiency and Sustainability*

In DCCSs, each device uses power to enable data collection, storage, communication, and processing. The limited energy capacity of these devices is constrained by factors such as battery life or power availability in remote locations [2]. In addition, increasing the number of sensor devices, IoT, edge or cloud, and associated devices contributes to greenhouse gas emissions. A BBC study says the amount of emissions we produce through electronic gadgets, the Internet, and their supporting systems is similar to that produced by the airline industry worldwide (Available online: <https://www.bbc.com/future/article/20200305-why-your-internet-habits-are-not-as-clean-as-you-think>, accessed on 3 September 2023). IoT/edge devices are the same. In order to decarbonize the atmosphere, we need energy-efficient methods that prioritize resource conservation and environmental governance. Hence, energy efficiency for DCCS is one of the primary challenges in the current energy and planetary conditions, making it necessary to efficiently manage energy to guarantee sustainable and uninterrupted operation while minimizing the carbon footprint.

### *5.7. Controlling Data Gravity and Data Friction*

Data gravity describes the tendency of data to attract additional data. This is based on the idea that the more data and applications are stored at a particular location, the more attractive it becomes for other data to be stored there. Data friction is resistance that impedes data transfer. Due to the dynamic allocation performed by DCCS, there is a high probability of data gravity and friction occurring on specific devices in the network. Data gravity and friction pose challenges to efficient resource utilization and performance in the computing continuum [146]. These challenges can be controlled through intelligent data placement, replication, caching, and efficient data movement strategies, but more research is needed to confirm this.

## **6. Conclusions**

The computing paradigm has evolved over decades from a computer with room-sized resources to tiny computing devices, enabling seamless interaction between devices and humans. This evolution trajectory continues to accelerate, promising unprecedented capabilities that will reshape industries, science, and society at large. This paper provides an overview of the evolution of computing paradigms from the 1960s to the present day in an effort to illustrate the change. We ranged from large-scale mainframe computers to current distributed computing continuum systems (DCCSs). We analyzed the architectures, components, benefits and limitations of each computing model. We discussed the major advantages of DCCSs suitable for current application scenarios and computational demands. We examined appropriate use cases with respect to traditional technologies and the advantages that come with DCCSs. We provided an illustrative example use case for each application to understand DCCS use and its advantages. As DCCSs are a growing

field of research, a wide range of open research challenges and opportunities are available for further research. We discussed possible open research challenges and suggest suitable directions for further research.

**Author Contributions:** Conceptualization, P.K.D.; Data curation, P.K.D.; Formal analysis, P.K.D. and I.M.; Funding acquisition, P.K.D. and S.D.; Investigation, P.K.D.; Methodology, P.K.D.; Project administration, S.D.; Resources, P.K.D. and S.D.; Software, P.K.D.; Supervision, S.D.; Validation, P.K.D. and I.M.; Visualization, P.K.D. and I.M.; Writing—original draft, P.K.D., I.M. and V.C.P.; Writing—review editing, V.C.P., B.S. and S.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** Research has partially received funding from grant agreement Nos. 101070186 (TEADAL) and 101079214 (AloTwin) by EU Horizon.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

### Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AGVs	Automated Guided Vehicles
AMQP	Advanced Message Queuing Protocol
API	Application Program Interface
AR	Augmented Reality
ATM	Automated Teller Machine
AUVs	Autonomous Underwater Vehicles
CAGR	Compound Annual Growth Rate
CDM	Continuous Diagnostics and Mitigation
CDN	Content Delivery Network
CEI	Clustered Edge Intelligence
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CT	Computerized Tomography
DCCS	Distributed Computing Continuum System
DDS	Data Distribution Service
DRL	Deep Reinforcement Learning
FEP	Free Energy Principle
FDG	Federated Domain Generalization
FL	Federated Learning
GAN	Generative Adversarial Networks
GKR	Graphical Knowledge Representation
ICU	Intensive Care Unit
IoT	Internet of Things
LLM	Large Language Models
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
SAR	Search-and-Rescue Robots
SLO	Service Level Objective
UGVs	Unmanned Ground Vehicles
VR	Virtual Reality
WAN	Wide-Area Networks
WMR	Wearable Mobile Robots
ZTA	Zero-Trust Architecture
ZTP	Zero-Touch Provisioning

## References

1. De Donno, M.; Tange, K.; Dragoni, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *IEEE Access* **2019**, *7*, 150936–150948. [[CrossRef](#)]
2. Yuan, J.; Xiao, H.; Shen, Z.; Zhang, T.; Jin, J. ELECT: Energy-efficient intelligent edge–cloud collaboration for remote IoT services. *Future Gener. Comput. Syst.* **2023**, *147*, 179–194. [[CrossRef](#)]
3. Alsamhi, S.H.; Shvetsov, A.V.; Kumar, S.; Hassan, J.; Alhartomi, M.A.; Shvetsova, S.V.; Sahal, R.; Hawbani, A. Computing in the sky: A survey on intelligent ubiquitous computing for uav-assisted 6g networks and industry 4.0/5.0. *Drones* **2022**, *6*, 177. [[CrossRef](#)]
4. Ometov, A.; Molua, O.L.; Komarov, M.; Nurmi, J. A survey of security in cloud, edge, and fog computing. *Sensors* **2022**, *22*, 927. [[CrossRef](#)] [[PubMed](#)]
5. Ren, J.; Zhang, D.; He, S.; Zhang, Y.; Li, T. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Comput. Surv.* **2019**, *52*, 1–36. [[CrossRef](#)]
6. Zhang, Y.; Ren, J.; Liu, J.; Xu, C.; Guo, H.; Liu, Y. A survey on emerging computing paradigms for big data. *Chin. J. Electron.* **2017**, *26*, 1–12. [[CrossRef](#)]
7. Angel, N.A.; Ravindran, D.; Vincent, P.D.R.; Srinivasan, K.; Hu, Y.C. Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies. *Sensors* **2021**, *22*, 196. [[CrossRef](#)] [[PubMed](#)]
8. Lyytinen, K.; Yoo, Y. Ubiquitous computing. *Commun. ACM* **2002**, *45*, 63–96.
9. Dustdar, S.; Pujol, V.C.; Donta, P.K. On distributed computing continuum systems. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 4092–4105. [[CrossRef](#)]
10. Casamayor Pujol, V.; Morichetta, A.; Murturi, I.; Kumar Donta, P.; Dustdar, S. Fundamental research challenges for distributed computing continuum systems. *Information* **2023**, *14*, 198. [[CrossRef](#)]
11. Donta, P.K.; Dustdar, S. The promising role of representation learning for distributed computing continuum systems. In Proceedings of the 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), Newark, CA, USA, 15–18 August 2022 ; pp. 126–132. [[CrossRef](#)]
12. Orive, A.; Agirre, A.; Truong, H.L.; Sarachaga, I.; Marcos, M. Quality of Service Aware Orchestration for Cloud–Edge Continuum Applications. *Sensors* **2022**, *22*, 1755. [[CrossRef](#)] [[PubMed](#)]
13. Filho, C.P.; Marques, E., Jr.; Chang, V.; Dos Santos, L.; Bernardini, F.; Pires, P.F.; Ochi, L.; Delicato, F.C. A systematic literature review on distributed machine learning in edge computing. *Sensors* **2022**, *22*, 2665. [[CrossRef](#)]
14. Iansiti, M.; Clark, K.B. Integration and dynamic capability: Evidence from product development in automobiles and mainframe computers. *Ind. Corp. Chang.* **1994**, *3*, 557–605. [[CrossRef](#)]
15. Greenstein, S.M. Lock-in and the costs of switching mainframe computer vendors: What do buyers see? *Ind. Corp. Chang.* **1997**, *6*, 247–273. [[CrossRef](#)]
16. Schwiegelshohn, U.; Badia, R.M.; Bubak, M.; Danelutto, M.; Dustdar, S.; Gagliardi, F.; Geiger, A.; Hluchy, L.; Kranzlmüller, D.; Laure, E.; et al. Perspectives on grid computing. *Future Gener. Comput. Syst.* **2010**, *26*, 1104–1115. [[CrossRef](#)]
17. Casanova, H. Distributed computing research issues in grid computing. *ACM SIGAct News* **2002**, *33*, 50–70. [[CrossRef](#)]
18. Yu, J.; Buyya, R. A taxonomy of workflow management systems for grid computing. *J. Grid Comput.* **2005**, *3*, 171–200. [[CrossRef](#)]
19. Yeo, C.S.; Buyya, R. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Softw. Pract. Exp.* **2006**, *36*, 1381–1419. [[CrossRef](#)]
20. Baker, M.; Buyya, R. Cluster computing: The commodity supercomputer. *Softw. Pract. Exp.* **1999**, *29*, 551–576. [[CrossRef](#)]
21. Barak, A.; La’adan, O. The MOSIX multicomputer operating system for high performance cluster computing. *Future Gener. Comput. Syst.* **1998**, *13*, 361–372. [[CrossRef](#)]
22. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* **2016**, *3*, 854–864. [[CrossRef](#)]
23. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 169–186. [[CrossRef](#)]
24. Buyya, R.; Srirama, S.N. *Fog and Edge Computing: Principles and Paradigms*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
25. Yi, S.; Hao, Z.; Qin, Z.; Li, Q. Fog computing: Platform and applications. In Proceedings of the 2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb), Washington, DC, USA, 12–13 November 2015; pp. 73–78. [[CrossRef](#)]
26. Avasalcai, C.; Murturi, I.; Dustdar, S. Edge and fog: A survey, use cases, and future challenges. In *Fog Computing: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2020; pp. 43–65. [[CrossRef](#)]
27. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource scheduling in edge computing: A survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2131–2165. [[CrossRef](#)]
28. Martin Wisniewski, L.; Bec, J.M.; Boguszewski, G.; Gamatié, A. Hardware Solutions for Low-Power Smart Edge Computing. *J. Low Power Electron. Appl.* **2022**, *12*, 61. [[CrossRef](#)]
29. Sulieman, N.A.; Ricciardi Celsi, L.; Li, W.; Zomaya, A.; Villari, M. Edge-oriented computing: A survey on research and use cases. *Energies* **2022**, *15*, 452. [[CrossRef](#)]
30. Shi, W.; Dustdar, S. The promise of edge computing. *Computer* **2016**, *49*, 78–81. [[CrossRef](#)]
31. Li, C.; Xue, Y.; Wang, J.; Zhang, W.; Li, T. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Comput. Surv.* **2018**, *51*, 1–34. [[CrossRef](#)]



32. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J.P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330. [[CrossRef](#)]
33. Dustdar, S.; Murturi, I. Towards distributed edge-based systems. In Proceedings of the 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI), Atlanta, GA, USA, 28–31 October 2020; pp. 1–9. [[CrossRef](#)]
34. Taibi, D.; Spillner, J.; Wawruch, K. Serverless computing—where are we now, and where are we heading? *IEEE Softw.* **2020**, *38*, 25–31. [[CrossRef](#)]
35. Shafiei, H.; Khonsari, A.; Mousavi, P. Serverless computing: A survey of opportunities, challenges, and applications. *ACM Comput. Surv.* **2022**, *54*, 1–32. [[CrossRef](#)]
36. Wen, J.; Chen, Z.; Jin, X.; Liu, X. Rise of the planet of serverless computing: A systematic review. *ACM Trans. Softw. Eng. Methodol.* **2023**, *32*, 1–61. [[CrossRef](#)]
37. Poojara, S.R.; Dehury, C.K.; Jakovits, P.; Srirama, S.N. Serverless data pipeline approaches for IoT data in fog and cloud computing. *Future Gener. Comput. Syst.* **2022**, *130*, 91–105. [[CrossRef](#)]
38. Poojara, S.; Dehury, C.K.; Jakovits, P.; Srirama, S.N. Serverless Data Pipelines for IoT Data Analytics: A Cloud Vendors Perspective and Solutions. In *Predictive Analytics in Cloud, Fog, and Edge Computing: Perspectives and Practices of Blockchain, IoT, and 5G*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 107–132. [[CrossRef](#)]
39. Li, Z.; Guo, L.; Cheng, J.; Chen, Q.; He, B.; Guo, M. The serverless computing survey: A technical primer for design architecture. *ACM Comput. Surv.* **2022**, *54*, 1–34. [[CrossRef](#)]
40. Naranjo, D.M.; Risco, S.; de Alfonso, C.; Pérez, A.; Blanquer, I.; Moltó, G. Accelerated serverless computing based on GPU virtualization. *J. Parallel Distrib. Comput.* **2020**, *139*, 32–42. [[CrossRef](#)]
41. Castro, P.; Ishakian, V.; Muthusamy, V.; Slominski, A. The rise of serverless computing. *Commun. ACM* **2019**, *62*, 44–54. [[CrossRef](#)]
42. Donta, P.K.; Sedlak, B.; Casamayor Pujol, V.; Dustdar, S. Governance and sustainability of distributed continuum systems: A big data approach. *J. Big Data* **2023**, *10*, 53. [[CrossRef](#)]
43. Beckman, P.; Dongarra, J.; Ferrier, N.; Fox, G.; Moore, T.; Reed, D.; Beck, M. Harnessing the computing continuum for programming our world. In *Fog Computing: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2020; pp. 215–230. [[CrossRef](#)]
44. Ketu, S.; Mishra, P.K. Cloud, fog and mist computing in IoT: An indication of emerging opportunities. *IETE Tech. Rev.* **2022**, *39*, 713–724. [[CrossRef](#)]
45. Masip-Bruin, X.; Marín-Tordera, E.; Sánchez-López, S.; Garcia, J.; Jukan, A.; Juan Ferrer, A.; Queralt, A.; Salis, A.; Bartoli, A.; Cankar, M.; et al. Managing the cloud continuum: Lessons learnt from a real fog-to-cloud deployment. *Sensors* **2021**, *21*, 2974. [[CrossRef](#)]
46. Pujol, V.C.; Donta, P.K.; Morichetta, A.; Murturi, I.; Dustdar, S. Edge Intelligence—Research Opportunities for Distributed Computing Continuum Systems. *IEEE Internet Comput.* **2023**, *27*, 53–74. [[CrossRef](#)]
47. Huang, J.; Li, R.; An, J.; Ntalasha, D.; Yang, F.; Li, K. Energy-efficient resource utilization for heterogeneous embedded computing systems. *IEEE Trans. Comput.* **2017**, *66*, 1518–1531. [[CrossRef](#)]
48. Leveson, N.G. Software safety in embedded computer systems. *Commun. ACM* **1991**, *34*, 34–46. [[CrossRef](#)]
49. Das, A.; Kumar, A.; Veeravalli, B. Energy-aware task mapping and scheduling for reliable embedded computing systems. *ACM Trans. Embed. Comput. Syst.* **2014**, *13*, 1–27. [[CrossRef](#)]
50. Rodrigues, J.J.; Segundo, D.B.D.R.; Junqueira, H.A.; Sabino, M.H.; Prince, R.M.; Al-Muhtadi, J.; De Albuquerque, V.H.C. Enabling technologies for the internet of health things. *IEEE Access* **2018**, *6*, 13129–13141. [[CrossRef](#)]
51. Nahavandi, D.; Alizadehsani, R.; Khosravi, A.; Acharya, U.R. Application of artificial intelligence in wearable devices: Opportunities and challenges. *Comput. Methods Programs Biomed.* **2022**, *213*, 106541. [[CrossRef](#)] [[PubMed](#)]
52. Scilingo, E.P.; Valenza, G. Recent advances on wearable electronics and embedded computing systems for biomedical applications. *Electronics* **2017**, *6*, 12. [[CrossRef](#)]
53. Iqbal, S.M.; Mahgoub, I.; Du, E.; Leavitt, M.A.; Asghar, W. Advances in healthcare wearable devices. *NPJ Flex. Electron.* **2021**, *5*, 9. [[CrossRef](#)]
54. Portilla, L.; Loganathan, K.; Faber, H.; Eid, A.; Hester, J.G.; Tentzeris, M.M.; Fattori, M.; Cantatore, E.; Jiang, C.; Nathan, A.; et al. Wirelessly powered large-area electronics for the Internet of Things. *Nat. Electron.* **2023**, *6*, 10–17. [[CrossRef](#)]
55. Ali, I.; Ahmedy, I.; Gani, A.; Munir, M.U.; Anisi, M.H. Data collection in studies on Internet of things (IoT), wireless sensor networks (WSNs), and sensor cloud (SC): Similarities and differences. *IEEE Access* **2022**, *10*, 33909–33931. [[CrossRef](#)]
56. Hussain, S.; Ullah, S.S.; Ali, I. An efficient content source verification scheme for multi-receiver in NDN-based Internet of Things. *Clust. Comput.* **2022**, *25*, 1749–1764. [[CrossRef](#)]
57. Bayılmış, C.; Ebleme, M.A.; Çavuşoğlu, Ü.; Küçük, K.; Sevin, A. A survey on communication protocols and performance evaluations for Internet of Things. *Digit. Commun. Netw.* **2022**, *8*, 1094–1104. [[CrossRef](#)]
58. Dizdarević, J.; Carpio, F.; Jukan, A.; Masip-Bruin, X. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comput. Surv.* **2019**, *51*, 1–29. [[CrossRef](#)]
59. Khanna, A.; Kaur, S. Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. *Comput. Electron. Agric.* **2019**, *157*, 218–231. [[CrossRef](#)]
60. Alavikia, Z.; Shabro, M. A comprehensive layered approach for implementing internet of things-enabled smart grid: A survey. *Digit. Commun. Netw.* **2022**, *8*, 388–410. [[CrossRef](#)]

61. Goudarzi, A.; Ghayoor, F.; Waseem, M.; Fahad, S.; Traore, I. A Survey on IoT-Enabled Smart Grids: Emerging, Applications, Challenges, and Outlook. *Energies* **2022**, *15*, 6984. [[CrossRef](#)]
62. Aazhang, B.; Ahokangas, P.; Alves, H.; Alouini, M.S.; Beek, J.; Benn, H.; Bennis, M.; Belfiore, J.; Strinati, E.; Chen, F.; et al. Key drivers and research challenges for 6G ubiquitous wireless intelligence (white paper). *Univ. Oulu* **2019**, 1–36.
63. Lovén, L.; Leppänen, T.; Peltonen, E.; Partala, J.; Harjula, E.; Porambage, P.; Ylianttila, M.; Riekkilä, J. EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks. In Proceedings of the 6G Wireless Summit, Levi, Finland, 24–26 March 2019.
64. Peltonen, E.; Bennis, M.; Capobianco, M.; Debbah, M.; Ding, A.; Gil-Castiñeira, F.; Jurmu, M.; Karvonen, T.; Kelanti, M.; Kliks, A.; et al. 6G white paper on edge intelligence. *arXiv* **2020**, arXiv:2004.14850. <https://doi.org/10.48550/arXiv.2004.14850>.
65. López, O.A.; Rosabal, O.M.; Ruiz-Guirola, D.; Raghuvanshi, P.; Mikhaylov, K.; Lovén, L.; Iyer, S. Energy-Sustainable IoT Connectivity: Vision, Technological Enablers, Challenges, and Future Directions. *arXiv* **2023**, arXiv:2306.02444. <https://doi.org/10.48550/arXiv.2306.02444>.
66. Rashid, A.T.; Elder, L. Mobile phones and development: An analysis of IDRC-supported projects. *Electron. J. Inf. Syst. Dev. Ctries.* **2009**, *36*, 1–16. [[CrossRef](#)]
67. Duncombe, R. Researching impact of mobile phones for development: Concepts, methods and lessons for practice. *Inf. Technol. Dev.* **2011**, *17*, 268–288. [[CrossRef](#)]
68. Hennessy, J.L.; Patterson, D.A. *Computer Architecture: A Quantitative Approach*; Elsevier: Amsterdam, The Netherlands, 2011.
69. Wang, X.; Li, J.; Ning, Z.; Song, Q.; Guo, L.; Guo, S.; Obaidat, M.S. Wireless powered mobile edge computing networks: A survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [[CrossRef](#)]
70. Yang, L.; Jiang, H.; Shi, J.; Xue, X.; Ren, P.; Feng, Y.; Chen, J. Achieving Cooperative Mobile-Edge Computing Using Helper Scheduling. *IEEE Trans. Commun.* **2023**, *7*, 3419–3436. [[CrossRef](#)]
71. Yadav, A.M.; Sharma, S. Cooperative task scheduling secured with blockchain in sustainable mobile edge computing. *Sustain. Comput. Inform. Syst.* **2023**, *37*, 100843. [[CrossRef](#)]
72. Feng, C.; Han, P.; Zhang, X.; Yang, B.; Liu, Y.; Guo, L. Computation offloading in mobile edge computing networks: A survey. *J. Netw. Comput. Appl.* **2022**, *202*, 103366. [[CrossRef](#)]
73. Peng, H.; Davidson, S.; Shi, R.; Song, S.L.; Taylor, M. Chiptlet Cloud: Building AI Supercomputers for Serving Large Generative Language Models. *arXiv* **2023**, arXiv:2307.02666. <https://doi.org/10.48550/arXiv.2307.02666>.
74. Xu, M.; Du, H.; Niyato, D.; Kang, J.; Xiong, Z.; Mao, S.; Han, Z.; Jamalipour, A.; Kim, D.I.; Leung, V.; et al. Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services. *arXiv* **2023**, arXiv:2303.16129. <https://doi.org/10.48550/arXiv.2303.16129>.
75. Dhar, S.; Guo, J.; Liu, J.; Tripathi, S.; Kurup, U.; Shah, M. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Trans. Internet Things* **2021**, *2*, 1–49. [[CrossRef](#)]
76. Saravanan, K.; Kouzani, A.Z. Advancements in On-Device Deep Neural Networks. *Information* **2023**, *14*, 470. [[CrossRef](#)]
77. Xu, M.; Song, C.; Tian, Y.; Agrawal, N.; Granqvist, F.; van Dalen, R.; Zhang, X.; Argueta, A.; Han, S.; Deng, Y.; et al. Training Large-Vocabulary Neural Language Models by Private Federated Learning for Resource-Constrained Devices. In Proceedings of the ICASSP 2023–2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [[CrossRef](#)]
78. Wang, B.; Zhang, Y.J.; Cao, Y.; Li, B.; McMahan, H.B.; Oh, S.; Xu, Z.; Zaheer, M. Can Public Large Language Models Help Private Cross-device Federated Learning? *arXiv* **2023**, arXiv:2305.12132. <https://doi.org/10.48550/arXiv.2305.12132>.
79. Park, H.; Kim, S. Overviewing AI-Dedicated Hardware for On-Device AI in Smartphones. In *Artificial Intelligence and Hardware Accelerators*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 127–150. [[CrossRef](#)]
80. Yi, R.; Guo, L.; Wei, S.; Zhou, A.; Wang, S.; Xu, M. EdgeMoE: Fast On-Device Inference of MoE-based Large Language Models. *arXiv* **2023**, arXiv:2308.14352. <https://doi.org/10.48550/arXiv.2308.14352>.
81. Adepu, S.; Adler, R.F. A comparison of performance and preference on mobile devices vs. desktop computers. In Proceedings of the 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 20–22 October 2016; pp. 1–7. [[CrossRef](#)]
82. Straker, L.; Jones, K.J.; Miller, J. A comparison of the postures assumed when using laptop computers and desktop computers. *Appl. Ergon.* **1997**, *28*, 263–268. [[CrossRef](#)]
83. Oyanagi, Y. Future of supercomputing. *J. Comput. Appl. Math.* **2002**, *149*, 147–153. [[CrossRef](#)]
84. Suarez, E.; Eicker, N.; Lippert, T. Modular supercomputing architecture: From idea to production. In *Contemporary High Performance Computing*; CRC Press: Boca Raton, FL, USA, 2019; pp. 223–255. [[CrossRef](#)]
85. Oral, S.; Vazhkudai, S.S.; Wang, F.; Zimmer, C.; Brumgard, C.; Hanley, J.; Markomanolis, G.; Miller, R.; Leverman, D.; Atchley, S.; et al. End-to-end i/o portfolio for the summit supercomputing ecosystem. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–19 November 2019; pp. 1–14. [[CrossRef](#)]
86. Wang, T.; Zhou, J.; Liu, A.; Bhuiyan, M.Z.A.; Wang, G.; Jia, W. Fog-based computing and storage offloading for data synchronization in IoT. *IEEE Internet Things J.* **2018**, *6*, 4272–4282. [[CrossRef](#)]
87. Rosendo, D.; Costan, A.; Valduriez, P.; Antoniu, G. Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review. *J. Parallel Distrib. Comput.* **2022**, *166*, 71–94. [[CrossRef](#)]

88. Waheed, A.; Shah, M.A.; Mohsin, S.M.; Khan, A.; Maple, C.; Aslam, S.; Shamshirband, S. A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks. *IEEE Access* **2022**, *10*, 3580–3600. [[CrossRef](#)]
89. Cheng, Y.L.; Lim, M.H.; Hui, K.H. Impact of internet of things paradigm towards energy consumption prediction: A systematic literature review. *Sustain. Cities Soc.* **2022**, *78*, 103624. [[CrossRef](#)]
90. Hashim Mohammed, B.; Sallehuddin, H.; Safie, N.; Husairi, A.; Abu Bakar, N.A.; Yahya, F.; Ali, I.; AbdelGhany Mohamed, S. Building Information Modeling and Internet of Things Integration in the Construction Industry: A Scoping Study. *Adv. Civ. Eng.* **2022**, *2022*, 7886497. [[CrossRef](#)]
91. Alekseeva, D.; Ometov, A.; Arponen, O.; Lohan, E.S. The future of computing paradigms for medical and emergency applications. *Comput. Sci. Rev.* **2022**, *45*, 100494. [[CrossRef](#)]
92. Ravi, B.; Varghese, B.; Murturi, I.; Donta, P.K.; Dustdar, S.; Dehury, C.K.; Srirama, S.N. Stochastic Modeling for Intelligent Software-Defined Vehicular Networks: A Survey. *Computers* **2023**, *12*, 162. [[CrossRef](#)]
93. Zhu, F.; Lv, Y.; Chen, Y.; Wang, X.; Xiong, G.; Wang, F.Y. Parallel transportation systems: Toward IoT-enabled smart urban traffic control and management. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 4063–4071. [[CrossRef](#)]
94. Chen, C.; Liu, B.; Wan, S.; Qiao, P.; Pei, Q. An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 1840–1852. [[CrossRef](#)]
95. Deveci, M.; Gokasar, I.; Pamucar, D.; Zaidan, A.A.; Wen, X.; Gupta, B.B. Evaluation of Cooperative Intelligent Transportation System scenarios for resilience in transportation using type-2 neutrosophic fuzzy VIKOR. *Transp. Res. Part A Policy Pract.* **2023**, *172*, 103666. [[CrossRef](#)]
96. Boukerche, A.; Wang, J. Machine learning-based traffic prediction models for intelligent transportation systems. *Comput. Netw.* **2020**, *181*, 107530. [[CrossRef](#)]
97. Shahverdy, M.; Fathy, M.; Berangi, R.; Sabokrou, M. Driver behavior detection and classification using deep convolutional neural networks. *Expert Syst. Appl.* **2020**, *149*, 113240. [[CrossRef](#)]
98. Pujol, V.C.; Dustdar, S. Fog Robotics—Understanding the Research Challenges. *IEEE Internet Comput.* **2021**, *25*, 10–17. [[CrossRef](#)]
99. Wang, G.; Wang, W.; Ding, P.; Liu, Y.; Wang, H.; Fan, Z.; Bai, H.; Hongbiao, Z.; Du, Z. Development of a search and rescue robot system for the underground building environment. *J. Field Robot.* **2023**, *40*, 655–683. [[CrossRef](#)]
100. Kim, T.H.; Bae, S.H.; Han, C.H.; Hahn, B. The Design of a Low-Cost Sensing and Control Architecture for a Search and Rescue Assistant Robot. *Machines* **2023**, *11*, 329. [[CrossRef](#)]
101. Militano, L.; Arteaga, A.; Toffetti, G.; Mitton, N. The Cloud-to-Edge-to-IoT Continuum as an Enabler for Search and Rescue Operations. *Future Internet* **2023**, *15*, 55. [[CrossRef](#)]
102. Jácome, M.Y.; Alvear Villaroel, F.; Figueroa Olmedo, J. Ground Robot for Search and Rescue Management. In Proceedings of the International Conference on Applied Technologies, Virtual, 23–25 November 2022; pp. 399–411. [[CrossRef](#)]
103. Feng, S.; Shi, H.; Huang, L.; Shen, S.; Yu, S.; Peng, H.; Wu, C. Unknown hostile environment-oriented autonomous WSN deployment using a mobile robot. *J. Netw. Comput. Appl.* **2021**, *182*, 103053. [[CrossRef](#)]
104. Mouradian, C.; Yangui, S.; Glietho, R.H. Robots as-a-service in cloud computing: Search and rescue in large-scale disasters case study. In Proceedings of the 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 12–15 January 2018; pp. 1–7. [[CrossRef](#)]
105. Lyu, M.; Zhao, Y.; Huang, C.; Huang, H. Unmanned Aerial Vehicles for Search and Rescue: A Survey. *Remote Sens.* **2023**, *15*, 3266. [[CrossRef](#)]
106. Badidi, E.; Mahrez, Z.; Sabir, E. Fog computing for smart cities’ big data management and analytics: A review. *Future Internet* **2020**, *12*, 190. [[CrossRef](#)]
107. Liu, Y.; Yang, C.; Jiang, L.; Xie, S.; Zhang, Y. Intelligent edge computing for IoT-based energy management in smart cities. *IEEE Netw.* **2019**, *33*, 111–117. [[CrossRef](#)]
108. Zhang, C. Design and application of fog computing and Internet of Things service platform for smart city. *Future Gener. Comput. Syst.* **2020**, *112*, 630–640. [[CrossRef](#)]
109. Lv, Z.; Chen, D.; Lou, R.; Wang, Q. Intelligent edge computing based on machine learning for smart city. *Future Gener. Comput. Syst.* **2021**, *115*, 90–99. [[CrossRef](#)]
110. Kashef, M.; Visvizi, A.; Troisi, O. Smart city as a smart service system: Human–computer interaction and smart city surveillance systems. *Comput. Hum. Behav.* **2021**, *124*, 106923. [[CrossRef](#)]
111. Silva, B.N.; Khan, M.; Han, K. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustain. Cities Soc.* **2018**, *38*, 697–713. [[CrossRef](#)]
112. Marques, P.; Manfroi, D.; Deitos, E.; Cegoni, J.; Castilhos, R.; Rochol, J.; Pignaton, E.; Kunst, R. An IoT-based smart cities infrastructure architecture applied to a waste management scenario. *Ad Hoc Netw.* **2019**, *87*, 200–208. [[CrossRef](#)]
113. Pardini, K.; Rodrigues, J.J.; Diallo, O.; Das, A.K.; de Albuquerque, V.H.C.; Kozlov, S.A. A smart waste management solution geared towards citizens. *Sensors* **2020**, *20*, 2380. [[CrossRef](#)] [[PubMed](#)]
114. Pardini, K.; Rodrigues, J.J.; Kozlov, S.A.; Kumar, N.; Furtado, V. IoT-based solid waste management solutions: A survey. *J. Sens. Actuator Netw.* **2019**, *8*, 5. [[CrossRef](#)]
115. Sharma, M.; Joshi, S.; Kannan, D.; Govindan, K.; Singh, R.; Purohit, H. Internet of Things (IoT) adoption barriers of smart cities’ waste management: An Indian context. *J. Clean. Prod.* **2020**, *270*, 122047. [[CrossRef](#)]

116. Aazam, M.; St-Hilaire, M.; Lung, C.H.; Lambadaris, I. Cloud-based smart waste management for smart cities. In Proceedings of the 2016 IEEE 21st international workshop on computer aided modelling and design of communication links and networks (CAMAD), Toronto, ON, Canada, 23–25 October 2016; pp. 188–193. [CrossRef]
117. Sallang, N.C.A.; Islam, M.T.; Islam, M.S.; Arshad, H. A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment. *IEEE Access* **2021**, *9*, 153560–153574. [CrossRef]
118. Wang, C.; Qin, J.; Qu, C.; Ran, X.; Liu, C.; Chen, B. A smart municipal waste management system based on deep-learning and Internet of Things. *Waste Manag.* **2021**, *135*, 20–29. [CrossRef]
119. Dang, L.M.; Piran, M.J.; Han, D.; Min, K.; Moon, H. A survey on internet of things and cloud computing for healthcare. *Electronics* **2019**, *8*, 768. [CrossRef]
120. Alam, A.; Qazi, S.; Iqbal, N.; Raza, K. Fog, edge and pervasive computing in intelligent internet of things driven applications in healthcare: Challenges, limitations and future use. *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2020; pp. 1–26. [CrossRef]
121. Mutlag, A.A.; Abd Ghani, M.K.; Arunkumar, N.a.; Mohammed, M.A.; Mohd, O. Enabling technologies for fog computing in healthcare IoT systems. *Future Gener. Comput. Syst.* **2019**, *90*, 62–78. [CrossRef]
122. Poncette, A.S.; Mosch, L.; Spies, C.; Schmieding, M.; Schiefenhövel, F.; Krampe, H.; Balzer, F. Improvements in patient monitoring in the intensive care unit: Survey study. *J. Med. Internet Res.* **2020**, *22*, e19091. [CrossRef] [PubMed]
123. Naik, B.N.; Gupta, R.; Singh, A.; Soni, S.L.; Puri, G. Real-time smart patient monitoring and assessment amid COVID-19 pandemic—An alternative approach to remote monitoring. *J. Med. Syst.* **2020**, *44*, 131. [CrossRef] [PubMed]
124. Davoudi, A.; Malhotra, K.R.; Shickel, B.; Siegel, S.; Williams, S.; Ruppert, M.; Bihorac, E.; Ozrazgat-Baslanti, T.; Tighe, P.J.; Bihorac, A.; et al. Intelligent ICU for autonomous patient monitoring using pervasive sensing and deep learning. *Sci. Rep.* **2019**, *9*, 8020. [CrossRef]
125. Khan, M.A.; Din, I.U.; Kim, B.S.; Almogren, A. Visualization of Remote Patient Monitoring System Based on Internet of Medical Things. *Sustainability* **2023**, *15*, 8120. [CrossRef]
126. Habib, C.; Makhoul, A.; Darazi, R.; Couturier, R. Health risk assessment and decision-making for patient monitoring and decision-support using wireless body sensor networks. *Inf. Fusion* **2019**, *47*, 10–22. [CrossRef]
127. Rehm, G.B.; Woo, S.H.; Chen, X.L.; Kuhn, B.T.; Cortes-Puch, I.; Anderson, N.R.; Adams, J.Y.; Chuah, C.N. Leveraging IoTs and machine learning for patient diagnosis and ventilation management in the intensive care unit. *IEEE Pervasive Comput.* **2020**, *19*, 68–78. [CrossRef]
128. Balaji, T.; Annavarapu, C.S.R.; Bablani, A. Machine learning algorithms for social media analysis: A survey. *Comput. Sci. Rev.* **2021**, *40*, 100395.
129. Sarker, I.H. Machine learning: Algorithms, real-world applications and research directions. *SN Comput. Sci.* **2021**, *2*, 160. [CrossRef]
130. Samie, F.; Bauer, L.; Henkel, J. From cloud down to things: An overview of machine learning in internet of things. *IEEE Internet Things J.* **2019**, *6*, 4921–4934. [CrossRef]
131. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
132. Polikar, R. Ensemble learning. In *Ensemble Machine Learning: Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–34. [CrossRef]
133. van de Ven, G.M.; Tuytelaars, T.; Tolias, A.S. Three types of incremental learning. *Nat. Mach. Intell.* **2022**, *4*, 1185–1197. [CrossRef]
134. Yamauchi, K.; Yamaguchi, N.; Ishii, N. Incremental learning methods with retrieving of interfered patterns. *IEEE Trans. Neural Netw.* **1999**, *10*, 1351–1365. [CrossRef]
135. Li, Y.; Wang, X.; Zeng, R.; Donta, P.K.; Murturi, I.; Huang, M.; Dustdar, S. Federated Domain Generalization: A Survey. *arXiv* **2023**, arXiv:2306.01334. <https://doi.org/10.48550/arXiv.2306.01334>.
136. Donta, P.K.; Srirama, S.N.; Amgoth, T.; Annavarapu, C.S.R. Survey on recent advances in IoT application layer protocols and machine learning scope for research directions. *Digit. Commun. Netw.* **2022**, *8*, 727–744. [CrossRef]
137. Donta, P.K.; Dustdar, S. Towards Intelligent Data Protocols for the Edge. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023; pp. 372–380. [CrossRef]
138. Bajrami, X.; Gashi, B.; Murturi, I. Face recognition performance using linear discriminant analysis and deep neural networks. *Int. J. Appl. Pattern Recognit.* **2018**, *5*, 240–250. [CrossRef]
139. Cox, D.R. Causality: Some statistical aspects. *J. R. Stat. Soc. Ser. A* **1992**, *155*, 291–301. [CrossRef]
140. Chen, P.; Qi, Y.; Hou, D. CauseInfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. *IEEE Trans. Serv. Comput.* **2016**, *12*, 214–230. [CrossRef]
141. Liu, Q.; Wang, C.; Wang, Q. Bayesian Uncertainty Inferencing for Fault Diagnosis of Intelligent Instruments in IoT Systems. *Appl. Sci.* **2023**, *13*, 5380. [CrossRef]
142. Al Ridhawi, I.; Aloqaily, M.; Karray, F.; Guizani, M.; Debbah, M. Realizing the tactile internet through intelligent zero touch networks. *IEEE Netw.* **2022**. [CrossRef]
143. Cheikhrouhou, S.; Maamar, Z.; Mars, R.; Kallel, S. A time interval-based approach for business process fragmentation over cloud and edge resources. *Serv. Oriented Comput. Appl.* **2022**, *16*, 263–278. [CrossRef]

144. Murturi, I.; Dustdar, S. Decent: A decentralized configurator for controlling elasticity in dynamic edge networks. *ACM Trans. Internet Technol.* **2022**, *22*, 1–21. [[CrossRef](#)]
145. Dehury, C.K.; Donta, P.K.; Dustdar, S.; Srirama, S.N. CCEI-IoT: Clustered and Cohesive Edge Intelligence in Internet of Things. In Proceedings of the 2022 IEEE International Conference on Edge Computing and Communications (EDGE), Barcelona, Spain, 11–15 July 2022; pp. 33–40. [[CrossRef](#)]
146. Sedlak, B.; Pujol, V.C.; Donta, P.K.; Dustdar, S. Controlling Data Gravity and Data Friction: From Metrics to Multidimensional Elasticity Strategies. In Proceedings of the 2023 IEEE International Conference on Software Services Engineering (SSE), Chicago, IL, USA, 2–8 July 2023; pp. 43–49. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.