Future Generation Computer Systems 159 (2024) 134–150

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

From cloud and fog computing to federated-fog computing: A comparative analysis of computational resources in real-time IoT applications based on semantic interoperability

Edgar Huaranga-Junco [a], Salvador González-Gerpe [b], Manuel Castillo-Cara [c,*], Andrea Cimmino [b], Raúl García-Castro [b]

[a] Universidad de Lima, Lima, Peru
[b] Universidad Politécnica de Madrid, Madrid, Spain
[c] Universidad Nacional de Educación a Distancia, Madrid, Spain

ARTICLE INFO

Keywords:
Fog computing
Federated-fog computing
IoT architectures
Semantic interoperability
Performance evaluation
Data federation

ABSTRACT

In contemporary computing paradigms, the evolution from cloud computing to fog computing and the recent emergence of federated-fog computing have introduced new challenges pertaining to semantic interoperability, particularly in the context of real-time applications. Fog computing, by shifting computational processes closer to the network edge at the local area network level, aims to mitigate latency and enhance efficiency by minimising data transfers to the cloud. Building upon this, federated-fog computing extends the paradigm by distributing computing resources across diverse organisations and locations, while maintaining centralised management and control. This research article addresses the inherent problematics in achieving semantic interoperability within the evolving architectures of cloud computing, fog computing, and federated-fog computing. Experimental investigations are conducted on a diverse node-based testbed, simulating various end-user devices, to emphasise the critical role of semantic interoperability in facilitating seamless data exchange and integration. Furthermore, the efficacy of federated-fog computing is rigorously evaluated in comparison to traditional fog and cloud computing frameworks. Specifically, the assessment focuses on critical factors such as latency time and computational resource utilisation while processing real-time data streams generated by Internet of Things (IoT) devices. The findings of this study underscore the advantages of federated-fog computing over conventional cloud and fog computing paradigms, particularly in the realm of real-time IoT applications demanding high performance (lowering CPU usage to 20%) and low latency (with picks up to 300ms). The research contributes valuable insights into the optimisation of processing architectures for contemporary computing paradigms, offering implications for the advancement of semantic interoperability in the context of emerging federated-fog computing for IoT applications.

1. Introduction

In recent years, the evolution of wireless networks, mobile devices, and computing paradigms has spurred the development of diverse information and communication-assisted services [1,2]. Internet of Things (IoT) technologies play an increasingly vital role in monitoring and facilitating control and production processes, spanning from Smart City applications to rural area management [3,4]. In other domains, some research introduces a hierarchical multi-tier approach integrating edge, fog, and cloud computing, which aligns with the emerging compute continuum paradigm featuring innovations like adaptive data compression, homomorphic encryption, and priority-based offloading mechanisms to address limitations in current computational architectures [5]. With a primary focus on enhancing network operation, fog computing has emerged as a technique to extend the cloud computing model to the network's edge, aiming to enhance overall network performance [6,7].

This technology addresses the mobility, geographical, and latency requirements of various IoT applications, including those on smartphones [3,8,9]. The increasing deployment of interconnected devices has spurred the emergence of power management and self-healing

* Corresponding author.
E-mail addresses: ehuarang@ulima.edu.pe (E. Huaranga-Junco), salvador.gonzalez.gerpe@upm.es (S. González-Gerpe), manuelcastillo@dia.uned.es (M. Castillo-Cara), andreajesus.cimmino@upm.es (A. Cimmino), r.garcia@upm.es (R. García-Castro).
URL: https://www.manuelcastillo.eu (M. Castillo-Cara).

https://doi.org/10.1016/j.future.2024.05.001
Received 13 October 2023; Received in revised form 29 April 2024; Accepted 2 May 2024
Available online 10 May 2024

mechanisms as critical design parameters for fog computing platforms which are essential for enabling the compute continuum [1,10]. Consequently, IoT devices generate vast amounts of data in various formats and at high speeds [7,11]. The efficient analysis and management of this information within cloud or fog computing architectures poses challenges [12,13]. In particular, time-sensitive data in smart cities, traffic control, and autonomous vehicles require real-time processing for swift and effective solutions [1,2,4]. Furthermore, concerns include data that compromise privacy and geographically distributed data [14, 15].

Despite recent progress, creating and implementing robust IoT systems that meet end-user application requirements pose several challenges [16,17]. In addition to enhancing communication platform performance, designing a resilient and scalable data processing architecture is crucial [7,13,18]. Data processing in IoT architectures has shifted towards distributed paradigms, such as federated-fog computing [16,19].

The emergence of the federated-fog computing paradigm addresses the growing demands for computing and communication requirements in data federation, which is a key aspect of the compute continuum [20, 21]. Researchers in this new paradigm must design and develop IoT distributed systems comprising numerous data sources and smart data-intensive processing to meet end-users' Quality of Service (QoS) needs and application requirements [8,19]. Privacy concerns and geographic dispersion further complicate the landscape [15]. Taking into account this context, it is crucial to focus on two main parameters of network design for wireless and wired networks: power and resource management and network resilience mechanisms [10,11,18].

In this study, we present the innovative IoT federated-fog computing framework and investigate its implications for computational resource consumption and latency in wired networks [11]. Our analysis is grounded in the backdrop of classical computing architectures, such as fog and cloud computing, which pose limitations for the implementation of real-time IoT applications relying on semantic interoperability [9,21]. Addressing inherent challenges in this domain, including heterogeneity and scalability issues in data federation [19, 21], our research contributes to the development of scalable architectures and systems for the compute continuum. Hence, our investigation extends to the comparison of this novel architecture with established cloud and fog computing models, shedding light on the distinct advantages offered by federated-fog computing. Specifically, we delve into a specific instance of the fog computing framework, termed inverted-fog computing, where fog nodes assume dual roles in initial data filtration and storage. Through a series of rigorous experiments, including simulations of real-time data warehousing based on semantic interoperability [22,23], we discern that federated IoT frameworks exhibit superior performance in latency and resource utilisation when compared to traditional cloud and fog computing paradigms.

In this context the systems must be able to exchange and understand data in a transparent way despite the heterogeneity of their data stacks (protocols to exchange data, formats, or data models). To this end, a semantic interoperability approach is adopted in the article to cope with the heterogeneity of IoT vendors that rely on different protocols, formats, and models. As a result, the study presented in this article lays over semantic interoperable data architectures. The significance of our findings lies in their contribution to the evolving landscape of IoT architectures, particularly in addressing challenges related to semantic interoperability and data federation [10,24]. Furthermore, our study represents a pioneering effort in evaluating various federated architectures within the context of semantic interoperability for IoT [22]. By elucidating the advantages and challenges inherent in these architectures, our work provides insights into the design and implementation of distributed and decentralised management of resources and application deployment in the compute continuum. This article presents the following contributions.

- We develop and implement two innovative IoT architectures, inverted- and federated-fog computing, for real-time IoT applications that rely on semantic interoperability.
- We assess resource consumption and latency times on all four IoT architectures, including cloud and fog computing, as well as the newly introduced inverted- and federated-fog computing.
- We argue that new federated IoT architectures present a balanced resource consumption and latency time for deploying real-time applications based on semantic interoperability.
- We assert that these architectures ensure a balanced distribution of geographically dispersed data while maximising scalability, reliability, availability, and data ownership, among other benefits.

Finally, this paper is structured as follows: Section 2 presents a comprehensive review of related work on IoT architecture solutions, semantic interoperability, and data federation. Following this, Section 3 outlines the principal characteristics of the use case application, IoT architecture ecosystem, and Semantic interoperability specifications. In Section 4, a detailed description of the studied architectures, their stream data flows, and testbed specifications are provided. Finally, Section 5 presents and analyses the results of the experiment, while Section 6 provides a summary of the findings and suggests avenues for future research in this field.

2. Related work

There are numerous concerns regarding the creation and execution of dependable computing infrastructures that can fulfil end-user application prerequisites. As well as enhancing the function of the communication platforms, a significant issue is developing an architecture for processing data that is scalable and dependable. To this end, the architecture of distributed systems has evolved from a cloud paradigm to a fog/edge paradigm. Currently, it has shifted to federated-fog computing.

2.1. IoT computational architectures

The framework of Information and Communication Technologies (ICT) within a smart city context encompasses one or multiple Internet of Things (IoT) applications that integrate technologies such as low-power IoT networks, device management, and analytic or event stream processing [25,26]. These integrated systems enable the extraction of raw data from smart objects and sensors, the subsequent processing, and the derivation of insights to improve IoT applications and infrastructures [7]. Consequently, IoT designs commonly adopt a multi-layer architecture, typically leveraging edge/fog computing paradigms [27,28]. The fog computing architecture comprises a core level connected to the cloud and an edge level connected to peripheral devices, such as sensors or smartphones [10]. The core level encompasses the main computing components such as the main database, Complex Event Processing (CEP), brokers, and other elements [29], while the edge level comprises peripheral components like wireless sensor and actuator networks, fog nodes (with a broker and local CEP), and smartphones [30].

In Rodrigues et al. [5] introduces the VitalSense model, aimed at revolutionising healthcare services in smart cities by leveraging IoT vital sign data. It addresses limitations in existing computational architectures by proposing a hierarchical multitier approach integrating edge, fog, and cloud computing. Key innovations include adaptive data compression, homomorphic encryption, multi-tier notifications, low-latency health traceability, a serverless execution engine, and priority-based offloading mechanisms. Preliminary evaluations demonstrate the potential for disruptive healthcare services, underscoring VitalSense's importance in reshaping healthcare delivery within urban environments. About fog computing architectures, effective communication

among harvested nodes is crucial, given their diverse features, including location, distribution, scalability, device density, mobility support, and real-time and standardised data models [1]. Despite notable progress, the development and implementation of robust IoT-based systems that meet the application needs of the end user face unresolved challenges [10]. Emerging IoT architectures, notably federated-fog computing, allocate data storage at the edge level rather than the core level [12].

Federated-fog computing, an evolving field, aims to enable efficient and scalable data processing in IoT environments [21]. This distributed computing model merges federated learning and fog computing principles [18], leveraging fog nodes (edge devices) proximate to IoT devices for computation, storage, and communication tasks. The primary objective is to enhance intelligence and decision-making near IoT devices, thereby reducing latency, conserving network bandwidth, and improving privacy [18].

A typical federated-fog computing architecture includes IoT devices (sensor nodes, actuators, etc.) producing data in the IoT ecosystem [7], fog nodes responsible for collecting, processing, and analysing data from IoT devices [10], and cloud infrastructure that serves as a central hub for system management and coordination, and provides additional computational resources and long-term storage capabilities [10]. Semantic interoperability ensures meaningful data exchange between various IoT devices and fog nodes [29], employing standardised data models, semantic technologies, and ontologies to foster data interoperability, integration, and discovery [31,32].

2.2. Semantic interoperability and data federation

Semantic interoperability plays a pivotal role in enabling transparent data exchange and consumption between systems. Although numerous proposals for implementing semantic interoperability exist in the literature [17], traditional cloud-based approaches have been surpassed by fog computing, which offers distinct advantages but lacks comprehensive analysis regarding semantic interoperability [9]. In semantic interoperable approaches, practitioners often resort to creating ad hoc codes to accommodate heterogeneous data [24]. However, this flexible approach requires significant investment in personnel and code maintenance, limiting its reusability [24].

Federated fog computing is highly dependent on semantic interoperability to facilitate seamless integration and interaction between various IoT devices and fog nodes [10]. Leveraging semantic technologies such as ontologies, the Resource Description Framework (RDF), and reasoning mechanisms is central to this process, ensuring a universal understanding of data and promoting data sharing, discovery, and integration across various domains [23].

On the contrary, data federation involves combining and accessing data from disparate sources while treating them as a unified database [21]. This allows organisations to use data from different systems without the need for data replication or complex integration procedures [19]. Data federation aims to achieve a unified view of data by integrating and accessing it from distributed sources, eliminating the need for data duplication, simplifying data integration complexity and allowing real-time access to data from diverse systems [31].

Methods for data federation are described in [20]: (i) query rewriting methods transform queries aimed at a federated database into executable on separate data sources, facilitating seamless retrieval and integration of data from multiple sources; and (ii) schema mapping methods identify relationships between schemas of different data sources, aiding data integration at the schema level by defining mappings between attributes, relationships, and data types. Furthermore, data virtualisation provides an abstraction layer that grants access to and integration of data without physical transfer or replication, enabling real-time access to federated data through a consolidated view of distributed data sources [29].

In scenarios like IoT, once semantic interoperability is achieved, the federation allows to consume data of decentralised systems rather than traditional approaches that require to centralise data into a database. Note that semantic interoperability involves publishing RDF data and, therefore, data federation is achieved based on the W3C standard SPARQL 1.1 protocol [33]. This protocol broadcasts to all data endpoints a query and gathers their responses into a unified query result. Note that the W3C SPARQL federation does not follow a vertical or horizontal approach, as it is agnostic of the data content of each distributed data source.

2.3. Implications

Classical computing architectures, such as fog and cloud computing, present limitations in the implementation of real-time IoT applications that rely on semantic interoperability [9]. This study examines the innovative federated-fog computing architecture and addresses several inherent challenges in this type of IoT architecture [21]. We deploy our federated-fog computing architecture, taking into account these challenges.

One of the primary challenges encountered in data federation is heterogeneity [19]. Effectively managing disparate data sources with varying data models, structures, and query languages requires the use of techniques such as data integration, schema mapping, and query translation to address this heterogeneity. Another significant challenge is scalability [21]. As the number of data sources increases, maintaining the scalability of data federation systems becomes increasingly complex. It requires the application of efficient techniques to optimise queries, distribute data, and process data in parallel to handle large-scale federated databases effectively [31].

Moreover, fog computing and federated-fog computing face challenges in achieving seamless integration and interaction among various IoT devices and fog nodes, particularly in terms of semantic interoperability [10]. Traditional fog computing approaches often require ad hoc coding to accommodate heterogeneous data, leading to high costs in personnel and code maintenance [24].

In addressing these issues, our study introduces a novel architecture for IoT federated-fog computing and conducts an analysis of resource consumption and latency [18]. We compare and contrast this innovative architecture with well-established cloud and fog computing architectures [12]. A series of tests were carried out to assess the performance of each architecture, simulating an IoT application for real-time data warehousing based on semantic interoperability [10]. Additionally, we propose a new architecture for data federation called inverted-fog computing, which lies between fog computing and federated-fog computing [29].

To the best of our knowledge, this study represents the first effort to evaluate various federated architectures in semantic interoperability for IoT [23]. The discussion surrounding the challenges and advantages of IoT architectures in our study is intricately linked to the current state of the art in IoT architectures, particularly with respect to semantic interoperability and data federation. Additionally, in this article the focus is on a semantic interoperability environment with limited data and from a particular domain, in contrast to studies such as [22,23], where the study has been carried out with large-scale semantic data from different domains.

Finally, the emergence of the compute continuum paradigm promises to simplify the execution of distributed applications by managing the heterogeneity and dynamism of widespread computing resources, from the edge to the cloud. In this context, our work presents a novel IoT federated-fog computing framework that aligns with the compute continuum vision. Although our study primarily focuses on fog environments, our proposed architecture can be seen as a stepping stone towards the compute continuum, as it enables seamless integration and management of distributed computing resources across multiple tiers. By addressing the challenges of semantic interoperability

and data federation in real-time IoT applications, our research contributes to the development of scalable and efficient architectures for the compute continuum. Therefore, our work fits well within the scope of this special issue, as it investigates and gathers research contributions on the emerging compute continuum, seeking solutions for running distributed applications while efficiently managing heterogeneous and widespread computing resources.

3. Background: Tools and the IoT architecture ecosystem

The evolution of IoT technologies has come about from the convergence of wireless technologies, micro-electromechanical systems, micro-services and the Internet, generating a common communication language between them. This refers to a network of interconnected objects through the Internet. This requires the integration of technologies and mechanisms in an orderly manner and maximising the quality of service provided by the architecture as a whole.

In this section, we describe in detail the layers that compose the different IoT architectures in which our experiments are focused, their components, and the key functional aspects of the proposal.

3.1. Tools: Software and hardware

Now the following section outlines the primary hardware and software components of the testbed that is developed to conduct experiments.

3.1.1. Hardware

The implemented IoT architecture comprises of two primary tiers, the edge level and core level (see Section 3.2). The first processing node that carries out the initial data processing is located at the edge level, while the core level, which emulates the cloud, is situated at a secondary level. Consequently, the architecture is composed of two virtual machines, one each at the edge level and core level. Both virtual machines are situated on the same local network. Thus, the edge-level virtual machine is a low-performance device emulating the features of a Raspberry Pi v4 with a 2-core 64-bit 1.4 GHz processor, a 2 GB RAM LPDDR2 SDRAM, a 20 GB hard disk, and an Ubuntu Server 20.04 operating system (without a Graphical User Interface (GUI)). Meanwhile, to maintain control over the surroundings (e.g., network latencies), the core level has been deployed on-premises through local resources. More specifically, the core level is operational on a 4-core 64-bit 1.4 GHz processor, 8 GB LPDDR2 SDRAM RAM, a 30 GB hard disk, and the Ubuntu Server 20.04 operating system (without a GUI).

3.1.2. Sysstat

Sysstat [34] is an open-source software package for Unix systems that provides a range of tools to monitor and collect data on system performance. It includes several programmes and utilities that enable system administrators to access information about the computational resources being used by the system [7]. In this study, we have acquired various metrics concerning CPU, RAM, hard drive, and network data transmission (for detailed parameters, refer to Section 4.3). The commands are set in Crontab to measure the computational resource consumption at the onset of the stress test explained in Section 4.4.

3.1.3. Helio

Helio [35] is a software framework that enables the execution of various lifecycle tasks of knowledge graphs [36]. One of the built-in capabilities that the framework provides is the translation of heterogeneous data into homogeneous RDF expressed according to an ontology. Semantically interoperable and transparent data exchange across numerous IoT architectures is achieved utilising Helio in the experiments conducted. This tool is deployed in various scenarios at various levels, namely edge and core, to assess the computational impact of enabling the semantic interoperable layer.

3.1.4. Triplestore

Semantic interoperable data is expressed using the RDF, which is a W3C standard [37]. Databases that store RDF data are known as triplestore. These databases allow for the storage and querying of RDF data, among other operations related to the knowledge graph lifecycle. For this experiment, the RDF database used is GraphDB [38] triplestore.[1] GraphDB features an API that makes it possible to store and query RDF data using the SPARQL language, which is also a W3C standard [39]. During the experiments, GraphDB is deployed at distinct levels, namely edge and core. Subsequently, the experimental results highlight the computational effect that these operations might have on federated-fog architectures.

3.2. IoT architecture ecosystem

In this work, we have created and implemented four primary IoT architectures for benchmark evaluation (Section 4 provides details about each architecture). Technical term abbreviations are explained when first used. For the purpose of this section, we have classified them into two main groups, as illustrated in Fig. 1. The initial group examines the traditional IoT structures encompassing cloud and fog computing (see Fig. 1(a)). Meanwhile, the latter group examines the contemporary IoT structures comprising inverted- and federated-fog computing (see Fig. 1(b)).

3.2.1. Cloud and fog computing

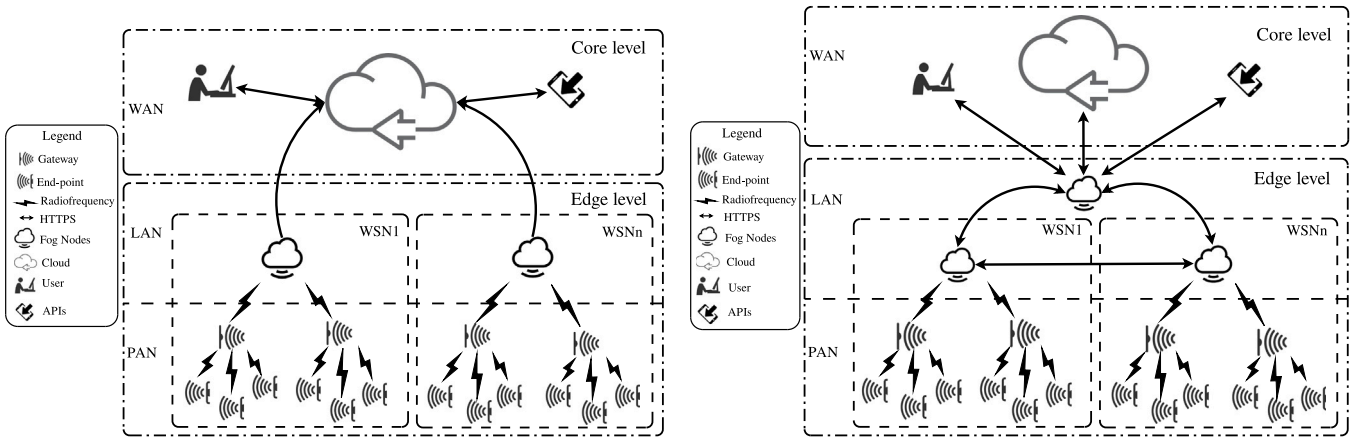
The concept of fog computing represents a natural expansion of cloud computing. It involves the use of intelligent devices, referred to as fog nodes, which are situated close to the end user for performing data processing and providing user services. Typically, application protocols that use TCP/IP are necessary for gathering data from IoT devices and transmitting them to the cloud. Fog computing, in comparison to cloud computing, refers to a horizontal architecture that can be duplicated at different network levels. It ultimately provides advantages such as cognition, increased security, real-time analytics, reduced latency and bandwidth requirements, and offline availability. Consequently, architectures that incorporate fog computing expedite data processing and event response by eliminating the need for cloud analysis, resulting in improved efficiency. Sensitive data are safeguarded through analysis within the local network, despite exposure to the cloud. This exposure can be mitigated by a federated architecture, thereby enhancing service levels, privacy, and security.

In this context, the computing architecture – be it cloud or fog – is bifurcated into two tiers: edge and core [10]. The technology components, mechanisms, and features of each level correspond to the services they offer, as illustrated in Fig. 1(a). On one hand, we have the vertical component advancing Wireless Sensor Networks (WSNs) at the edge level, while on the other hand, we have various network areas in the horizontal component. The regions consist of the Personal Area Network (PAN), which comprises the sensors and Gateway; the Local Area Network (LAN), which acts as the connection between the Gateway and the fog nodes; and the Wide Area Network (WAN), which links the fog nodes with the cloud node.

At the edge level, the most important and critical element of the fog computing architecture under consideration is the fog node, which is located within the LAN layer (see Fig. 1(a)). It acts as a point of connection between the edge level and the core level of the platform. Moreover, it has the capability to receive, translate and transmit data to the triplestore. Therefore, the main function of the fog node in our IoT architecture is to decentralise our platform. It has the ability to host the Helio tool and the triplestore, as outlined in Section 4.

At the core level, the studied IoT architecture features the traditional cloud component, which is linked to the fog node through the

[1] https://graphdb.ontotext.com/



(a) Cloud and fog computing. (b) Inverted- and federated-fog computing.

Fig. 1. Reference ecosystems for the IoT architectures.

HTTP protocol, with a specific emphasis on implementing the POST method. Ultimately, HTTP requests are used by APIs and end-users seeking requested information from the cloud node. As described in Section 4, if the cloud node functions as a storage component, it can retrieve and provide this information. However, it may also serve as a resource orchestration component considering that it is the fog nodes that store this data.

3.2.2. *Inverted- and federated-fog computing*

The primary aim of developing decentralised IoT architectures is to enhance service and resource optimisation for deployed components within a WSN, whether on the hardware or software side. Among these components, fog/edge computing architectures are mainly characterised by fog nodes performing data processing and service deployment at the edge level. These fog nodes are in closer proximity to end-users rather than the central processing system (i.e., the cloud node). It appears that cloud and fog computing structures do not completely eliminate the need for processing and, notably, storage (see Fig. 1(a)). This is where federated-fog computing arises, in decentralised data storage. Therefore, federated architectures are a promising paradigm for future sixth-generation wireless systems to support network edge intelligence for semantic interoperability applications. The main characteristics of these architectures are as follows:

- Decentralised storage is utilised as there is no central point of storage, instead, information is stored in various fog nodes. This means that the cloud node does not hold any information produced by the WSNs.
- Data ownership lies with the end user, as there is no central information storage node. The access to data is subject to policies and requirements determined by the corresponding fog nodes.
- Decentralised processing and accessibility are also implemented. As information is stored in various fog nodes, they all require the same data model for full accessibility, in this instance RDF.
- The foundation of semantic interoperability necessitates a sturdy yet adaptable ontology network that incorporates established standard ontologies and encompasses the informational domain beyond these standards.
- Achieving semantic interoperability is essential. The vast amount of data from a diverse range of harvesting nodes needs to be adjusted to ensure that all components comprehend the coordinated management of data throughout the platform.
- Reliability, availability, and survivability (RAS) are essential factors in this process. Each node follows the same ontology and is accessible under a uniform data model within the architecture.
- The design is scalable and can accommodate the considerable size of IoT networks because of the uniform ontology. Therefore, the structure has the capacity to expand to various nodes, which are capable of communicating using a common protocol.

Furthermore, the federated-fog computing architecture outlined in this study (see Fig. 1(b)) aims to eliminate the need for cloud-based storage. As a result, information is distributed throughout various fog nodes. Federated-fog computing paradigm allows the control and federation of fog resources across multiple operating domains, with data stored in the fog nodes. Notable differences have been identified between cloud and fog computing, as well as inverted- and federated-fog computing (see Figs. 1(a) and 1(b)), have been identified:

- All fog nodes must be connected to each other, as it is an essential requirement for accessing data in such architectures. For this purpose, the fog nodes must be tagged with an IP address.
- To ensure this, fog nodes store information in an inverted- and federated-fog computing architecture, with each node distributing information related to each WSN through a triplestore.
- In this model, a master fog node orchestrates data read queries. The deployment of the federated query engine is essential as it is responsible for both requesting and standardising the data from users and APIs.
- External queries are channelled through the master fog node. For example, when a user or API initiates a query, the master fog node extracts the necessary information and forwards the request to the different fog nodes that are deployed in each WSN.

Finally, this study also explores an alternative variant of federated-fog computing, namely inverted-fog computing, where Helio is not installed on the fog node and only the triplestore is deployed in the fog node.

3.3. *Semantic interoperability*

Interoperability is the characteristic that allows systems to exchange and understand information in a transparent way [40]. Implementing this feature has become one of the most relevant challenges to address in the IoT domain due to the heterogeneity of vendors that rely on different protocols, formats, and models. As a result, the benefit of semantic interoperability is to allow heterogeneous systems to exchange and consume data seamlessly.

To implement semantic interoperability, an ontology is established as a consensus common data model and all data must be expressed according to it and thus in any RDF serialisation [41]. How the data from the original IoT environment (sensors, gateways, etc.) are translated

into RDF, and how it is ensured that it is compliant with the ontology is the challenge that is being addressed today.

In order to make an IoT environment semantically interoperable, there are different approaches [4]. One approach consists of a developer coding an adapter that takes the known data that are being exchanged and then translates them into a valid RDF payload that follows the consensus ontology. Another approach consists in using existing tools that are endowed with the ability to translate some data into RDF following an ontology, i.e., perform data harmonisation.

In this article, semantic interoperability is implemented by using the Helio to perform data harmonisation. Helio relies on declarative mappings that hold different translation rules that, given a set of heterogeneous sources, are able to produce harmonised data, i.e., RDF data following a specific ontology. Thus, the ontology used to express the harmonised data is described and the mappings developed.

Thanks to the fact that Helio has multiple connectors to retrieve data from heterogeneous sources and, publishes the harmonised data through a HTTP REST API its usage covers the technical, syntactic, and semantic interoperability. On the one hand, technical interoperability is achieved by harmonising the protocol to retrieve data; in particular, Helio provides an HTTP REST API. On the other hand, syntactic interoperability is achieved thanks to its harmonisation of data publishing all the data as RDF. Finally, semantic interoperability is implemented due to the usage of ontologies and, particularly in the context of this article, the SAREF standard.

3.3.1. Ontology

The RDF data should be expressed according to a semantic model, specifically an ontology expressed in W3C Web Ontology Language (OWL) [31,32]. Although there are numerous ontologies, certain ones are endorsed by standardisation bodies. Such standard ontologies are appropriate for expressing RDF data because they enable interoperability by design. A commonly recognised ontology for expressing IoT data is the Smart Applications REFerence Ontology (SAREF), and its extensions [42]. SAREF is a standard ontology developed by the European Telecommunications Standards Institute (ETSI) that provides a shared reference model for smart homes and IoT devices. This ontology offers "building blocks" that can be used to separate and modify various parts of the ontology based on specific requirements. The concept of a device serves as a starting point. These physical devices are intentionally created to accomplish specific purposes within households, communal and public buildings, and offices. The device executes one or more functions as described in the ontology model to carry out these tasks.

3.3.2. Data harmonisation

To achieve data harmonisation, the Helio tool is used, as described in Section 3.1.3. For this purpose, a collection of mappings with multiple translation rules is created. The input for these mappings is a data stream in JSON format received from sensors, which is then translated into RDF using the JSON-LD 1.1 (here on in referred to as JSON-LD) serialisation format following the SAREF ontology. JSON-LD is chosen as the preferred format over others, such as TURTLE, due to its proximity to non-semantic experts and developers. Although it appears as regular JSON data, the JSON-LD format is, in fact, RDF. More information on data harmonisation can be found in the Helio mapping.[2] and the JSON-LD context[3]

The mapping contains two parts. The initial section is responsible for restructuring the JSON data from the sensors (see Appendix). The payload incorporates details on the measurement characteristics, including the value, timestamp, and related parameter. The mapping compares the parameter value with a `saref:Property` and a `saref:UnitOfMeasure` that are stated in the mapping context. The procedure for the first segment is shown in Fig. 2. The second part of the mapping consists of the construction and storage of the data in the triplestore instance (see Appendix). The mapping takes the JSON-LD and changes the serialisation into NT, then the result is sent as a POST request to the triplestore instance.

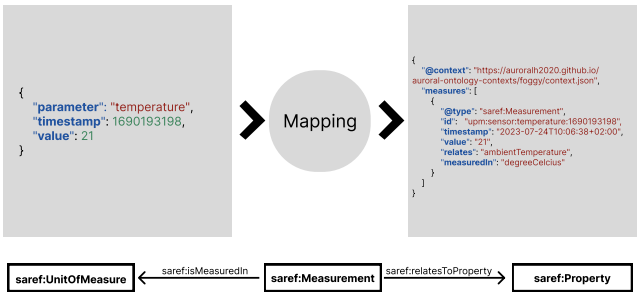


Fig. 2. Mapping function and its relative ontology represented in the process.

4. Contextual-streaming data flow

Our application is implemented using the following four architectural approaches: traditional cloud and fog computing, and inverted- and federated fog computing. In order to evaluate the performance of these four architectures, each of them was put under the same workload to evaluate the computational performance in the same fraction of time, i.e. real-time analysis.

4.1. IoT architecture components

The development of a centralised or distributed computational architecture for IoT applications necessitates the utilisation and integration of various services including identification, communication, data analysis or actuation, among others. To this end, our architectures illustrated in Fig. 3 present four different evaluation scenarios based on the deployment of the Helio and triplestore components. Note that these architectures are semantically interoperable and, without this feature, it would not be possible to exchange data and consume it seamlessly. Therefore, the semantic interoperability and its benefits is a paramount feature for these IoT architectures. This is the main benefit of adopting a semantic interoperable approach.

4.1.1. Scenario 1: Cloud computing

Cloud Computing (here on in referred to as Scenario 1) refers to the paradigm whereby information processing is performed by the core level, which is equipped with Helio and triplestore engines. The fog node found at the edge level is a passive component that oversees information transfer from the edge level to the core level without prior data processing.

4.1.2. Scenario 2: Fog computing

Fog computing (here on in referred to as Scenario 2) operates with the triplestore engine solely deployed at the core level, while the Helio engine is deployed at the edge level on the fog node. Therefore, the fog node is responsible for translating the data format and transmitting it to the core level for storage in the triplestore. Fog computing features initial data processing at the edge level, utilising the computational capacity of fog nodes. Thus, the initial data processing may encompass tasks such as cleaning and event detection, or, in the present scenario, the translation of packets from the JSON object to the JSON-LD format.

[2] https://auroralh2020.github.io/auroral-ontology-contexts/foggy/mapping.ftl

[3] https://auroralh2020.github.io/auroral-ontology-contexts/foggy/context.json

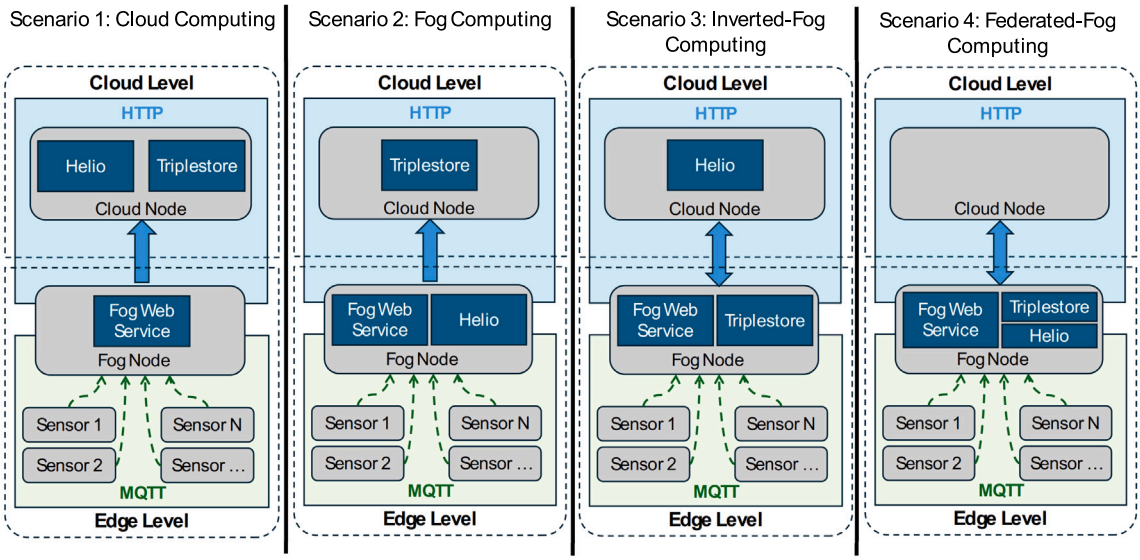


Fig. 3. Different scenarios proposed for the evaluation of the architectures.

4.1.3. Scenario 3: Inverted-fog computing

Inverted-fog computing (here on in referred to as Scenario 3) is a paradigm where only the Helio engine is deployed at the core level, while the triplestore engine is deployed at the edge level, specifically on the fog node. Consequently, the fog node becomes the data storage system. This scenario is an intriguing federated-fog computing particular case, given that the data are situated on the fog nodes, and the users are the ones who own the stored data. Data owners can set policies and requirements for external systems that wish to consume their data, with the ability to deploy geographically distributed data. This architecture enables the use of novel federated machine learning techniques for analysis and model development [15].

4.1.4. Scenario 4: Federated-fog computing

Federated-fog computing (here on in referred to as Scenario 4) involves deploying the Helio and triplestore engines in the fog node to process information. As a result, the core level's role is passive, with no involvement in the data's transformation or storage process. The data orchestration is performed by the fog node located at the edge level. In Scenario 3, users possess the stored data and have the ability to determine the policies and considerations for its external read consumption. The distinctive feature of this particular IoT architecture is that the cloud is devoid of all computational load and can therefore concentrate on resource-intensive tasks such as extensive data analysis and the innovative federated-machine learning technique.

4.2. Data orchestration

From the preceding discussion, it is paramount to analyse the distribution of the main components of semantic interoperability in the IoT architecture, namely, Helio and the triplestore (as shown in Fig. 4). These scenarios rely on HTTP protocol and, in particular, TCP communication is endowed as direct and synchronous by means of GET and POST requests. Although other protocols could be used in these scenarios, relying on direct HTTP communication facilitates measuring most of the experimental Key Performance Indicators (KPIs).

The combined effects of these two components, together with the federated query engine, enable the investigation of the deployment of computational resources concerning the orchestration of information flow between the edge and core levels. However, for the purposes outlined in this experiment, only the consumption of computational resources in the data transformation and storage process shall be examined.

4.2.1. Scenario 1: Cloud computing

Scenario 1 features two primary components, namely Helio and the triplestore, both of which operate at the core level. The fog node is a passive element in this IoT architecture. A depiction of the implementation of our system is visible in Fig. 4(a). With the aim of redirecting incoming packets from sensors, a web service is established, solely redirecting the JSON object containing the data to the core level. Specifically, the communication process proceeds as follows.

1. Sensor transmits the JSON packet to the fog node.
2. Fog node collects the JSON packet directed towards the core level, specifically to Helio. It is noteworthy that a web service has been established specifically for packet redirection.
3. At the core level, Helio translates the JSON packet into JSON-LD.
4. Helio transmits the JSON-LD packet package to be stored within the triplestore, which is likewise located at the core level.
5. Finally, the triplestore stores the JSON-LD packet and concludes the communication process.

4.2.2. Scenario 2: Fog computing

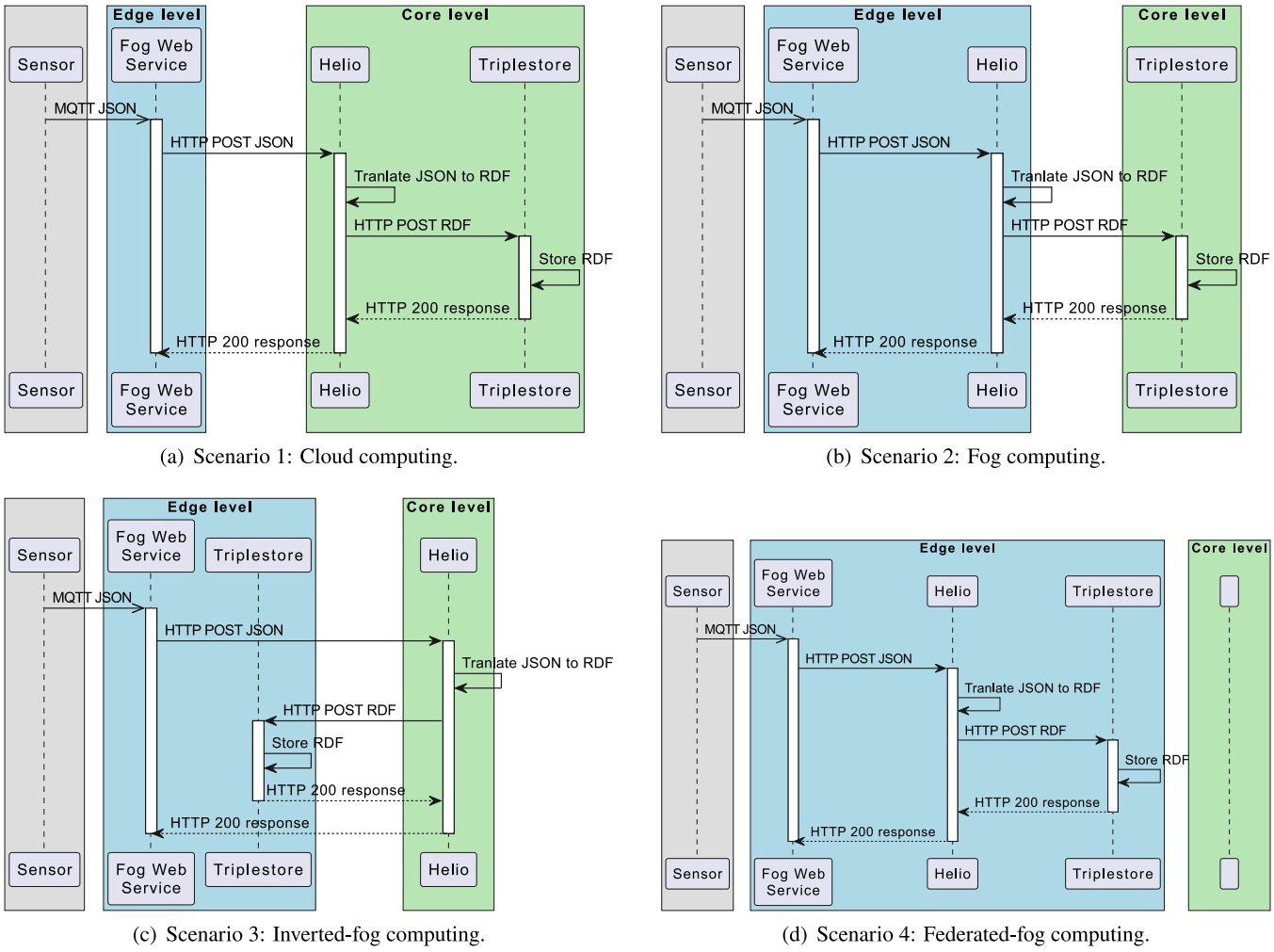
Scenario 2 varies from Scenario 1 as the Helio is deployed in fog nodes, i.e. at the edge level. Fig. 4(b) illustrates the communication procedure in the IoT Architecture for this Scenario. In essence, the communication process is established as follows:

1. The sensor transmits the JSON packet to the fog node.
2. Fog node receives the JSON packet that is sent to Helio, located at the edge level.
3. Helio translates the JSON packet into JSON-LD.
4. Helio transmits the JSON-LD packet to the triplestore, located at the core level.
5. Finally, the triplestore stores the JSON-LD packet and concludes the communication process.

4.2.3. Scenario 3: Inverted-fog computing

Scenario 3 is a unique situation and stems from the conventional fog computing architecture, but it is reversed from Scenario 2. Here, data translation occurs at the core level while data storage takes place at the edge level. The communication method in the IoT Architecture for this scenario is illustrated in Fig. 4(c). Specifically, the communication process follows as follows:

1. The sensor transmits the JSON packet directly to the core level, i.e., to the cloud.



(a) Scenario 1: Cloud computing.

(b) Scenario 2: Fog computing.

(c) Scenario 3: Inverted-fog computing.

(d) Scenario 4: Federated-fog computing.

Fig. 4. Data flow schema for the four scenarios.

2. Cloud receives the JSON packet and sends it to Helio, located at the core level.
3. Helio translates the JSON packet into JSON-LD.
4. Helio transmits the JSON-LD packet to the triplestore, located at the edge level.
5. Finally, the triplestore stores the JSON-LD packet and concludes the communication process.

4.2.4. Scenario 4: Federated-fog computing

Scenario 4 represents a federated architecture where both the Helio and triplestore engines are situated in the fog nodes. In contrast to Scenario 1, the core level in this scenario does not have any active components, making it a passive element. Fig. 4(d) illustrates the communication process in the IoT architecture for this scenario. Thus, the communication process is established accordingly.

1. The sensor transmits the JSON packet to the fog node.
2. The fog node receives the JSON packet that is sent to Helio, located at the edge level.
3. Helio translates the JSON packet into JSON-LD.
4. Helio transmits the JSON-LD packet to the triplestore, also located at the edge level.
5. Finally, the triplestore stores the JSON-LD packet and concludes the communication process.

4.3. Computational evaluation parameters

The analysis is based on a genuine implementation of four different scenarios. Moreover, the behaviour of authorised system users is modelled to improve the analysis of system scalability when handling data (refer to Section 4.4 for further details). The performance assessment encompasses both cost and efficiency analyses that factor in the use of computational resources and latency. Table 1, exhibits quantitative measurements obtained from the Sysstat software, fully explained in Section 3.1.2. The Time Latency parameter has been computed through a distinct procedure, comprehensively described in Section 4.3.2.

4.3.1. Computational resources

To monitor computationally usage in various scenarios assessed during our research, Sysstat software [34] was utilised. The primary evaluation parameters are categorised into four system blocks: CPU, RAM, Hard Disk and Network (details provided in Table 1). Specific parameter analysis is presented below.

CPU

- CPU usage (here on in referred to as CPU): Percentage of CPU utilisation that occurred while executing at the user level (application). Note that this field includes the time spent running virtual processors.

Table 1
Definition and representation of the performance parameters considered.

System	Parameter	Unit of measure	Sample values
CPU	Usage	Percentage	(1, 5, etc.)
RAM	Usage	Percentage	(15, 30, etc.)
	KB waiting to get written	Kbytes	(100, 600, etc.)
Hard disk	Transfers per second	TPS	(15, 30, etc.)
	KB read	Kbytes/s	(100, 200, etc.)
	KB write	Kbytes/s	(100, 200, etc.)
Network	Packets received per second	rxpck/s	(1, 5, etc.)
	Packets transmitted per second	txpck/s	(1, 5, etc.)
Latency	Translation Time	Millisecond	(100, 200, etc.)
	Transaction Time	Millisecond	(100, 200, etc.)

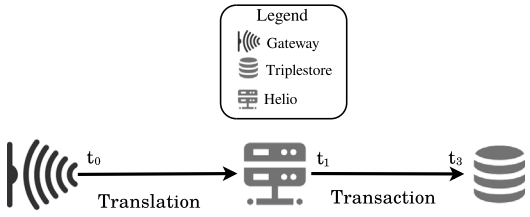


Fig. 5. Time Latency flow schema for the four scenarios.

RAM

- RAM (here on in referred to as RAM): Percentage of RAM memory used.
- KB waiting to get written (here on in referred to as kbdirty): Amount of memory in kilobytes waiting to be written back to the disk.

Hard disk

- Transfer per second (here on in referred to as TPS): Total number of transfers per second that are issued to physical devices. A transfer is an I/O request to a physical device. Multiple logical requests can be combined into a single I/O request to the device. A transfer is of indeterminate size.
- KB read (here on in referred to as KB_read): Number of kilobytes read from the device per second.
- KB write (here on in referred to as KB_write): Number of kilobytes written to the device per second.

Network

- Packets received per second (here on in referred to as rxpck): Total number of packets received per second.
- Packets transmitted per second (here on in referred to as txpck): Total number of packets transmitted per second.

4.3.2. Time latency

Latency time is considered for both Helio and the triplestore irrespective of their deployment location in the four analysed scenarios. Note that depending on the situation, both Helio and the triplestore are situated at a particular level in the architecture, either the edge level or the core level (see Fig. 5).

Although the architecture of the scenarios is distributed, there is no need of synchronizing clocks due to the fact that the time latency is measured with the requests lapse time. Thanks to the fact that all the experiments are based on direct and synchronous HTTP request, the lapse time is the time latency.

As a result, the time latency encodes the time taken to translate from JSON to JSON-LD and store the transaction in the triplestore. Consequently, in our experiment, latency is defined by:

- Translation time (here on in referred to as translation): is the time that takes JSON data from sensors to reach Helio and translate from JSON to JSON-LD. Therefore, it is the time that Helio needs to translate the JSON into RDF and prepare the query to insert the data into the triplestore. As shown in Section 4.3.2 the translation time [10] is defined by:

  $$Translation = t_1 - t_0$$

  where:

  - t_0 is the time in which the JSON packet is sent from the gateway. t_1 is the time in which Helio receives the JSON packet and converts it from JSON to JSON-LD.

- Transaction time (here on in referred to as transaction): is the time that it takes to send the query from Helio to the triplestore. Note that this last operation does not finish until triplestore inserts and stores the data correctly. As shown in Section 4.3.2 the transaction time [10] is defined by:

  $$Transaction = t_2 - t_1$$

  where:

  - t_2 is the time in which Helio sends the JSON-LD packet to be stored in the triplestore and the triplestore responds with an acknowledgement of the insert operation.

4.4. Testbed and case study application

A genuine testbed is created to conduct performance analysis of four distinct scenarios, concentrating on the chosen application as the use case. In order to achieve this, the behaviour is modelled using entity Sensors. The model aids in stress-testing the system by releasing a large volume of JSON packages to identify the maximum number of packets that the system can support, specifically the fog node at the edge level.

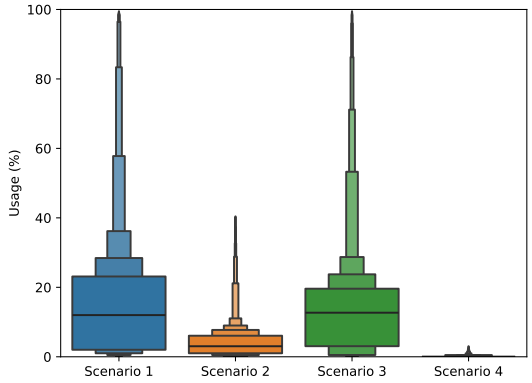
Hence, we have developed a shell script that transmits a uniform quantity of JSON objects every minute to achieve the highest number of packets that the fog node can handle. This is particularly significant in Scenario 4, where Helio and triplestore are deployed at the core level. Additionally, the identical stress test payload is sustained for 4 h, and we employ the same testbed and payload for all scenarios. The results presented in Section 5 indicate the mean of ten distinct testbeds conducted.

5. Performance evaluation

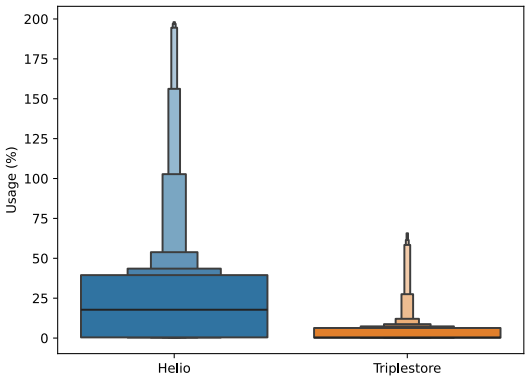
This section discusses the impact of computational resource consumption and latency time for the four scenarios. The results obtained from the various parameters in Section 4.3 are examined in detail. Results for a given parameter are grouped according to whether they were gathered at the edge level or core level for a better visual representation of the findings. We will subsequently include a discussion section that analyses the various findings obtained. This will provide users with recommendations for the efficient deployment of IoT architectures.

5.1. CPU

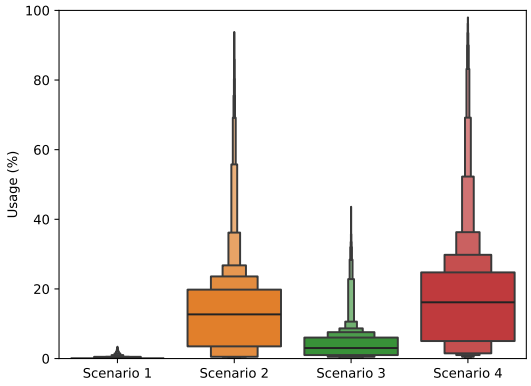
The initial evaluation parameter is the CPU usage in the scenarios, including both the edge and core levels (see Fig. 6). The CPU usage measurement is expressed as a percentage of usage (%) obtained from the stress test's results. Furthermore, benefiting from the deployment of Helio and the triplestore as Docker containers, we present the separate CPU consumption of each of them in order to assess the workload



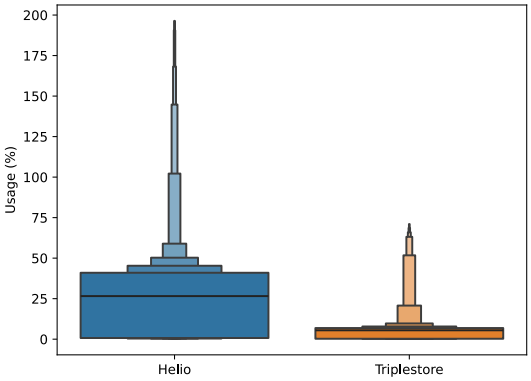
(a) All scenarios at the core level.



(b) Helio and the triplestore for Scenario 1 at the core level.



(c) All scenarios at the edge level.



(d) Helio and the triplestore for Scenario 4 at the edge level.

Fig. 6. CPU Usage (%) parameter in the stress test at both the core and edge levels.

of these two software engines independently. Note that the described scenarios and the corresponding data orchestration can be found in Section 4.

The analysis of CPU usage across different scenarios reveals distinct patterns at both the core and edge levels. At the core level, Scenario 1, incorporating both Helio and the triplestore, demonstrates the highest CPU consumption, often exceeding 30% and occasionally spiking to 80%. In contrast, Scenario 3, which solely utilises Helio, exhibits slightly lower but still significant CPU usage. Notably, data translation from JSON to JSON-LD accounts for the majority of CPU consumption, while triplestore usage remains minimal due to its data storage function. Scenarios 2 and 4, focusing on triplestore and gateway usage respectively, show comparatively lower CPU consumption, around 20%. Conversely, at the edge level, Scenarios 2 and 4 present the highest CPU consumption, suggesting efficient CPU utilisation with the deployment of Helio and triplestore. This strategy can alleviate CPU usage at the core level. Despite substantial CPU usage, a considerable portion remains unutilised, indicating the importance of CPU management in scenario deployment.

5.2. RAM

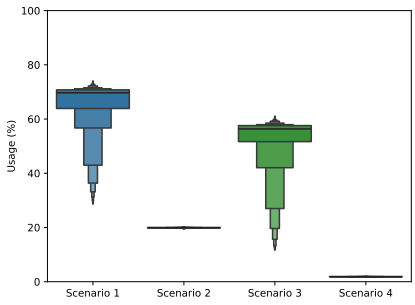
RAM consumption is a critical parameter at the edge level due to the limited resources microcomputers possess. It serves as a bottleneck parameter. During stress test design, the payload that fog nodes can withstand is taken into account to prevent collapsing of RAM and operating system crashes. Thus, Fig. 7 illustrates the evaluation of two parameters: RAM usage (in %) and the amount of memory (in KB) waiting to be written back to the disk, referred to as kbdirty. These parameters were evaluated at both the edge and core levels.

We have noted a comparably consistent trend to that previously examined for CPU usage. More specifically, the analysis of the scenarios reveals distinct patterns in RAM usage across different levels of the IoT architecture. Scenarios 1 and 3 demonstrate elevated RAM utilisation at the core level, while scenarios 2 and 4 exhibit high RAM consumption at the edge level, surpassing 70%. The primary contributor to RAM usage is Helio's JSON to JSON-LD data translation process, exceeding 50%. In contrast, the triplestore's RAM consumption remains low, as it primarily handles data waiting to be written onto the disk. While kbdirty, indicating pending disk writes, is not critical, RAM consumption reaching 90% at the edge level poses a risk of packet loss and compromises QoS. Implementing frameworks akin to Scenarios 2 and 4 necessitates careful consideration of WSN load to avert RAM overload and maintain service quality.

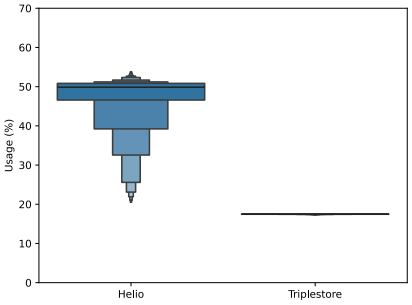
5.3. Hard disk

We analyse the TPS, KB_read and KB_write disk system parameters. Fig. 8 displays their consumption throughout the designed testbed. The scenarios with triplestore exhibit the highest load: scenarios 1 and 2 at the core level, and scenarios 3 and 4 at the edge level. Note that this behaviour closely resembles that which has been previously studied in kbdirty (see Figs. 7(c) and 7(f)). Therefore, kbdirty and disk parameters are highly interrelated and influenced by the triplestore.

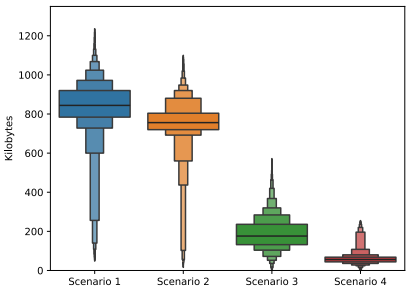
The deployment location of the triplestore significantly influences hard disk parameters at both the edge and core levels across all scenarios. Helio demonstrates minimal impact on these parameters. Transaction Per Second (TPS) ranges from 0–20 with occasional peaks of up to 70 transactions per second, indicating moderate to low load.



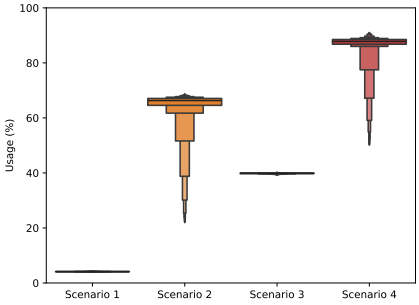
(a) RAM usage (%) for all scenarios at the core level.



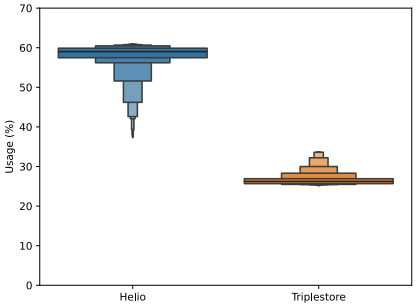
(b) RAM usage (%) for Helio and the triplestore for Scenario 1 at the core level.



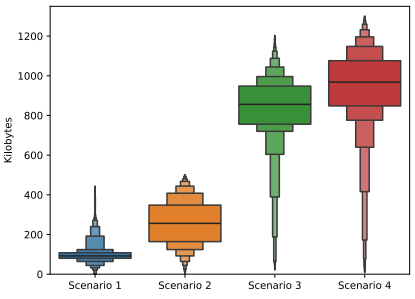
(c) kbdirty (Kilobytes) for all scenarios at the core level.



(d) RAM usage (%) for all scenarios at the edge level.

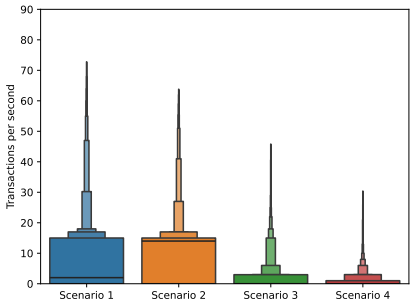


(e) RAM usage (%) for Helio and the triplestore for Scenario 4 at the edge level.

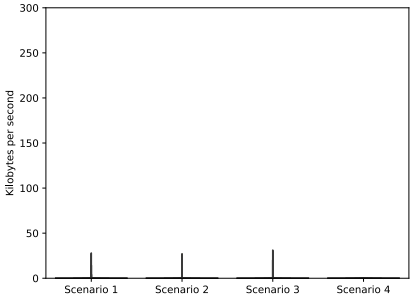


(f) kbdirty (Kilobytes) for all scenarios at the edge level.

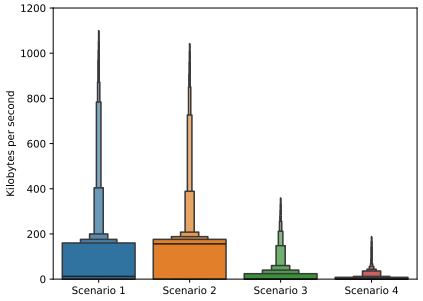
Fig. 7. RAM parameters in the stress test at both the core and edge levels.



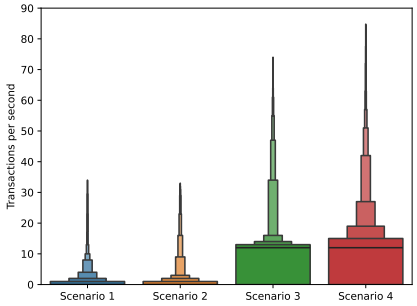
(a) Transfers per second (TPS) at the core level.



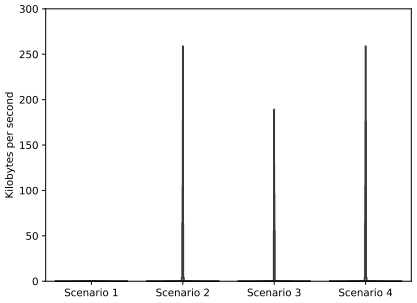
(b) Kilobytes read from the device per second (KB_read) at the core level.



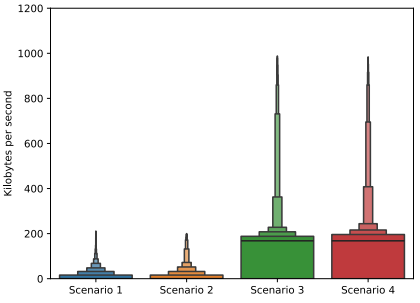
(c) Kilobytes write to the device per second (KB_write) at the core level.



(d) Transfers per second (TPS) at the edge level.

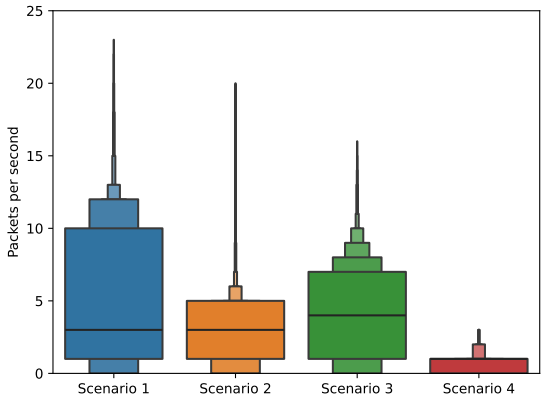


(e) Kilobytes read from the device per second (KB_read) at the edge level.

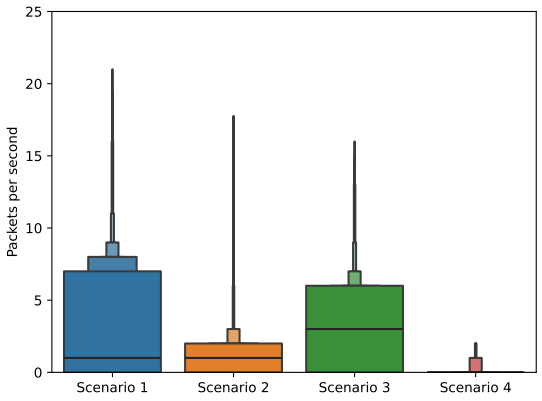


(f) Kilobytes write to the device per second (KB_write) at the edge level.

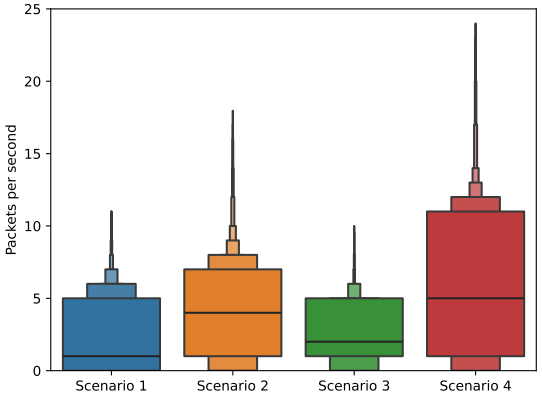
Fig. 8. Hard disk parameters in the stress test at both the core and edge levels.



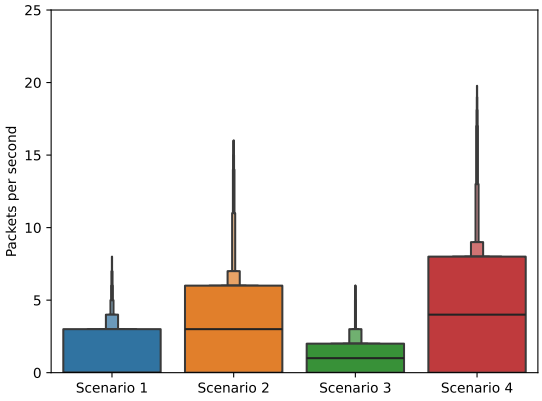
(a) Packets received per second (rxpck) at the core level.



(b) Packets transmitted per second (txpck) at the core level.



(c) Packets received per second (rxpck) at the edge level.



(d) Packets transmitted per second (txpck) at the edge level.

Fig. 9. Network parameters in the stress test at both the core and edge levels.

KB_read shows no significant load samples due to the stress test focusing on data transformation and disk write operations. Similarly, KB_write follows the TPS pattern, emphasising disk write operations. Overall, the triplestore is the primary contributor to hard disk load.

In conclusion, TPS and KB_write are not critical consumption parameters, suggesting that the triplestore can be implemented at the edge level without issues. Helio's minimal impact allows for normal deployment. Thus, the hard disk is not a critical system parameter in any of the studied scenarios.

5.4. Network

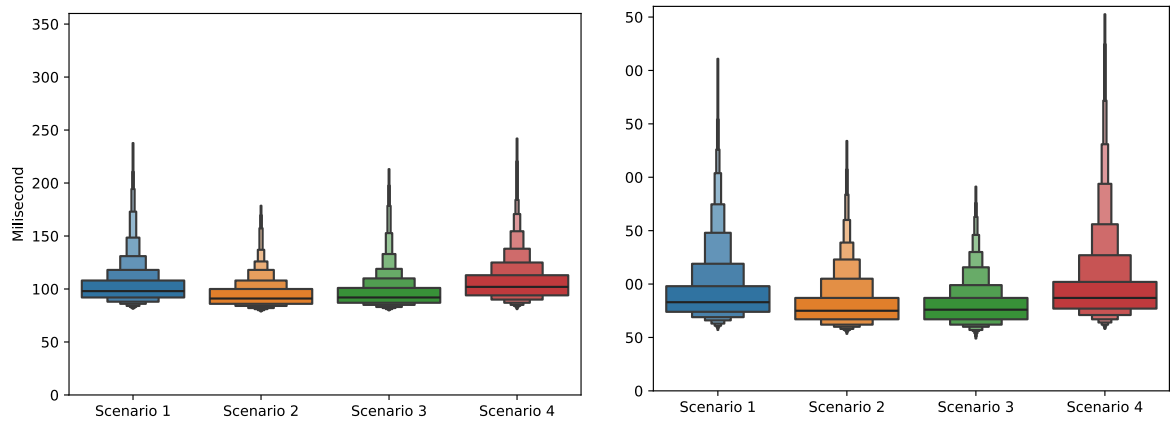
We investigate network usage across different scenarios at both the edge and core levels, as depicted in Fig. 9. The parameters being studied are rxpck and txpck, which signify received and transmitted packages correspondingly. In contrast to the hard disk parameter (see Fig. 8), we infer that network usage patterns are analogous to those of both the CPU (see Fig. 6) and RAM (see Fig. 7). Therefore, the highest level of consumption occurs in the deployment of Helio and triplestore. These Scenarios are 1 and 3 at the core level and Scenarios 2 and 4 at the edge level.

The examination of the rxpck and txpck behaviours reveals consistent patterns across all scenarios, both at the core and edge levels, indicating high similarity in data reception and transmission. While Helio's deployment slightly increases network consumption compared to the triplestore, notable distinctions emerge in Scenarios 1 and 4 regarding the absence of Helio and triplestore deployment between the edge and core levels. In Scenario 1, the fog node acts as a gateway forwarding data in JSON packets to the core level, whereas in Scenario 4, no computational work occurs due to the absence of deployed components and data reaching this level. Overall, Helio bears the most substantial workload in terms of the network parameter, primarily due to its role in data transformation before transmission to the triplestore. Despite this, the network parameter remains within manageable bounds, with transmitted and received packets averaging between 0-12 per second across scenarios.

5.5. Time latency

Finally, the latency time of the data translation and transaction process is examined (see Fig. 10). The latency time is divided into two main segments, which are detailed in Section 4.3.2. The initial stage is the translation process from JSON to JSON-LD (see Fig. 10(a)). The subsequent phase involves the transaction process wherein the transformed data is retained in the triplestore and a acknowledgement sent to the triplestore (see Fig. 10(b)). Note that the calculation of latency time differs for each situation and does not segregate based on edge and core level.

In analysing the scenarios, it is evident that the latency time for both data translation and transaction processes remains consistent across all scenarios, with minimal differences observed between them. Notably, Scenarios 1 and 4 exhibit higher latency times compared to Scenarios 2 and 3. Scenario 1, where both Helio and the triplestore are located at the core level, and Scenario 4, where both are at the edge level, demonstrate elevated latency times. Conversely, in Scenarios 2 and 3, where Helio and the triplestore are deployed at different levels (edge



(a) Translation time (miliseconds). (b) Transaction time (miliseconds).

Fig. 10. Time latency parameters in the stress test for the four scenarios.

Table 2
Definition and representation of performance parameters at the core level.

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
CPU	Stable	not critical	Stable	Not critical
RAM	Stable	Not critical	Stable	Non-critical
kbdirty	Stable	Stable	Non-critical	Non-critical
TPS	Stable	Stable	Non-critical	Non-critical
KB_read	Non-critical	Non-critical	Non-critical	Non-critical
KB_write	Stable	Stable	Non-critical	Non-critical
rxpck	Stable	Non-critical	Stable	Non-critical
txpck	Stable	Non-critical	Stable	Non-critical
Translation	Stable	Stable	Stable	Stable
Transaction	Stable	Stable	Stable	Stable

Table 3
Definition and representation of performance parameters at the edge level.

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
CPU	Non-critical	Critical	Non-critical	Critical
RAM	Non-critical	Critical	Stable	Critical
kbdirty	Non-critical	Non-critical	Stable	Stable
TPS	Non-critical	Non-critical	Stable	Stable
KB_read	Non-critical	Non-critical	Non-critical	Non-critical
KB_write	Non-critical	Non-critical	Stable	Stable
rxpck	Stable	Stable	Stable	Stable
txpck	Stable	Stable	Stable	Stable

and core, respectively), latency times are notably reduced. The computational load, particularly concerning CPU and RAM utilisation, directly influences latency times, showcasing that distributing Helio and the triplestore between edge and core levels can optimise performance.

Overall, the slight differences in latency times between scenarios, primarily attributed to the payload carried by fog nodes and network conditions, are negligible. However, transaction times exhibit greater instability and fluctuations, with peaks up to 300 ms observed in Scenario 4, while translation times remain stable and consistent with minimal variance throughout the stress test. This indicates the importance of optimising computational load distribution between edge and core levels to minimise latency and ensure stable performance in IoT architectures.

5.6. Discussion

In light of the aforementioned results and to provide informative findings that can be applied to similar scenarios, Tables 2 and 3 outline the main outcomes of the evaluation conducted at the core and edge levels, respectively [30]. The assessment is categorised according to three levels of criticality during the deployment of each scenario. A parameter is deemed non-critical when it has minimal impact, critical when it directly affects service quality in the architecture and stable when although not critical, it warrants consideration when deploying components in one of the IoT architecture's layers.

The comparative analysis of computational resources in real-time IoT applications based on semantic interoperability, transitioning from cloud and fog computing to federated-fog computing, presents critical insights into resource allocation and system performance in the context of compute continuum. However, the study has certain limitations that must be acknowledged.

Firstly, while the study comprehensively evaluates various IoT architectures, it does not account for network latency since all experiments have been carried out following an approach for having 0 time latency. In real scenarios, such latency may affect the results, and further research is necessary to incorporate network latency in the evaluation.

Secondly, the emphasis on CPU and RAM considerations may overlook other critical aspects of resource management, potentially limiting the scope of optimisation strategies. It is crucial to explore additional resource management aspects to provide a more holistic view of IoT architecture optimisation.

Thirdly, the study's focus on specific scenarios and architectures based on semantic interoperability may restrict its applicability to broader IoT deployment contexts. To validate the findings across diverse scenarios and environments, further research is required.

Lastly, although the study highlights the advantages of novel architectures like inverted- and federated-fog computing, it may not fully address potential challenges or limitations associated with their implementation in real-world settings such as in [11]. It is essential to investigate these challenges to ensure the successful deployment of these architectures in practice.

Despite these limitations, the study offers valuable insights into the deployment of IoT architectures in the context of compute continuum. At the core level, characterised by abundant computational resources, no critical issues are observed across all scenarios. However, the edge level, where fog nodes operate with constrained resources, necessitates meticulous software deployment assessment. Parameters like RAM, and to a lesser extent CPU, emerge as pivotal factors, notably observed in Scenarios 2 and 4, involving the Helio engine.

Data translation imposes substantial computational costs, particularly concerning CPU and RAM at the edge level, where resource limitations are pronounced. Conversely, the deployment of the data storage system exhibits minimal CPU and RAM consumption, facilitating efficient deployment of the triplestore at the edge level while

maintaining a high QoS. Efficient deployment of the triplestore and Helio at the edge level optimises resource utilisation, contingent upon manageable fog node data loads. This strategic deployment strategy liberates core-level resources for additional tasks, such as data analysis and processing, thereby enhancing overall system efficiency.

Furthermore, network and hard disk load parameters demonstrate stability and non-critical behaviour at the edge level across all scenarios. Notably, latency times for translation and transaction remain consistent regardless of the IoT architecture employed, underscoring consistent QoS. Scenarios 3 and 4 emerge as particularly promising, leveraging fog nodes' computational prowess to efficiently allocate resources within IoT architectures. By alleviating burdens on translation and storage at the core level, these scenarios enable federated machine learning through geographic data distribution, empowering users to selectively disclose information based on predefined criteria. A meticulous evaluation of CPU and RAM at the edge level, considering variations in network load within IoT architecture, is imperative. Nevertheless, latency time, network, and hard disk considerations exert minimal influence on deployment decisions.

In summary, the application of Scenarios 3 and 4 in IoT infrastructure offers viable solutions in terms of both latency time and computational resource consumption, highlighting the significance of strategic resource allocation and performance optimisation in IoT architecture design, particularly in the context of compute continuum. Regarding technology transfer, our research offers practical implications for companies and cities seeking to implement IoT architectures for various applications. By optimising resource allocation and performance metrics across different levels of IoT deployment, our findings can inform the design and implementation of scalable, efficient, and reliable IoT systems. For companies, this means leveraging our insights to enhance the development and deployment of IoT solutions tailored to specific industry needs, such as healthcare, manufacturing, or smart cities. Cities can utilise our research to build resilient and adaptable infrastructure for monitoring and managing urban services, promoting sustainability, efficiency, and citizen well-being. By integrating our strategies for resource optimisation and scalability, businesses and municipalities can realise the full potential of IoT technologies in improving operational efficiency, enhancing service delivery, and driving innovation in diverse domains.

6. Conclusions and open challenges

The following work presents a thorough examination and appraisal of the implementation of various IoT architectures and their evaluation. To accomplish this, we execute the assessment and analysis strategy by generating and storing data. Throughout this process, two primary software engines are employed: a data translation engine called Helio, and a data storage engine known as the triplestore. The study presents four scenarios analysing the four primary architectures for IoT deployment: cloud computing, fog computing, and the two novel architectures of inverted- and federated-fog computing. The primary variation among the four architectures lies in the efficient allocation of Helio and the triplestore at either the edge or core level.

A stress test was devised for comparative analysis, monitoring the computational consumption parameters and latency time using a stress test. Our results demonstrate that adopting a controlled deployment strategy for Helio and the triplestore at the edge level can liberate substantial resources at the core level. Importantly, this does not compromise the QoS or latency time. Similarly, it is suggested that CPU and RAM are vital when developing a resource decentralisation plan, as seen in the inverted- and federated-fog computing designs. Moreover, the geographically dispersed data in these architectures makes federated-machine learning efficient.

In terms of future work, the authors of this study find it essential to scrutinise the same procedure applied to data reading. To meet the objective, a SPARQL-based federated data reading system should be developed. The novel inverted and federated-fog computing architectures would benefit from a secure and encrypted geographically distributed data sharing system. It is crucial to evaluate the workload at both the edge and core levels while performing federated machine learning. Moreover, we aim to compare sharding and federation techniques in federated IoT architectures, examining their effectiveness for data management and scalability. Our goal is to provide insights into selecting optimal strategies for enhancing performance in federated IoT systems. In addition, the impact of including offloading in the experiment scenarios will be analysed in following experiments.

CRediT authorship contribution statement

Edgar Huaranga-Junco: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Salvador González-Gerpe:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Manuel Castillo-Cara:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Andrea Cimmino:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Raúl García-Castro:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Raúl García-Castro reports financial support was provided by European Commission. Manuel Castillo-Cara reports travel was provided by Ibero-American Program of Science and Technology for Development. Raúl García-Castro reports financial support was provided by Madrid government. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

This scientific experiment contains the extended data and source code available in Zenodo [43] and GitHub [44].

Acknowledgements

The research leading to these results has been funded by the European Union's Horizon 2020 Research and Innovation Programme through the AURORAL project, Grant Agreement No. 101016854; by the CYTED, Grant No. 520rt0011; by the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with the Universidad Politécnica de Madrid in the Excellence Programme for University Teaching Staff, in the context of the V PRICIT (Regional Programme of Research and Technological Innovation); and by UNED funding for open access publishing.

Appendix. Helio mapping

The Helio mapping developed for the experimental tests is shown below. Remember that the mapping does the translation from JSON to JSON-LD 1.1 and, in addition, then performs the transaction process to the triplestore.

mapping

```
<#assign start0 = .now>
<#assign jpath=handlers"JsonHandler">
<#assign unit_map = '{"humidity":"relativeHumidityUnit", "pressure": \
  "hectopascal", "temperature":"degreeCelcius", "sound":"decibel", \
  "luminosity":"lux","co2":"partsPerMillion", "uv":"wattPerSquareMetre", \
  "o3":"partsPerMillion", "pm2_5":"partsPerMillion", \
  "pm10":"partsPerMillion"}' >
<#assign property_map = '{"humidity": "relativeHumidity", "pressure": \
  "atmosphoricPressure", "temperature":"ambientTemperature", "sound": \
  "noiseLevel", "luminosity":"illuminance","co2":"cO2Concentration", \
  "uv":"uVIndex","o3":"o3Concentration", "pm2_5":"pM2.5Concentration", \
  "pm10":"pM10Concentration"}' >
<#assign datax = sensor_data?eval>

<#assign rdf>
{
   "@context":"https://raw.githubusercontent.com/edgarhuaranga/ \
    test-mapper/main/foggy-context2.json",
   "measures":[
      <#list datax.data as row>
         <#assign name =row.parameter>
         <#assign timestamp =row.timestamp?c?number>
         <#assign value = row.value>
      {
               "@type": "saref:Measurement",
               "id": "urn:uuid:upm:sensor:[=name]: \
                 [=timestamp?replace",",""]",
               "timestamp":"[=timestamp?number_to_datetime? \
                iso"Europe/Rome"]",
               "value": "[=value?replace',','']",
               "relates": "[=jpath.filtername, property_map]",
               "measuredIn": "[=jpath.filtername, unit_map]"
           }
           <#if row?is_last><#else>,</#if>
      </#list>
   ]
}
</#assign>

<#assign config = {"data-format" : "json-ld", "output-format" : "nt"}>
<@action type="RdfCast" data=rdf conf=config; result>
<#assign rdf = result>
</@action>

<#assign start1 = .now>

<#assign body="INSERT DATA { GRAPH <https://ts.fogger3.linkeddata.es/abced> \
  { " +rdf+ " } }">
<#assign graph="https://ts.fogger3.linkeddata.es/repositories/data-cloud/ \
  statements">
<#assign dataset>
{
         "method": "POST",
         "url": "[=graph]",
         "body": "[=body?js_string]",
         "headers" : {  "Content-Type" : "application/sparql-update" }
}
</#assign>

<#assign result = providers"HttpProvider",dataset>
[=result]

<#assign start2 = .now>
<#-- Inicio, TiempoFinTransformacion, TiempoFinTransaccion -->
[=start0?iso_utc_ms], [=start1?iso_utc_ms], [=start2?iso_utc_ms]
```

References

[1] M. El-Hosseini, H. ZainEldin, H. Arafat, M. Badawy, A fire detection model based on power-aware scheduling for IoT-sensors in smart cities with partial coverage, J. Ambient Intell. Humaniz. Comput. 12 (2) (2021) 2629–2648.

[2] A. Hammoud, M. Kantardjian, A. Najjar, A. Mourad, H. Otrok, Z. Dziong, N. Guizani, Dynamic fog federation scheme for internet of vehicles, IEEE Trans. Netw. Serv. Manag. (2022).

[3] J. Rosa-Bilbao, J. Boubeta-Puig, A. Rutle, CEPEDALoCo: An event-driven architecture for integrating complex event processing and blockchain through low-code, Internet Things 22 (2023) 100802, http://dx.doi.org/10.1016/j.iot.2023.100802.

[4] O. Gómez-Carmona, D. Buján-Carballal, D. Casado-Mansilla, D.L. de Ipiña, J. Cano-Benito, A. Cimmino, M. Poveda-Villalón, R. García-Castro, J. Almela-Miralles, D. Apostolidis, A. Drosou, D. Tzovaras, M. Wagner, M. Guadalupe-Rodriguez, D. Salinas, D. Esteller, M. Riera-Rovira, A. González, J. Clavijo-Ágreda, A. Díez-Frias, M. del Carmen Bocanegra-Yáñez, R. Pedro-Henriques, E. Ferreira-Nunes, M. Lux, N. Bujalkova, Mind the gap: The AURORAL ecosystem for the digital transformation of smart communities and rural areas, Technol. Soc. 74 (2023) 102304, http://dx.doi.org/10.1016/j.techsoc.2023.102304.

[5] V.F. Rodrigues, R. da Rosa Righi, C.A. da Costa, F.A. Zeiser, B. Eskofier, A. Maier, D. Kim, Digital health in smart cities: Rethinking the remote health monitoring architecture on combining edge, fog, and cloud, Health Technol. (2023) 1–24.

[6] A.M. Albaseer, M. Abdallah, A. Al-Fuqaha, A. Erbad, Fine-grained data selection for improved energy efficiency of federated edge learning, IEEE Trans. Netw. Sci. Eng. (2021).

[7] M. Vera-Panez, K. Cuadros-Claro, M. Castillo-Cara, L. Orozco-Barbosa, BeeGOns!: A wireless sensor node for fog-computing in smart city applications, IEEE Trans.

Comput.-Aided Des. Integr. Circuits Syst. (2023) 1–5, http://dx.doi.org/10.1109/TCAD.2023.3305575.

[8] G. Ortiz, J. Boubeta-Puig, J. Criado, D. Corral-Plaza, A. Garcia-de Prado, I. Medina-Bulo, L. Iribarne, A microservice architecture for real-time IoT data processing: A reusable web of things approach for smart ports, Comput. Stand. Interfaces 81 (2022) 103604.

[9] B. Negash, T. Westerlund, H. Tenhunen, Towards an interoperable Internet of Things through a web of virtual things at the Fog layer, Future Gener. Comput. Syst. 91 (2019) 96–107.

[10] G. Mondragón-Ruiz, A. Tenorio-Trigoso, M. Castillo-Cara, B. Caminero, C. Carrión, An experimental study of fog and cloud computing in CEP-based Real-Time IoT applications, J. Cloud Comput. 10 (1) (2021) 1–17, http://dx.doi.org/10.1186/s13677-021-00245-7.

[11] M. Diamanti, P. Charatsaris, E.E. Tsiropoulou, S. Papavassiliou, Incentive mechanism and resource allocation for edge-fog networks driven by multi-dimensional contract and game theories, IEEE Open J. Commun. Soc. 3 (2022) 435–452, http://dx.doi.org/10.1109/OJCOMS.2022.3154536.

[12] M. Suárez-Albela, T.M. Fernández-Caramés, P. Fraga-Lamas, L. Castedo, A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications, Sensors 17 (9) (2017) http://dx.doi.org/10.3390/s17091978.

[13] S.M. Rajagopal, M. Supriya, R. Buyya, FedSDM: Federated learning based smart decision making module for ECG data in IoT integrated Edge-Fog-Cloud computing environments, Internet Things (2023) 100784.

[14] J. Boubeta-Puig, J. Rosa-Bilbao, J. Mendling, CEPchain: A graphical model-driven solution for integrating complex event processing and blockchain, Expert Syst. Appl. 184 (2021) 115578.

[15] Y. Liu, Y. Dong, H. Wang, H. Jiang, Q. Xu, Distributed fog computing and federated-learning-enabled secure aggregation for IoT devices, IEEE Internet Things J. 9 (21) (2022) 21025–21037.

[16] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330, http://dx.doi.org/10.1016/j.sysarc.2019.02.009.

[17] F. Burzlaff, N. Wilken, C. Bartelt, H. Stuckenschmidt, Semantic interoperability methods for smart service systems: A survey, IEEE Trans. Eng. Manage. (2019).

[18] Z. Rejiba, X. Masip-Bruin, E. Marín-Tordera, Towards user-centric, switching cost-aware fog node selection strategies, Future Gener. Comput. Syst. 117 (2021) 359–368, http://dx.doi.org/10.1016/j.future.2020.12.006.

[19] C. Anglano, M. Canonico, P. Castagno, M. Guazzone, M. Sereno, A game-theoretic approach to coalition formation in fog provider federations, in: 2018 Third International Conference on Fog and Mobile Edge Computing, FMEC, 2018, pp. 123–130, http://dx.doi.org/10.1109/FMEC.2018.8364054.

[20] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for internet of things: Recent advances, taxonomy, and open challenges, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1759–1799.

[21] A. Hazra, M. Adhikari, S. Nandy, K. Doulani, V.G. Menon, Federated-learning-aided next-generation edge networks for intelligent services, IEEE Netw. 36 (3) (2022) 56–64.

[22] M. Serrano, A. Gyrard, M. Boniface, P. Grace, N. Georgantas, R. Agarwal, P. Barnagui, F. Carrez, B. Almeida, T. Teixeira, et al., Cross-domain interoperability using federated interoperable semantic IoT/Cloud testbeds and applications: The FIESTA-IoT approach, in: Building the Future Internet Through FIRE, River Publishers, 2022, pp. 287–321.

[23] P.L.d.L. de Souza, W.L.d.L. de Souza, R.R. Ciferri, Semantic interoperability in the Internet of Things: A systematic literature review, in: S. Latifi (Ed.), ITNG 2022 19th International Conference on Information Technology-New Generations, Springer International Publishing, Cham, 2022, pp. 333–340.

[24] H. Rahman, M.I. Hussain, A comprehensive survey on semantic interoperability for Internet of Things: State-of-the-art and research challenges, Trans. Emerg. Telecommun. Technol. 31 (12) (2020) e3902.

[25] B. Hammi, R. Khatoun, S. Zeadally, A. Fayad, L. Khoukhi, IoT technologies for smart cities, IET Netw. 7 (1) (2018) 1–13.

[26] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, S. Guizani, Internet-of-Things-based smart cities: Recent advances and challenges, IEEE Commun. Mag. 55 (9) (2017) 16–24.

[27] A.M.S. Osman, A novel big data analytics framework for smart cities, Future Gener. Comput. Syst. 91 (2019) 620–633, http://dx.doi.org/10.1016/j.future.2018.06.046.

[28] C. Harrison, B. Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, P. Williams, Foundations for smarter cities, IBM J. Res. Dev. 54 (4) (2010) 1–16.

[29] P. Boobalan, S.P. Ramu, Q.V. Pham, K. Dev, S. Pandya, P.K.R. Maddikunta, T.R. Gadekallu, T. Huynh-The, Fusion of federated learning and industrial internet of things: A survey, Comput. Netw. 212 (2022) 109048.

[30] A. Tenorio-Trigoso, M. Castillo-Cara, G. Mondragón-Ruiz, C. Carrión, B. Caminero, An analysis of computational resources of event-driven streaming data flow for Internet of Things: A case study, Comput. J. (2022) http://dx.doi.org/10.1093/comjnl/bxab143.

[31] S. Sengupta, J. Garcia, X. Masip-Bruin, A literature survey on ontology of different computing platforms in smart environments, 2018, http://dx.doi.org/10.48550/ARXIV.1803.00087.

[32] R. Studer, V.R. Benjamins, D. Fensel, Knowledge engineering: Principles and methods, Data Knowl. Eng. 25 (1–2) (1998) 161–197.

[33] L. Feigenbaum, G. Todd-Williams, K. Grant-Clark, E. Torres, SPARQL 1.1 protocol, 2013, W3C Recommendation.

[34] S. Godard, SYSSTAT software, 2023, URL: http://sebastien.godard.pagesperso-orange.fr/. (Online: Accessed 18 October 2023).

[35] A. Cimmino, R. García-Castro, Helio software, 2024, URL: https://github.com/helio-ecosystem. (Online: Accessed 30 January 2024).

[36] A. Cimmino, R. García-Castro, Helio: a framework for implementing the life cycle of knowledge graphs, Semant. Web (2022) 1–27.

[37] World Wide Web Consortium and others, RDF 1.1 Concepts and Abstract Syntax, World Wide Web Consortium, 2014.

[38] A. Pereira, A. Trifan, R.P. Lopes, J.L. Oliveira, Systematic review of question answering over knowledge bases, IET Softw. 16 (1) (2022) 1–13.

[39] G. Sharma, V. Tripathi, V. Singh, A systematic analysis of trending NOSQL database tools and techniques: A survey, in: AIP Conference Proceedings, 2782, (1) AIP Publishing, 2023.

[40] A.M. Ouksel, A. Sheth, Semantic interoperability in global information systems, ACM Sigmod Rec. 28 (1) (1999) 5–12.

[41] S. Suwanmanee, D. Benslimane, P. Thiran, OWL-based approach for semantic interoperability, in: 19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA Papers), 2005, pp. 145–150, http://dx.doi.org/10.1109/AINA.2005.271.

[42] Smart Appliances, Smartm2m; smart appliances; reference ontology and onem2m mapping, 2017, RTS/SmartM2M-103264v2, Rev 2.

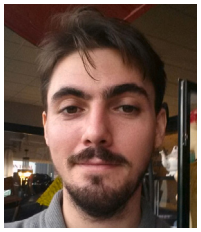
[43] E. Huaranga-Junco, S. González-Gerpe, M. Castillo-Cara, A. Cimmino, R. García-Castro, manwestc/FOGES: FOGES v0.0.1, Zenodo, 2024, http://dx.doi.org/10.5281/zenodo.10669669.

[44] E. Huaranga-Junco, S. González-Gerpe, M. Castillo-Cara, A. Cimmino, R. García-Castro, From cloud and fog computing to federated-fog computing: A comparative analysis of computational resources in real-time Iot applications based on semantic interoperability, 2024, URL: https://github.com/manwestc/FOGES.

Edgar Huaranga-Junco received his MSc on Artificial Intelligence from the Universidad Politécnica de Madrid in July 2023. He has been working as teacher and research assistant at Universidad de Lima (Peru). His research includes Wireless Sensor Networks and applied Artificial Intelligence.

Salvador González-Gerpe is a researcher in Artificial Intelligence with extensive knowledge focused on the semantic characterisation of Digital Twins. Graduated in computer engineering with a master's degree in artificial intelligence. Currently, working on a PhD in Artificial Intelligence, with the main focus on semantic characterisation and interoperability between different Digital Twins.

Manuel Castillo-Cara received the Ph.D. degree from the Universidad de Castilla-La Mancha in July 2018. He has been working on university educational issues at the Computer Science as an Assistant Professor at Universidad Nacional de Educación a Distancia (Spain). His current research is focused on Intelligent Ubiquitous Technologies, especially on in Distributed Computing, Pattern Recognition, Artificial Intelligence and Indoor localisation.



Andrea Cimmino received the computer science degree and the PhD degree in Software Engineering at the Universidad de Sevilla (US). He is currently an Associate Professor at Universidad Politécnica de Madrid with the Computer Science Department, UPM. His research activities focus on semantic web, semantic interoperability, data integration, and IoT discovery.

Raúl García-Castro is Associate Professor at the Computer Science School of the Universidad Politécnica de Madrid (UPM), Spain. In 2008 he obtained a Ph.D. in Computer Science and Artificial Intelligence at UPM, which obtained the Ph.D. Extraordinary Award. His research focusses on ontological engineering, semantic interoperability, and ontology-based data and application integration. He regularly participates in standardisation bodies and in the programme committees of conferences and workshops that are most relevant in his field, having also organised several international conferences and workshops.