A Scalable Computing Continuum Framework for Ambient Assisted Living

Jahedul Anowar, Koushikur Islam, Kanaka Sai Jagarlamudi, Bahman Javadi, Rodrigo N. Calheiros
School of Computer, Data and Mathematical Sciences
Western Sydney University
Sydney, Australia
22101869@student.westernsydney.edu.au, {k.shohag, k.jagarlamudi, b.javadi, r.calheiros}@westernsydney.edu.au

***Abstract*—This paper presents a scalable computing-continuum framework for Ambient Assisted Living (AAL)–based IoT applications. It incorporates diverse Internet of Things (IoT) data sources, including environmental sensors, facial-recognition-enabled indoor drones, depth cameras, and monitoring robots. These devices operate across a continuum comprising edge devices, an edge server, and a cloud layer. The framework is evaluated through a Large Language Model (LLM)-based web application that accepts natural-language queries and retrieves data from IoT sources connected to the continuum. Experimental results demonstrate that latency-sensitive and privacy-critical data is stored and processed at the edge, while computation-intensive or non-sensitive data are offloaded to cloud. This Quality of Service (QoS)-aware data placement and query-routing strategy enables scalable operation across heterogeneous compute units. This approach sustains throughput under concurrent workloads while delivering near real-time responses. Evaluation results for data retrieval and delivery for up to 150 simultaneous queries show end-to-end response times of under 3.5 seconds at the edge layer and under 10 seconds at the cloud layer, demonstrating the framework's scalability.**

***Index Terms*—Computing Continuum, Ambient Assisted Living, Quality of Service, Internet of Things**

I. INTRODUCTION TO THE PROBLEM

Effective utilisation of IoT data by applications depends on adaptable data acquisition, processing, and dissemination pipelines that meet application requirements [1]. Some applications may require various low-latency data for time-critical tasks, such as alerting staff to patient falls in aged-care facilities. A few others may need compute-intensive processing, for example, when they rely on thermal or point-cloud data. IoT data may also contain sensitive or personal information that should not be disclosed to third parties. These diverse application requirements and data characteristics call for an adaptive processing paradigm that can maintain latency and accuracy while preserving individuals' privacy.

Processing workloads are distributed across edge, fog, and cloud resources in computing continuum architectures [2], offering a promising solution to address the scalability, latency, and privacy challenges arising from the diverse characteristics of IoT data [3]. Existing continuum approaches lack intelligent data-routing mechanisms to select the most suitable processing layer, thereby limiting their ability to simultaneously maintain latency, accuracy, and privacy under dynamically varying IoT data and application requirements. This paper presents a computing continuum framework that incorporates an intelligent scaling mechanism to distribute the processing loads to the most suitable resources to improve these metrics.

This paper is organised as follows. Section II introduces the proposed computing continuum framework's application domain and motivates its scalability requirements. Section III describes the framework's architecture and implementation. Section IV compares the framework with related work. Section V evaluates its scalability under concurrent workloads and routing mechanism. Sections VI and VII discuss the broader impact, future research directions, and concludes the paper.

II. MOTIVATION: COMPUTING CONTINUUM FRAMEWORK FOR AMBIENT ASSISTED LIVING

Assisted living environments are experiencing a global healthcare worker shortage. According to World Health Organisation (WHO), there is a 5.8 million nursing staff shortage as of 2023 [4]. Internet of Things (IoT) systems can play a vital role in countering this shortage [5]. IoT generate significant data that can be presented to available care workers—as insights and alerts—to make informed decisions.

Generating alerts in ambient assisted living environments (AAL) requires low-latency, privacy-sensitive IoT data acquisition and processing, imposing strict constraints on data sharing and processing locations. Monitoring in AAL settings also demands substantial computational resources, particularly when processing thermal or vision-based data. This diversity in data characteristics, data volumes, and queries require appropriate processing resources and intelligent routing mechanisms capable of scaling with workload intensity to meet application Quality of Service (QoS) requirements. To address these scalability, latency, and privacy challenges, the proposed computing continuum framework distributes data storage and processing across edge and cloud layers in a QoS-aware manner, thereby enabling scalable operation under low latency while preserving privacy.

III. ARCHITECTURE AND IMPLEMENTATION OF THE PROPOSED COMPUTING CONTINUUM FRAMEWORK

The proposed computing continuum framework is designed and implemented to scale and support the dynamically varying IoT data and application requirements in AAL environments. A typical computing continuum framework is organised into

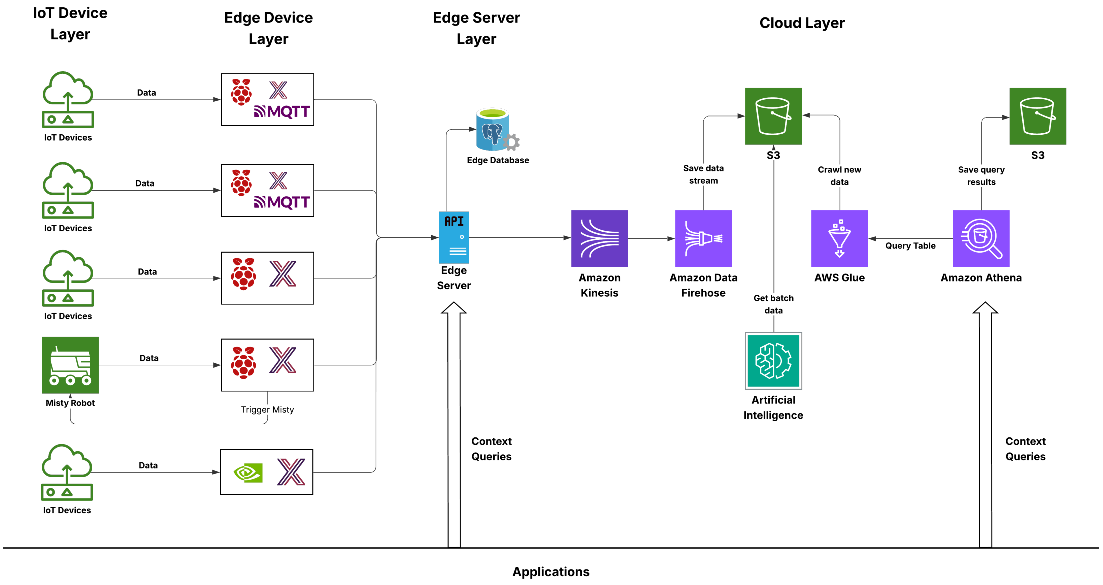


Fig. 1. Architectural overview of our proposed computing continuum framework.

three layers: data endpoints, edge, and cloud layers [6]. The proposed framework extends such traditional models by extending the resource spectrum to achieve higher scalability. As Fig. 1 depicts, its architecture comprises four layers.

Unlike traditional models that employ a single edge layer, the edge tier in the proposed framework is divided into two layers: (i) the edge device layer and (ii) the edge server layer. This separation enhances scalability and availability by facilitating data processing at different edge resources. The four layers in the proposed computing continuum framework are: (1) IoT device layer, (2) edge device layer, (3) edge server layer, and (4) cloud layer.

A. IoT Device Layer

The proposed framework is designed to integrate a wide range of IoT devices, varying in terms of communication technologies and protocols. The framework supports data acquisition over MQTT, REST, and WebSockets, which are popular communication protocols used in IoT devices. It also supports data transmission over various communication technologies, such as Wi-Fi, Zigbee, LAN, and Bluetooth. These communication protocols and channels ensures that the framework supports diverse IoT devices and ensures interoperability.

TABLE I
IoT DEVICES USED IN THE IMPLEMENTED FRAMEWORK AND THE DATA THEY GENERATE.

IoT Device	Data Attributes Generated
Door and Window Sensor	Binary state data (open/closed), timestamps, and basic device health information.
Temperature and Humidity Sensor	Environmental telemetry including temperature, humidity, pressure, and timestamps.
Misty Robot II	Human-centric perceptual data such as face recognition outputs, pose information, proximity sensing, and timestamps.
DJI Tello Drone	Visual and telemetry data including video streams, object detection outputs, positional telemetry, and timestamps.
ZED 2i Stereo Camera	Visual and spatial data including object detections, depth measurements, point clouds, and occupancy information.

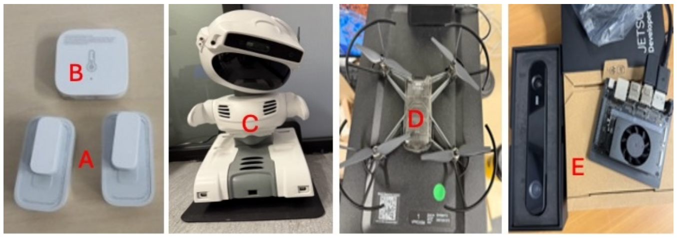


Fig. 2. IoT devices involved in the implementation.

Table I summarises the IoT devices incorporated in the framework implementation, while Fig. 2 provides a visual depiction of these deployed sensors and devices. Door and window[1] sensors (A in Fig. 2) generate textual data indicating the operational status (open/closed) of doors and windows, along with device-related attributes. Environmental[2] sensors (B in Fig. 2) produce textual telemetry data capturing temperature, humidity, and pressure information in the room. Misty Robot II[3] (C in Fig. 2) generates human-centric perception results, including facial recognition outputs to detect intruders. The DJI Tello[4] drone (D in Fig. 2) produces visual and telemetry data, including video streams, object detection outputs,

---

[1] https://www.aqarastore.com.au/products/door-and-window-sensor

[2] https://www.aqara.com/en/product/temperature-humidity-sensor

[3] https://www.mistyrobotics.com/misty-ii

[4] https://www.d1store.com.au/categories/tello-drone

and flight-related measurements. The ZED 2i[5] stereo camera (E in Fig. 2) generates visual outputs comprising multiple visual data types, including object recognition and spatial information. These data streams support evaluation of the framework's performance under distinct data characteristics and processing conditions (see Section V for details).

B. Edge Device Layer

The edge device layer collects, preprocesses, filters, and disseminates data generated by IoT devices. The edge devices in the implemented framework include Raspberry Pi 4s[6] and NVIDIA Jetson Orin Nanos [7]. This layer enforces a unified data schema through the EdgeX [8] framework, which is deployed on all edge devices. Through EdgeX, the framework achieves seamless data transmission across all layers.

The edge devices collect IoT data using multiple communication protocols, including ZigBee, MQTT, WebSockets, and REST. These data streams are filtered and normalised into a unified schema using the EdgeX Core Data API [7]. Then they are locally pre-processed and forwarded to higher layers. Communication among edge devices within this layer is facilitated using REST-based interfaces.

The edge device layer is designed to support collaborative operation among multiple edge devices, thereby eliminating a single point of failure and improving system resilience. Additionally, the layer enforces privacy by performing data pre-processing close to the data source, reducing the exposure of raw sensitive data beyond the local environment.

C. Edge Server Layer

The edge server layer processes the filtered data received from the edge device layer and disseminates it to the cloud-layer or applications. This layer is implemented using a high-performance computing node to ingest large volumes of real-time IoT data from the edge device layer. It incorporates a Flask[9] based Application Programming Interface (API) to facilitate reliable data exchange with edge devices, the cloud layer, and applications.

Sensor data aggregation and intelligent data distribution across persistent storage systems are the edge server layer's key functionalities. Privacy-preserving data storage in this layer is ensured through a NoSQL database (MongoDB[10]), which provides scalable query performance suitable for storing vast IoT data streams. PrestoDB[11] query engine is used to query this NoSQL database using standard SQL queries.

In addition to locally storing privacy-sensitive sensor data, the edge server layer's intelligent distribution mechanism selectively forwards data streams to the cloud layer and applications. The data transfer to the cloud layer is performed for workloads that demand substantial computational resources but can tolerate higher latency and relaxed privacy constraints.

D. Cloud Layer

The framework leverages scalable and highly available cloud resources to support data streams that require intensive processing. The framework is cloud-platform agnostic and can be integrated with any major cloud provider; current implementation incorporates Amazon Web Services (AWS)[12].

The cloud layer stores and manages privacy- and latency-tolerant data for tasks such as data visualisation and alert generation at the application level. Compared to edge layers, both data storage and extraction processes in the cloud layer exhibits consistent latency across different query scales and is cost-effective. While cloud storage is cost-effective for bulk, long-term processes, the framework reduces overall cost by avoiding unnecessary cloud invocations for latency-critical and privacy-sensitive queries by processing them at the edge layer.

E. The Application

A microservices-based web application has been developed to evaluate the proposed framework. This application retrieves IoT data from the edge and cloud layers through queries issued in natural-language. It infers the data and presents relevant results as text-based responses and dashboards. The application consists of three decoupled services: a visualisation and query interface, a Flask API to coordinate natural language processing, and a local-hosted Large Language Model (LLM) server by Ollama[13]. Such decoupling of services ensures dynamic context aggregation and large-scale reasoning, making the overall system horizontally scalable.

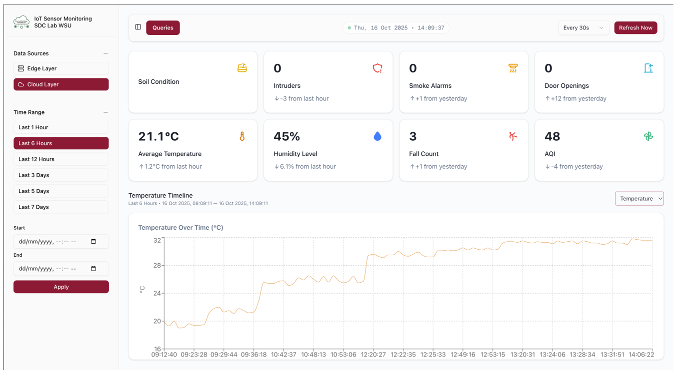


Fig. 3. An IoT data visualisation dashboard from the web application.

Fig. 3 depicts the web application's interface, which serves as the primary interaction point for caregivers and healthcare professionals. The dashboard visualises heterogeneous IoT data from the framework. The left panel allows users to choose the data source layer (edge or cloud layer) and set temporal filters—from the last hour to last 7 days or custom date ranges. The top panel displays real-time data measurements in summary cards. Some other cards specifically for the edge

---

[5] https://www.stereolabs.com/en-au/store/products/zed-2i

[6] https://www.raspberrypi.com/products/raspberry-pi-4-model-b

[7] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit

[8] https://docs.edgexfoundry.org/4.0

[9] https://flask.palletsprojects.com/en/stable

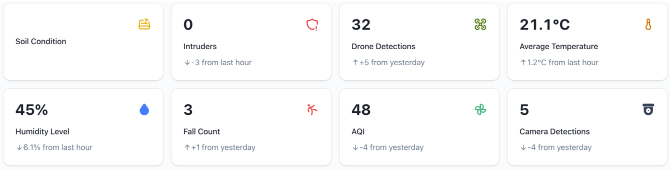
[10] https://www.mongodb.com

[11] https://prestodb.io

[12] https://aws.amazon.com

[13] https://ollama.com

layer is shown in Fig. 4. The bottom section features an interactive timeline chart showing selected data (temperature, in this case) throughout the chosen (6-hour) time-frame. From the edge Layer, users can access privacy-sensitive data of drones and cameras that are not stored in the cloud layer.



Soil Condition

0
Intruders
↓-3 from last hour

32
Drone Detections
↑+5 from yesterday

21.1°C
Average Temperature
↑1.2°C from last hour

45%
Humidity Level
↓6.1% from last hour

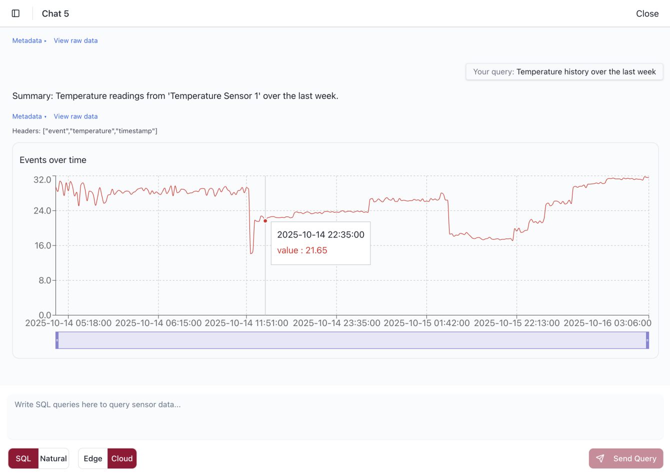
3
Fall Count
↑+1 from yesterday

48
AQI
↓-4 from yesterday

5
Camera Detections
↓-4 from yesterday

Fig. 4. A few IoT data display cards from the web application.

Natural Language to SQL (NL2SQL) translation engine is a key service within the application. It performs dynamic context aggregation by interpreting user queries based on a unified schema encompassing diverse sensor modalities, such as temperature, humidity, motion detection, door sensors, and assistive robots, in order to generate structured SQL queries. Fig. 5 presents an example natural language query that retrieves historical temperature over a week. The results are displayed as an interactive line chart. The visualisation type adapts dynamically depending on the nature of the returned results. The reasoning pipeline, built upon an LLM, is augmented with domain-specific ontologies to disambiguate temporal expressions (e.g., “last week”, “yesterday morning”) and resolve device aliases using fuzzy matching algorithms.



Your query: Temperature history over the last week

Summary: Temperature readings from 'Temperature Sensor 1' over the last week.

Headers: ["event","temperature","timestamp"]

Fig. 5. An example of a natural language query with relevant data visualisation.

Through NL2SQL translation engine, the application supports large-scale reasoning over semantically heterogeneous data sources without requiring end-users to possess SQL knowledge. The inference layer further enhances raw query outcomes using pattern detection heuristics, AI-generated summaries, and the transformation of tabular sensor data into context-related insights that are comprehensible to caregivers and healthcare practitioners.

IV. COMPARISON WITH STATE OF THE ART

Studies related to edge-cloud continuum frameworks predominantly focus on system deployment and orchestration, often leveraging container-based tools such as Kubernetes [8]. Some works extend this perspective to IoT system deployment over emerging 6G infrastructures [9]. However, there is limited research that integrates artificial intelligence and data visualisation capabilities within these frameworks. Among the closest related works is the edge-intelligence framework presented in [10], which supports monitoring of elderly individuals using non-wearable IoT sensors. This framework incorporates machine learning models for anomaly detection and automated alert generation. Nevertheless, it lacks purpose-based data exploration capabilities, restricting caregivers to querying raw IoT data rather than interacting with inferred results.

Existing edge intelligence frameworks primarily focus on model inference at the edge using raw data to support real-time decision-making in the AAL domain [11]. While such approaches enable AI-assisted classification and analysis, they do not address the accessibility challenge of empowering non-technical caregivers to directly query inferred IoT data. The proposed framework addresses this gap by integrating LLM-based natural-language querying within an edge–cloud continuum architecture. By leveraging the EdgeX platform to enforce a standardised IoT data schema for interoperability, and by tailoring the query mechanism to the contextual requirements of AAL environments, the framework enables both rapid response and accessible data interaction. This integration not only enhances usability but also contributes to the scalability and functional richness of edge–cloud continuum systems.

V. EVALUATION AND RESULTS

The implemented system’s performance is evaluated based on a series of latency benchmarks under varying query workloads. The testbed comprised heterogeneous IoT devices (see Table I) connected to multiple edge nodes, including Raspberry Pi 4 and 5 devices and an NVIDIA Jetson Nano. These nodes interfaced with an edge server deployed on a desktop machine (Intel i5 processor with 8 GB RAM), with a public cloud backend implemented on Amazon Web Services. The test-bed was set with two execution pathways within the framework. Both pathways involve the complete system pipeline, including NL2SQL translation using the Ollama-hosted LLM, query execution, and inference result summarisation. In the first pathway, queries are executed against the edge layer. In the second pathway, queries are sent to Amazon Athena[14] (cloud layer) to execute in a serverless environment over S3[15] data lakes. The analysis includes two types of comparisons. (i) A scalability analysis, measuring the system response times independently for edge layer and cloud layer execution pathways, ensuring the framework can achieve acceptable latency as the number of concurrent workload

---

[14] https://aws.amazon.com/athena
[15] https://aws.amazon.com/s3

increases. (ii) A computing continuum validation, investigating the QoS-aware routing strategy by studying the patterns of data distribution among heterogeneous IoT devices with different privacy, latency and data volume performance. This comparison confirms that the framework correctly routes the data to the appropriate layer as per application-specific needs.

A. *Scalability Analysis*

The framework's latency analysis under varying query scales was conducted, with queries ranging from 1 to 150 per trial across 5 trials in each layer. Fig. 6 presents these results, comparing the proposed system's cloud-only and edge-only tiers under concurrent query loads (1-150 queries), plotted on a logarithmic scale to clearly visualise order-of-magnitude differences between tiers.

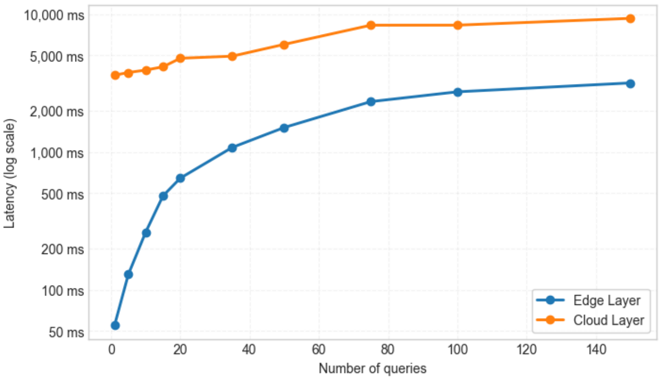


Fig. 6. Scalability analysis (average latency vs increasing number of queries) of edge layer and cloud layer under increasing concurrent workloads.

The edge layer exhibits a latency of 50 ms for a single query, increasing to approximately 600 ms for 20 parallel queries, and reaching 3,200 ms for 150 parallel queries. The cloud layer shows a higher baseline latency, ranging from 3,500 ms to 9,200 ms, due to network overheads associated with the remote execution. However, as the number of queries increases, cloud-layer latency grows more gradually and remains stable, indicating the elastic resource provisioning. This behaviour highlights a key trade-off: while the cloud incurs higher per-query latency, it is more suitable for large-scale parallel workloads. At higher concurrency levels (80+ queries), both layers exhibit improved scaling behaviour, attributable to the intelligent query routing strategy implemented in the framework. The scalability observed during peak loads demonstrates the QoS-conscious design of the routing mechanism, where the edge layer handles latency-sensitive queries and the cloud processes latency-tolerant workloads. This approach avoids unnecessary cloud invocations for latency-sensitive or privacy-critical tasks, while also preventing the edge from being burdened with computationally intensive processing, thereby reducing resource strain. Through selective utilisation of processing tiers, the framework achieves scalability while improving cost efficiency.

B. *Computing Continuum Validation*

The proposed framework's functionality is validated using six different IoT devices. They vary in terms of temporal, spatial, and privacy-related data characteristics. The experimentation was conducted by generating data objects for each IoT device and observing where framework stores the data. Fig. 7 depicts data distribution results across the continuum.

A distinct differentiation is observed in the results: privacy-sensitive and latency-critical data produced by the Misty Robot, Tello Drone, and ZED 2i camera are stored at the edge. Non-privacy-sensitive and latency-tolerant IoT data produced by the THP (Temperature–Humidity–Pressure) sensor is stored in the cloud layer. This device produces the data in bulk and at high-rates. The framework utilises the cloud layer's virtually unlimited storage and computational resources to scale and store this data in a cost-effective manner. Door sensors exhibit a hybrid storage behaviour, with most records stored at the edge and others in the cloud. Both the door sensor and THP sensors' data can be stored at either location or both simultaneously. For such cases, the proposed framework makes context-based data placement decisions based on dynamic aspects, such as network latency, freshness requirements, and storage quotas.

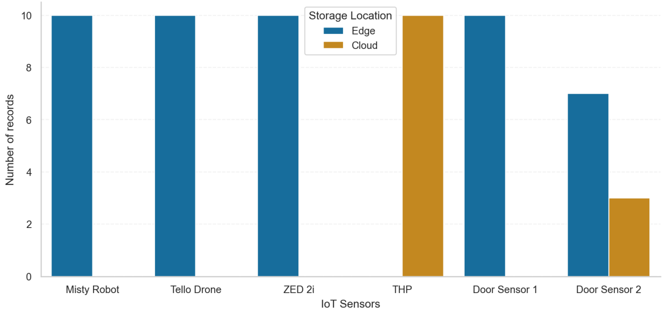


Fig. 7. Count of Records by Sensor and Storage Location

For each incoming data object, the framework calculates an aggregated score based on its latency and privacy characteristics to determine its suitability for each layer . The layer with the highest score is given preference. If the scores are equal, the data record is stored in both edge and cloud layers. Fig. 8 presents the storage decision scores for all IoT devices. Positive scores indicate edge preference, whereas negative scores indicate a cloud preference.

Based on Fig. 8 the storage decisions align with predictable and understandable circumstances: the Misty Robot has the greatest edge-affinity value (~0.25) with the least variance, indicating that this system is a real-time assistive technology that demands immediate responsiveness in order to detect faces and human interaction events. On the same note, Door Sensor 1 has a high edge-preference (~0.22), which is suitable in security critical event detection where latency has a direct influence on safety consequences. The THP sensor, on the other hand, is registering a negative score (~-0.08), which is a correct routing of bulk environmental telemetry to the

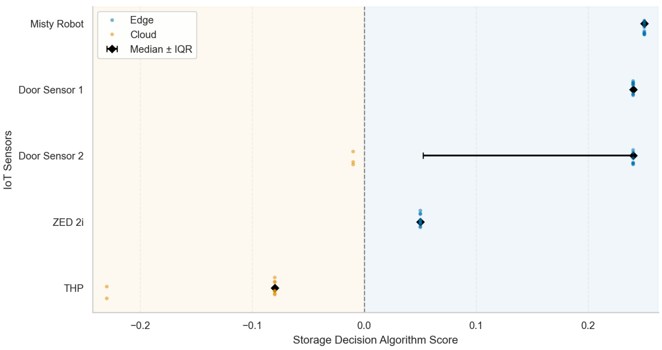


Fig. 8. Storage-Location Decision Algorithm Scores

cloud tier, where longer-term analytical queries are more prevalent than real-time monitoring. The ZED 2i camera and Door Sensor 2 are in the middle, but Door Sensor 2's higher interquartile range suggests routing decisions that adapt to operational changes rather than fixed configuration. The experimental results confirm the effectiveness of the framework in coordinating seamless interoperability across the edge-cloud continuum and still maintain deterministic, QoS-aware routing behaviour adaptable to the data features as well as application-specific, latency requirements.

VI. Impact and Future Work

The proposed computing continuum framework demonstrates a notable impact in the democratisation of IoT data access for non-technical caregivers, while maintaining privacy by placing sensitive data at the edge layer. It also exhibits strong scalability by enabling the intelligent distribution of data and queries across edge and cloud resources, accommodating increasing data volumes and concurrent query workloads with minimal performance degradation. By leveraging standard data structure, the framework enables seamless integration of heterogenous IoT devices, providing a replicable and extensible model for AAL-deployments.

The current implementation adopts a text-based query-response paradigm, and the cloud layer is entirely implemented using AWS cloud infrastructure. Future work will address these limitations by supporting voice and visual queries, as well as enabling federated queries across heterogeneous cloud providers. This extension will improve compliance with data sovereignty requirements and minimise vendor lock-in.

VII. Conclusion

This paper introduced a scalable computing-continuum framework for enabling IoT-based Ambient Assisted Living (AAL) environments. The framework unifies heterogeneous data sources within an integrated edge-cloud continuum consisting of edge devices, an edge server, and a cloud layer. It is integrated with an LLM-based query interface that converts natural language queries into sensor-specific data queries at scale, allowing non-technical caregivers and healthcare professionals to access complex IoT data without requiring SQL knowledge. Its QoS-aware routing mechanism effectively directs privacy-sensitive and bandwidth-intensive data to the edge layer, while transferring bulk telemetry and computation-intensive workload to the cloud, thereby achieving scalability while maintaining latency constraints, privacy requirements, and processing cost efficiency.

Experimental analysis demonstrated that the storage decisions are deterministic and interpretable, with more than 95% consistent in directing privacy-critical IoT data to the edge layer. The system successfully responded to 100% of 150 simultaneous queries in each layer. The edge layer provides near real-time responsiveness for latency- and privacy-sensitive queries under low to moderate concurrency levels, and maintains acceptable performance during peak workloads. The cloud layer exhibits predictable scalability, albeit with a higher baseline latency. These findings confirm that the proposed framework can support near real-time AAL responses under appropriate workload conditions, while scaling predictably across heterogeneous computing resources.

References

[1] K. S. Jagarlamudi, A. Zaslavsky, S. W. Loke, A. Hassani, and A. Medvedev, "Requirements, limitations and recommendations for enabling end-to-end quality of context-awareness in IoT middleware," *Sensors*, vol. 22, no. 4, p. 1632, 2022, doi: 10.3390/s22041632.

[2] P. K. Donta, I. Murturi, V. Casamayor Pujol, B. Sedlak, and S. Dustdar, "Exploring the potential of distributed computing continuum systems," *Computers*, vol. 12, no. 10, Art. no. 198, 2023, doi: 10.3390/computers12100198.

[3] K. Islam, G. D. C. Rodrigues, B. Javadi, and R. N. Calheiros, "MicroIntent: Intent-Based Placement Strategy for Microservice Application in the Compute Continuum Using LLMs," in *Proc. IEEE Int. Conf. on Smart Internet of Things (SmartIoT)*, pp. 124–131, 2025.

[4] World Health Organization. *State of the world's nursing report 2025*. EB156/15 report, World Health Organization, May 2025. Available: https://iris.who.int/server/api/core/bitstreams/a4173924-a18f-49b6-8bd1-9c2a4a098980/content. Accessed: Dec. 11, 2025.

[5] K. Y. Fang, J. A. Ginige, J. Basilakis, H. Bjering, and B. Javadi, "Smart homecare research translation into broader practice: Enablers, barriers and directions," *IEEE Access*, vol. 10, pp. 134726–134743, 2022, doi: 10.1109/ACCESS.2022.3224099.

[6] A. Al-Dulaimy, M. Jansen, B. Johansson, A. Trivedi, A. Iosup, M. Ashjaei, A. Galletta, D. Kimovski, R. Prodan, K. Tserpes, G. Kousiouris, C. Giannakos, I. Brandic, N. Ali, A. B. Bondi, and A. V. Papadopoulos, "The computing continuum: From IoT to the cloud," *Internet of Things*, vol. 27, Art. no. 101272, 2024, doi: 10.1016/j.iot.2024.101272.

[7] EdgeX Foundry. *EdgeX Foundry — Core Data API*. 2026. [Online]. Available: https://docs.edgexfoundry.org/4.0/api/core/Ch-APICoredata. Accessed: Jan. 25, 2026.

[8] G. Koukis, S. Skaperas, I. Kapetanidou, V. Tsaoussidis, and L. Mamatas, "An open-source experimentation framework for the edge cloud continuum," 2024.

[9] H. F. Shahid and E. Harjula, "Resource-aware orchestration of IoT applications in edge-cloud continuum with 6G," in *Proc. IEEE Int. Conf. on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops)*, pp. 245–246, Mar. 2024, doi: 10.1109/PerComWorkshops59983.2024.10503485.

[10] A. Ali, T. Montanaro, I. Sergi, S. Carrisi, D. Galli, C. Distante, and L. Patrono, "An innovative IoT and edge intelligence framework for monitoring elderly people using anomaly detection on data from non-wearable sensors," *Sensors*, vol. 25, no. 6, Art. no. 1735, 2025, doi: 10.3390/s25061735.

[11] V. Hayyolalam, M. Aloqaily, Ö. Özkasap, and M. Guizani, "Edge intelligence for empowering IoT-based healthcare systems," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 6–14, Jun. 2021, doi: 10.1109/MWC.001.2000345.