National College of Ireland

Adaptive Serverless FaaS Dataflow Framework for Real-Time IoT Analytics across the Cloud-Edge Continuum

MSc Research Project
MSc Cloud Computing

Pradnya Deshmukh
Student ID: X23149604

School of Computing
National College of Ireland

Supervisor: Prof Shivani Jaiswal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Pradnya Deshmukh
Student ID:	X23149604
Programme:	MSc Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof Shivani Jaiswal
Submission Due Date:	12/12/2024
Project Title:	Adaptive Serverless FaaS Dataflow Framework for Real-Time IoT Analytics across the Cloud-Edge Continuum
Word Count:	XXX
Page Count:	26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	27th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Adaptive Serverless FaaS Dataflow Framework for Real-Time IoT Analytics across the Cloud-Edge Continuum

Pradnya Deshmukh
X23149604

Abstract

Robust cloud-to-edge communication infrastructure is now available due to proliferation of 5G networks and IoT devices. Existing IoT systems, however, do not have efficient mechanisms for workflow orchestration or adaptation to dynamic environments. This research introduces an adaptive serverless framework for real time IoT analytics across the cloud edge continuum and uses intelligent workflow placement to improve the responsiveness of these workloads. The framework realizes a new machine learning based anomaly detection, time series prediction, and reinforcement learning approach to processing location determination. An autoencoder based anomaly detection model that identifies patterns which require special processing, LSTM networks for prediction of processing requirements and a Deep Q network for dynamic placement decision in the edge, hybrid and cloud environments are employed by the system. This framework has been implemented on AWS infrastructure and show results up to 13x superior placement efficiency than traditional rule based approaches with single request latencies of 63.92ms versus 832ms, and consistent performance up to large loads of 80k concurrent users. Its ML-driven decision making improves placement accuracy and adaptation to changing conditions significantly, according to evaluation. Specifically, this research advances the state of the art in the realm of IoT analytics by enabling a flexible orchestration mechanism that can adapt to a number of real world scenarios encountered in smart city, healthcare system, and other IoT applications.

1 Introduction

1.1 Research Problem and Background

With the exponential growth of internet of things (IoT) devices and 5G networks, it is possible to have seamless cloud to edge communication with massive amounts of data being generated, where real time processing and analytics are needed (Geldenhuys et al.; 2021). Nevertheless, current IoT systems suffer from inability to accommodate dynamic environments and distributed data processing resulting in suboptimal system performance and resource utilization (Bhatia et al.; 2022). Machine learning (ML) models have become critical for real time data processing and predictive analytics, but do so at the expense of complexity with regard to model deployment and data management across the cloud edge continuum (Bhadula et al.; 2024).

Recent serverless computing which uses Function-as-a-service (FaaS) with cloud backend system has been proposed to overcome the aforementioned drawbacks, providing a scalable and event driven platform for applications execution without (Patil et al.; 2021). Addressing latency sensitive applications, intermittent connectivity, and network bandwidth limitations (Kong et al.; 2022), edge computing has also become popular. Serverless computing, edge processing and ML models collectively offer new opportunities for ultra adaptive, efficient and intelligent IoT systems. While also presenting challenges in data control, model deployment and system orchestration across the cloud edge continuum (Benomar et al.; 2020) (Murshed et al.; 2021).

However, advanced orchestration mechanisms of existing frameworks for dynamic mapping of workflows over heterogeneous cloud-edge devices hinder their workflow placement optimization, resource utilization, and Quality of Service (QoS) efficiencies in IoT scenarios (Laso et al.; 2022). Hence, a new approach is required to combine the adaptive serverless computation with effective carrying out of ML model deployment and dynamic workflow orchestration balancing the low latency bottlenecks at the edge as well as complex cloud computations.

1.2 Motivation

This research is motivated by the large gap between the promise of serverless computing and the realities of cloud edge continuum in IoT scenarios. Current IoT systems have suboptimal performance and resource utilization due to inability to adapt to changing environments and QoS requirements. To make real time IoT analytics possible with changing workloads and network conditions, we need to develop an adaptive serverless FaaS dataflow framework.

By integrating ML models into this framework, we have an opportunity to improve decision making processes, optimise resource allocation and to increase overall system performance. Serverless computing and edge processing blended together makes it possible to bring responsiveness, efficiency, and scalability to IoT applications across different domains such as smart cities, healthcare, and Internet of Things.

1.3 Research Question

In what manner does the proposed adaptive serverless dataflow framework integrates ML approaches for real-time IoT analytics to make informed decisions on dynamic workflow placement, resource utilization, while meeting QoS requirements and maintaining stateless application integrity ?

1.4 Research Objective

The purpose of this research is to formulate an adaptive serverless FaaS analysis dataflow framework with cloud edge continuum real-time IoT analytics. In edge environments, the framework integrates machine learning for dynamic workflow decomposition, placement and relocation under latency, resource constraints as well as heterogeneity. the real-world IoT scenarios will be used to evaluate the framework and metrics would include latency reduction, QoS improvement and scalability.

1.5 Research Contributions

- Design and implementation of an adaptive serverless FaaS dataflow framework for IoT analytics across the cloud-edge continuum. This framework integrates edge computing, serverless functions, and cloud processing, coordinated by an intelligent orchestration engine. It addresses the challenges of distributed data processing and adaptation to dynamic environments in IoT systems.

- Development of ML-based orchestration mechanisms for dynamic workflow decomposition, placement, and relocation across heterogeneous computing resources. This includes the integration of machine learning models for predicting resource needs, identifying optimal function placements, and improving overall system performance. The orchestration engine uses both historical data patterns and real-time metrics to make informed decisions.

- Efficient pipelines in data processing and ML model deployment strategies that can be deployed on resource constrained edge device. The solution to that is to pre process everything at the edge to reduce the load on later stages and increase the speed at which time sensitive operations can run.

- Specification and implementation of elastic provisioning and handling mechanisms for serverless functions to respond to dynamically changing workloads. This works to allow the system to respond instantly to changes in workload or over time, addressing modern problems in IoT ecosystems.

- Real world IoT scenarios are used to provide comprehensive evaluation of the proposed framework based on latency reduction, QoS improvements and scalability. It includes a comparison of our work to traditional cloud based IoT architectures as well as existing serverless platforms for IoT analytics.

1.6 Thesis Organization

The thesis presents the adaptive serverless FaaS dataflow real time IoT analytics across the cloud-edge continuum as proposed. An introductory section discusses the research problem, background, motivation, question and objectives, and starts with the introduction. In the second section an extensive literature is reviewed concerning the most recent developments in the cloud edge computing, serverless workflows, IoT data analytics, and orchestration frameworks. The methodology section outlines the research approach by describing a framework design, choosing AWS services, data collection methods and evaluation strategies. The proposed framework's architecture, functional components, and operational mechanisms are presented in section 4. The implementation section presents the practical realization of the framework, including setting up the AWS services, developing serverless functions, integrating ML models, and deployment from the cloud to the edge continuum. The results and discussions section reports the results from the system evaluation, including performance analysis, comparative studies and the ability of the system in different IoT scenarios. The thesis ends with a discussion of the main findings, research implications, and suggestions for further work.

2 Related Work

In this literature review, we study the recent developments on cloud edge computing, serverless workflow, IoT data analytics and orchestration frameworks. The research covers a broad spectrum of topics related to distributed computing ranging from function deployment to data processing and application management throughout the cloud-to-edge continuum.

2.1 Cloud-Edge Continuum

Sicari et al. (2022) propose OpenWolf, an open-source serverless workflow engine for applications by federating the cloud-fog-edge tiers into a single continuum environment. They define a custom workflow manifest DSL to describe function interactions and implement an agent that can deploy architecture-independent functions and coordinate them according to the manifest. This tackles issues of existing Function as a Service (FaaS) approaches, namely the need of a cloud broker to orchestrate functions and the inability to run arbitrary jobs with multiple inputs. Various deployment scenarios are tested to validate the proposed approach using scalability, latency, and performance metrics. OpenWolf compares to existing literature since it addresses the requirement to reengineer applications to work well in the new heterogeneous cloud-edge environment. On a more reactive note, this work extends earlier serverless computing and cloud-edge continuum by introducing a proactive process placement across continuum nodes. The results indicate the success of OpenWolf to deploy and organize functions over the continuum while decreasing development speed, cost of service deployment, and ease of deployment.

This systematic mapping study investigates platforms for FaaS in the Cloud-to-Thing continuum from the deployment, placement, orchestration and execution angle of cloud to edge devices (Oliveira et al.; 2023). The publication trends, platform architecture and runtime capabilities for the FaaS programming model are reviewed using 33 primary studies found from four major online publication databases, observing a growing use of FaaS in the Cloud-to-Thing continuum, with most research focused on deployment and placement strategies for Cloud and Fog layers. However, as few of the platforms cover the Thing layer and rather many of them support function deployment to Cloud and Fog infrastructures, it was found lacking on research coverage of the Things layer. The traditional isolation to isolate other resource consumers is normally achieved by means of virtualization techniques like Docker containers and WebAssembly, but they do not provide sharing of resources at Fog and Thing layers in a satisfactory way. The authors then conclude that, while FaaS has advanced for the Cloud-to-Thing continuum, there remain numerous challenges, including insufficient support of the Things layer and insufficient flexibility in deploying and customizing supporting services.

In (Koukis et al.; 2024), the CODECO experimentation framework is introduced, an open-source solution for rapid deployments of Kubernetes-based edge cloud. The framework provides holistic experiment configuration, new abstractions and automation for various experiments on the edge cloud continuum. The resulting system adopts microservice-based architecture, provides novel abstractions for deployment of Kubernetes cluster, declarative cross-layer experimentation configuration, and automation features. It is lightweight, supports additional edge capabilities, and has automation features, when compared to existing solutions such as ClusterSlice. The authors demonstrate the framework's capabilities through three proof-of-concept experiments by analyzing the

performance of various network fabrics on different edge-oriented Kubernetes distributions, the automation of deployment of EdgeNet (an edge cloud orchestration system), and anomaly detection workflows for edge environment. The results demonstrate the automation and full cluster, app and experiment deployment provided by the framework.

In their work, (Farahani et al.; 2024) performed a mini survey of serverless workflow management systems (WMS) opportunities and challenges on the computing continuum. One important aspect of the study aimed to understand how the serverless paradigm can be used to optimize the service level objectives (SLOs), energy efficiency and economic costs. Based on that, the authors categorize the state-of-the-art WMS into cloud-based, edge-cloud continuum-based, and simulation-based WMS. Advantages like scalability and stateless cloud-native infrastructure, auto-scaling, and pay-per-use cost models were highlighted. But they also pointed out issues in designing serverless WMSs like inconveniences related to stream-processing WMS workflow, latency issues, data distribution and storage management challenges, and resource inefficiencies. It was concluded that academia and industry will need to join forces to overcome these challenges and realize the full potential of serverless WMSs on the computing continuum.

This literature studies different approaches to managing serverless workflow in the cloud-edge computing environment. The function orchestration enabled by OpenWolf is a notable solution for deploying and orchestrating functions across the cloud-fog-edge continuum, and the presented custom DSL for functions interactions. Nevertheless, these systems have some significant limitations.

The main drawbacks are lack of a comprehensive support to the Things side of the Cloud–Thing continuum: most of the platforms are oriented only to Cloud and Fog layers. Despite the containerization technologies, resource sharing at Thing and Fog layer is inadequate. However, stream processing workflows, latency optimization, and data distribution are still challenging problems for existing solutions. Frameworks such as CODECO strive to automate deployment and experimentation, but the space remains unfulfilled for dealing with complex workflows across diverse environments. Optimizing service level objectives, energy efficiency and economic costs in serverless architectures are still significant challenges to be tackled jointly by academia and industry.

2.2 Faas Workflow Models

In this paper, (Li et al.; 2022) introduce FaaSFlow, an efficient serverless workflow system that attempts to address the shortcomings of existing master-worker based workflow execution architectures in serverless computing environments. Quick data transfer and limited reliance on database are the key features of this system which aims to mitigate the scheduling and data movement overhead that are usual to serverless workflows by means of offloading data scheduling to each worker node. FaaStore library is used for dynamic memory allocation and efficient data storage and FaaSFlow has a workflow graph scheduler on the master node and the per-worker workflow engine on each worker node. Experimental results demonstrate that average FaaSFlow mitigates workflow scheduling overhead by 74.6%, and data transmission overhead by up to 95%. Furthermore, FaaSFlow-FaaStore reduced the throughput degradation of 23.0 percent when network bandwidth fluctuation and multiplied the network bandwidth utilization by 1.5X - 4X.

In (Paraskevoulakou and Kyriazis; 2023), the new approach to deploy ML pipelines via serverless computing is introduced, called Machine Learning FaaS (ML-FaaS) intended to use serverless architectures to perform efficient and scalable execution of data analytics

and ML/DL workflows. In this work, the authors propose to enable extension of serverless functions through added containers to address code size constraints. The ML-FaaS pipeline is designed in two phases: The data preparation and model training are held offline, and inference on new data is held online. Challenges like imbalanced datasets, cold start issues are addressed on the serverless platforms, and an AI-based "Function Regulator" is introduced to automate the finding of the optimal number of functions required in creating the serverless application. They show that function chaining can be used to solve complex ML tasks with stateless serverless functions. On the fraud detection task, the MLP model reached 84% recall and warm serverless executions had acceptable latency. It was also found that the ML-FaaS approach needs approximately only 50% of the development time of traditional cloud deployments.

Process-as-a-Service (PraaS) is a novel cloud abstraction building on top of FaaS to overcome its flaws while preserving its strength of elasticity and scalability. To overcome infrastructure and lack of sufficient control-flow orchestration capabilities of the existing control-flow based systems, DataFlower, a novel scheme for orchestrating serverless workflows based on data-flow paradigm was presented in (Li et al.; 2023). It decouples function computation logic from data transmission logic inside containers creating an asynchronous data transmission and computation communication overlap. DataFlower is implemented by redesigning serverless programming and execution model, rearchitecting workflow orchestration to support decentralization, data availability driven function triggering, and designing an efficient inter-function data transfer mechanism by using host container collaborative communication. The pressure-aware function scale mechanism is key, as are data sinks on each host node to cache input data before function triggering. Experimental results demonstrate that DataFlower reduces the 99th percentile latency by up to 35.4%, the peak throughput by up to 3.8 times, and reduces the container memory usage of up to 19.1 percent to 69.3 percent of FaaSFlow. The system is extremely effective at dealing with bursty workloads as well as different workflow structures and input sizes.

A comprehensive benchmarking framework named XFBench is presented in (Kulkarni et al.; 2024) to evaluate Function-as-a-Service (FaaS) workflow platforms on multiple cloud providers. It addresses the lack of standardized benchmark for FaaS workflows, that are becoming increasingly important in serverless computing. XFBench provides extensible workflow and function benchmarks, cloud service provider agnostic framework and configurable zero touch benchmarking capabilities. In this case, it works with the open-source XFaaS multi-cloud platform and can be deployed on FaaS workflow platforms, including Azure Durable Functions and AWS Step Functions. This includes thirty diverse, application focused functions for application workflows and two micro benchmark workflows spanning the breadth of computational patterns and resource needs. What sets XFBench apart is workflows, multi cloud provider support, and increased functionality and configurability in the workloads. Finally, the authors validated XFBench by deploying a spate of diverse workflows and workloads on platforms in Amazon Web Services (AWS) and Microsoft Azure in different global regions and found that performance and scaling behavior vastly differ across platforms.

The discussed literature provides a number of promising advancements of serverless computing and Function as a Service (FaaS) frameworks. Moreover, Dyninka, FaaSFlow, ML-FaaS, DataFlower, PraaS and XFBench attempt to cope with limitations of existing serverless architectures. These are solutions addressing the efficiency, overhead reduction, data transfer improvement, complexity workflow support. However, though they improve dramatically in latency, throughput and scalability, these solutions are also lacking. Yet

a lot of these frameworks tend to only work for certain use cases or cloud providers and don't have widespread usefulness. The solutions often require major modification to the existing architectures of serverless. Further, ML-FaaS continues to lack some of the cold start issues or provide addressing, and respectively, the stateless nature of serverless functions. As it is, benchmarks of the field are not standardized, leading to difficulty in assessing both relative performance and suitability of various solutions for different applications.

2.3 IoT Data Analytics

In (Escobar et al.; 2023), an innovative IoT framework for building dataflow applications in the Cloud to Edge Continuum is proposed. To tackle the challenges of the new IoT context, and take advantage of intelligence at all levels, the framework employs resources that are available in massive IoT scenarios. For lightweight services along the Cloud to Edge infrastructure, the design employs decentralized Pub/Sub communication with Zenoh technology and its serverless nanoservice architecture on top of WebAssembly. Technologies like Zenoh and WebAssembly are incorporated into this framework to have a decentralized, lightweight and efficient IoT architecture. On the basis of technical and functional requirements such as lightweight applications, mobility, scalability, data oriented processing, fluid computing, and trust and execution environments, the design decisions are justified. It brings decentralized Pub/Sub communication, dataflow programming, lightweight virtualization, and orchestration together. This architecture supports the development of advanced IoT applications for efficient data processing at the right level of the network, taking advantage of disaggregated resources securely and correctly.

Donati et al. (2024) propose a simulation and analysis framework of event driven AI workflows in serverless computing environments to process and decide on real time data processes. It's serverless functions, machine learning models, workflow orchestration for continuous data streams. It uses serverless functions, AI components, and workflows coordinated through cloud-based platforms. Impact of study is studied on key performance metrics such as latency, throughput and resource utilization. The observed tradeoff of latency versus throughput is measured, with average processing latency typically increasing with event rates and average throughput increasing. The use of different event rates has limited effect on resource utilization. In contrast to past studies, the study offers holistic solutions that combine disparate event driven architecture and serverless components. They conclude with a discussion of the strengths and weaknesses of their approach, and a consideration of how it can be used to optimally resource allocate strategies and to identify performance bottlenecks.

López Escobar et al. (2024) proposed an innovative dataflow framework for building Cloud-to-Edge Continuum applications to address the emerging computing and communication trends such as Edge Computing, 5G, and IIoT. To enable lightweight services across the Cloud to Edge infrastructure, in data centric environments and manage intelligently, the framework relies on decentralized publish subscribe communication using Zenoh and serverless nanoservice architecture with WebAssembly. Using Zenoh, the proposed decentralized approach is compared to the classical centralized MQTT choices in terms of performance under Mist, Fog-Mist, and Cloud-Fog-Mist network topologies. Results are consistent across different scenarios and payload sizes, proving that the decentralized Zenoh approach using MQTT consistently outperforms it in latency. It also discusses possible use cases such as mobile eHealth monitoring with smart ambulances,

and emergency emergency rescue with UAV swarms. The authors argue that the decentralization in Cloud-to-Edge architectures provides some additional advantages in terms of latency and scalability over the centralized ones.

Oliveira et al. (2024) present IoTDeploy, a solution for the deployment of IoT smart applications across the continuum of computing; The work presented attempts to address service orchestration and DevOps strategies for heterogeneous distributed environments. IoTDeploy has a CI/CD comprising a tool plugin that supports dynamic service migration. To build a pipeline model, the authors used technology such as GitHub, Docker, Kubernetes and Jenkins. Using a case study involving smart irrigation in agriculture, they assessed IoTDeploy. However, the experiment has shown deployment into the IoT distributed environment is reliable and resilient and migrates without interruption or data loss. IoTDeploy tackles the automatic service deployment gap in the context of IoT applications hybrid deployed over a geographical and computing continuum while inheriting the current CI/CD approaches but implementing dynamic service migration among stages.

Frameworks and architectures for elastic data analytics in IoT environments are the main focus of discussion in the literature, but these come with some limitations. Often, these frameworks tend to consider theoretical architectures lacking the actual implementation data or performance benchmarks, are not secured adequately nor do they address the challenges related to heterogeneous IoT devices and protocols. QoS parameters and resource optimization is discussed in some work, but does not present any concrete solutions for network instability and bandwidth restrictions. While superficial, the integration of AI/ML is not logged as much as there are documented methods on how to carry them out. Error handling and data integrity and consistency are handled poorly in most frameworks.

2.4 Dataflow Orchestration Techniques

Grigoropoulos and Lalis (2022) introduce Fractus, an orchestration framework for automated deployment of distributed applications along the drone-edge-cloud continuum. With Druxus and an integrated drone, edge, cloud platform, Fractus intends to offer a structured approach for developing and deploying next generation drone applications that utilize drone, edge and cloud resources as needed in a safe and privacyle compliant manner. The two placement and communication requirements are specified using structured descriptions and a mission aware deployment strategy that takes the area of operation of the drone into account and provides the components with connectivity without any transparency. Policy based access enforces safety and privacy constraints for mobility and sensor resources. Fractus is an effort to take the orchestration and management of drones a step further, enabling end to end deployment of applications which utilize edge and cloud resources in a transparent fashion. The results from a real drone and simulation setup demonstrated that Fractus provided the required functionality with a reduced development effort and a moderately high overhead.

Bocci et al. (2023) propose a new declarative methodology to securely place Function as a service (FaaS) orchestrations on Cloud Edge infrastructures, and make sure they satisfy functional and non functional requirements. It employs information flow analysis and padding techniques to thwart side channel data leaks. The methodology is implemented in a Prolog based prototype called SecFaaS2Fog. The authors identify three potential data leaks: service data leaks, weak node leaks, and control-flow leaks. To achieve these,

the security labels and the padding of conditional branches in the orchestration are used. Unlike prior work, SecFaaS2Fog takes information flow security (or infrastructure security) countermeasures into account when inserting latency-aware placement of the FaaS orchestrations. Experimental evaluations using FogSim simulator demonstrate that SecFaaS2Fog is able to successfully find eligible placements for execution times from 6 ms to 31 ms, driven by Parasol pre-periodic placements. However, SecFaaS2Fog is a feasible approach to safely deploy FaaS orchestrations in Cloud-Edge spaces.

XFaaS is a cross platform orchestration framework presented in (Khochare et al.; 2023) that leverages FaaS to automate continuous, cross cloud workflow for hybrid clouds. The goal is to break the vendor lock-in, portability and performance optimisation of FaaS work flows on top of multiple cloud providers. It also provides the capability of 'zero touch' function and workflow deployment across Amazon Web Services (AWS) and Microsoft Azure by automatically generating code wrappers, cloud queues, and managing FaaS engines natively. Through detailed benchmarks on FaaS workflow executions on Azure and AWS, the authors build performance models. To enable latency and cost reductions in hybrid cloud, they propose both intelligent workflow partitioning and function fusion techniques. Unlike single function portability solutions, XFaaS offers better cost modeling than prior fusions. Empirical results show that, in the placement and fusion space, the fusion technique employed by XFaaS can reduce latency by up to 75%, and cost by 57%; and placement techniques can further reduce the latency by up to 24% vs baselines. In addition, the fusion approach also finds less expensive and less latency configurations.

Pheromone, a scalable serverless platform based on function orchestration using data, is introduced by (Yu et al.; 2023). The platform offers fine grained exchange of data between functions decoupled from function flows. Through a two tier distributed scheduling hierarchy it exploits data locality and has high scalability. Using an on-node shared-memory object store for zero copy data exchange between local functions and direct inter node data transfer lowers the data sharing overhead, optimizes sharing, which is then scheduled in parallel by the operator. Pheromone overfits well established commercial and open source serverless platforms in terms of Invocation Latency and Data Exchange efficiency. The resulting evaluation indicates that Pheromone achieves between 10x and 450x better function invocation latency than Cloudburst and AWS Step Functions, respectively. Second, it scales well to large workflows with only millisecond scale orchestration overhead. With its rich expressiveness, high usability, wide applicability, we argue Pheromone is appropriate in both latency sensitive and data intensive applications.

There exist various orchestration frameworks for cloud based edge computing featured in the literature; but at the same time these frameworks have their limitations. But, Fractus improves edge invocation delay by 33%, while it lacks comprehensive security measures and advanced resource allocation optimization techniques. Though performance tradeoffs exist, SecFAs2Fog provides security aware FaaS placement. XFaaS solves the vendor lock-in and cross cloud deployment problem but only works on AWS and Azure platforms. Data centric performance improvements are provided by Pheromone but it does not provide broader orchestration features. Limitations common to current solutions include a lack of integration of security, performance and resource optimization in a single solution, and that real time adaptation to changing network conditions and workload patterns is hampered by maintaining guarantees on security and performance.

2.5 Critical Analysis

The review of literature brings out the significant advancements in the cloud edge computing, serverless workflows and IoT data analytics, and orchestration frameworks. There are, however, several research gaps. Most existing solutions provide limited functionality at edge and IoT device level and do not adequately support the entire cloud to thing continuum. Due to complex workflows across a heterogeneous environment that often require optimization of latency, utilization of resources, and minimizing cost effectiveness, current frameworks often struggle to successfully address such workflows. Most solutions also fail to address for security measures and real time adaptation to changing network conditions.

To overcome these limitations, the proposed approach brings an adaptive serverless FaaS dataflow framework to support IoT analytics across the cloud–edge continuum. At an edge computing, serverless functions and cloud processing level, this framework integrates an intelligent orchestration engine and edge computing. It addresses the problems of distributed data processing and adaptability to dynamic environments in the IoT system. The solution also employs ML based orchestration mechanisms for dynamic workflow decomposition, placement and relocation across heterogeneous computing resources. The framework seeks to reduce latency, improve QoS, and scale by deploying efficient data processing pipelines and ML models on resource constrained edge devices. We propose a comprehensive approach that addresses the gaps in the current solutions and achieves more robust, adaptive and efficient IoT analytics in the cloud-edge continuum.

3 Methodology

The research methodology is designed to yield an adaptive serverless framework for real time IoT analytics on the cloud edge continuum through a systematic approach. This approach is structured into five distinct layers: each part of overall research framework dedicated for data collection, Lambda functions, ML model deployment, processing location determination, and evaluation as presented in Figure 1.

3.1 Research Approach

To develop and evaluate the ML-driven workflow orchestration framework for IoT analytics, a systematic research methodology is followed based on the workflow patterns and stages. The methodology is structured into four distinct phases: data collection and analysis, model development, framework implementation, and evaluation.

The first phase after data collection and analysis methods starts with getting IoT sensor data from air quality monitoring devices. In this phase, the aim is to understand data patterns to identify critical features that are related to workflow placement decision, and system requirements analysis. The available historical data patterns are used to determine the processing demand characteristics and baseline metrics for performance evaluation. The measured data is PM2.5, PM10, temperature, humidity, CO2, TVOC, NO2.

The implementation of three specialized machine learning models forms the model development phase. We develop an autoencoder for detection of anomalies in the IoT data using PyTorch to determine the unusual pattern and, if necessary, upload their job to a specific processing location. We then explore LSTM networks for time-series

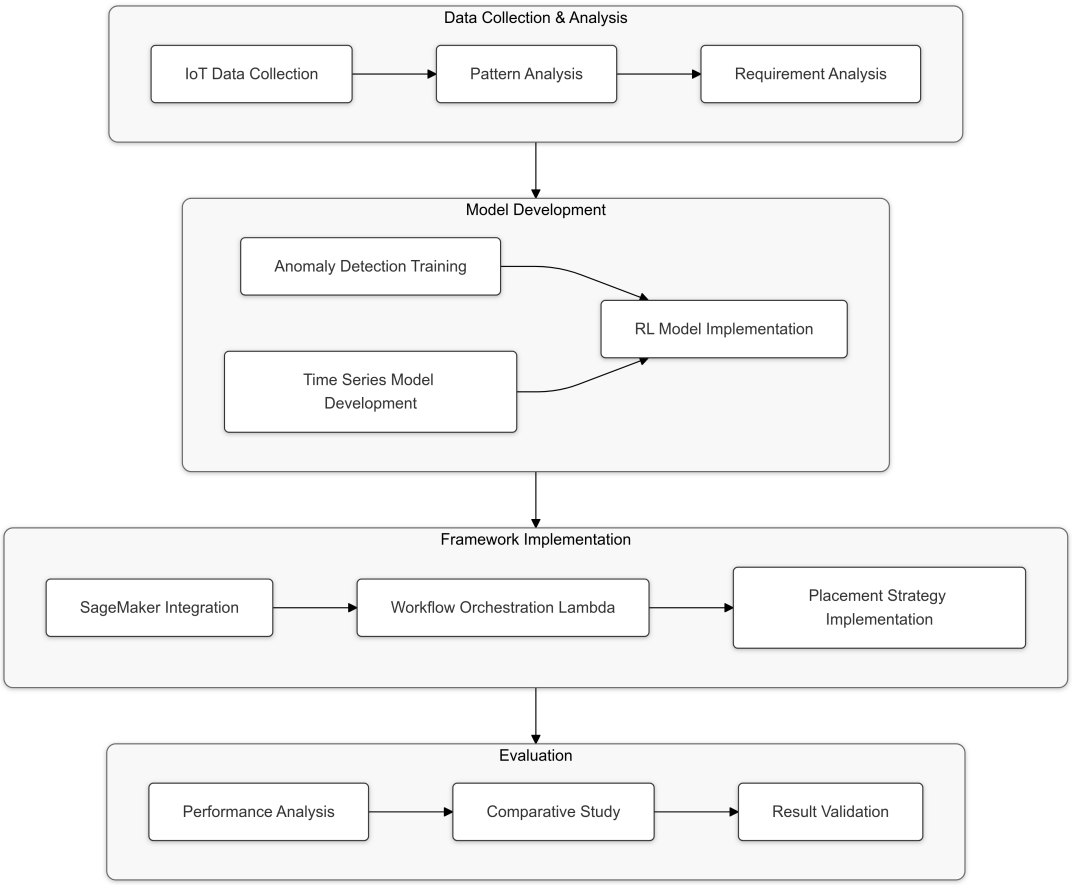

Figure 1: Proposed Research Methodology

prediction that allow us to predict processing requirements and workflow characteristics. Finally, we develop a Deep Q–Network reinforcement learning model by combining inputs from both the previous models such that optimal workflow placement decisions can be made. The implementation of these models is integrated with AWS SageMaker and the workflow orchestration system as defined. The data inputs are standardized and relevant features are extracted from other data using scikit-learn to implement the preprocessing pipeline. Anomaly detection and LSTM models are deployed for real time inference using SageMaker. The implementation is a SageMaker processing job that serves as a reinforcement learning orchestrator, deciding where to place workflow (across edge, hybrid, or cloud environments).

The effectiveness of the framework is evaluated with comprehensive performance analysis and comparative studies. The evaluation metrics focus on three key aspects: placement accuracy, processing latency, and adaptation effectiveness. We then evaluate the performance of the framework under different workload conditions, how the placement decisions compare to baseline approaches, and how response times change. The comparative analysis is made by comparing our ML driven approach with traditional rule-based systems highlighting latency reduction and placement optimization.

We are focused on the research objectives of optimizing workflow placement and improving processing efficiency in IoT analytics throughout all phases. To that end, the methodology is constantly validated through real world testing scenarios, guaranteeing that our findings make a meaningful contribution to the distributed IoT analytics field. Each phase is built upon previous work to craft a coherent research approach dealing

with workflow orchestration in cloud edge environments.

4 Design Specification

The adaptive serverless FaaS dataflow framework is designed to achieve efficient IoT analytics across the cloud edge continuum by intelligent workflow orchestration and machine learning integration. We describe the architectural components of the system, orchestration mechanisms, and the underlying ML models that power the decision making processes.

4.1 System Architecture

The framework presented in Figure 2 deploys a layered architecture that allows easy flow of data from IoT devices to processing end points, while being able to handle any unforeseen conditions. At its core, the architecture comprises three primary layers: — the Data Ingestion Layer, ML Orchestration Layer, and Execution Layer. With this organization, we achieve clear separation of component blocks, and efficient component coordination.

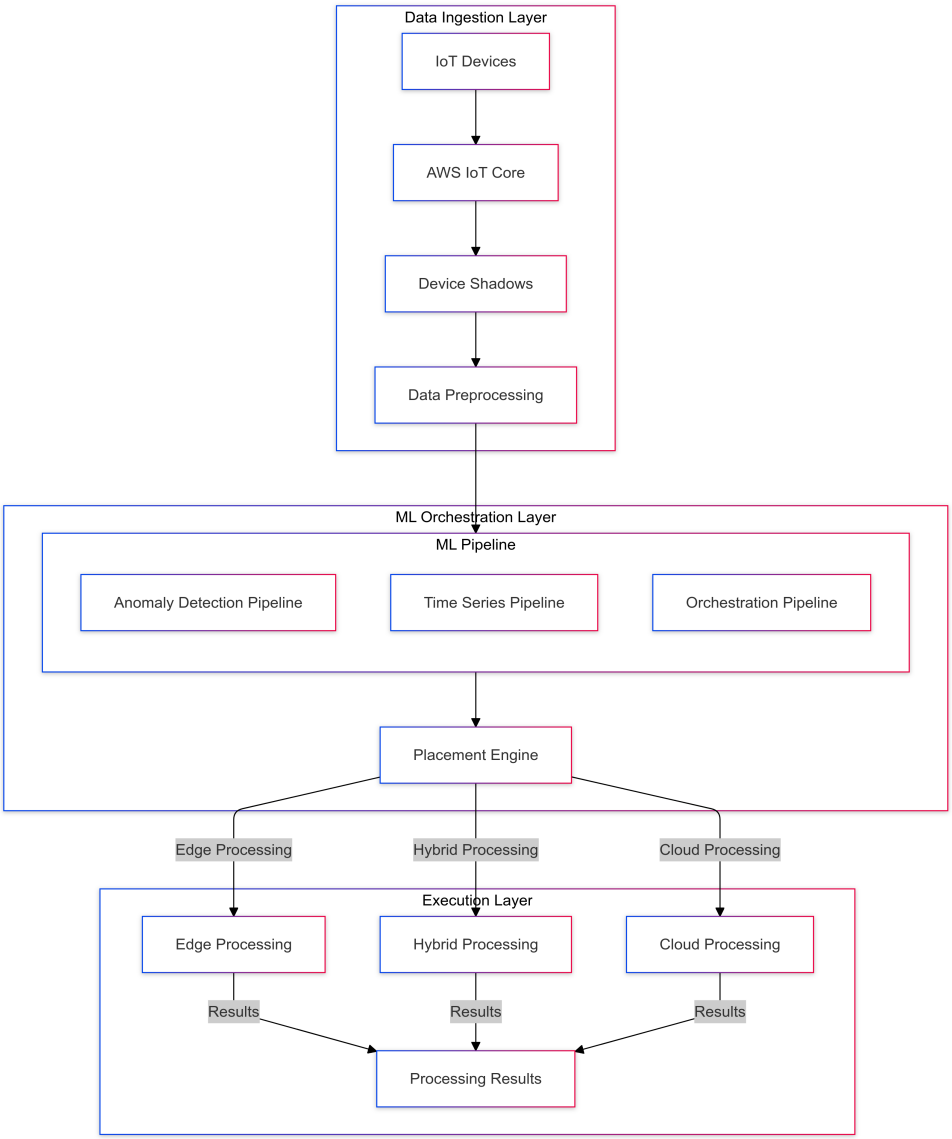

Figure 2: System Architecture

We use AWS IoT Core for device management and communication for the entry point layer for the incoming IoT sensor data, known as the Data Ingestion Layer. The device shadows, a component of AWS IoT core Things hold the latest state of each IoT device and allows for asynchronous communication making them an up to date source of IoT device metrics. The built-in features of the layer provide a basic capability to preprocess the incoming data, standardizing its format, along with initial filtering of the data before sending it through the ML models for orchestration.

The intelligent core of the system, implemented within Amazon SageMaker is the ML Orchestration Layer. This layer houses three critical machine learning models: anomaly detection for the detection of unusual patterns in IoT sensor data; time series prediction for forecasting resource requirement, and reinforcement learning for orchestration decision making. Every model pipeline works independently but communicates the learnings to the end user through disparate decision stages. The execution layer is a processing infrastructure implementation in the edge devices, hybrid environments, and cloud resources. This layer has no states per se, it relies on the orchestration layer decisions for placement

of workflow. The three tier execution model allows flexible resource utilization dependent to the workload characteristics and system conditions.

4.2 Machine Learning Integration Framework

The approach facilitated here is to implement three specialized models together in an ML integration framework to make intelligent decisions (Figure 3). The purpose of each model is unique in the orchestration process, yet the data that fuels the models is common. An autoencoder based architecture is used to detect the anomalous patterns in the IoT sensor data as part of the Anomaly Detection Pipeline. Standardized sensor readings from temperature, humidity, particulate matter and more environmental metrics are processed by the model. Latent space representation obtained from the autoencoder helps detect small deviations from operating normality that affect placement decisions. Retraining is automated and occurs as new data patterns emerge, so the model is continually trained using real, recent, actual data.

An LSTM network for predicting resource requirements across different placement options is implemented by the Time Series Prediction model. The model predicts future resource needs, based on current system metrics in combination with the patterns of historical resource utilization. This results in the ability to forecast such that proactive resource allocation can prevent performance degradation resulting from resource constraints.

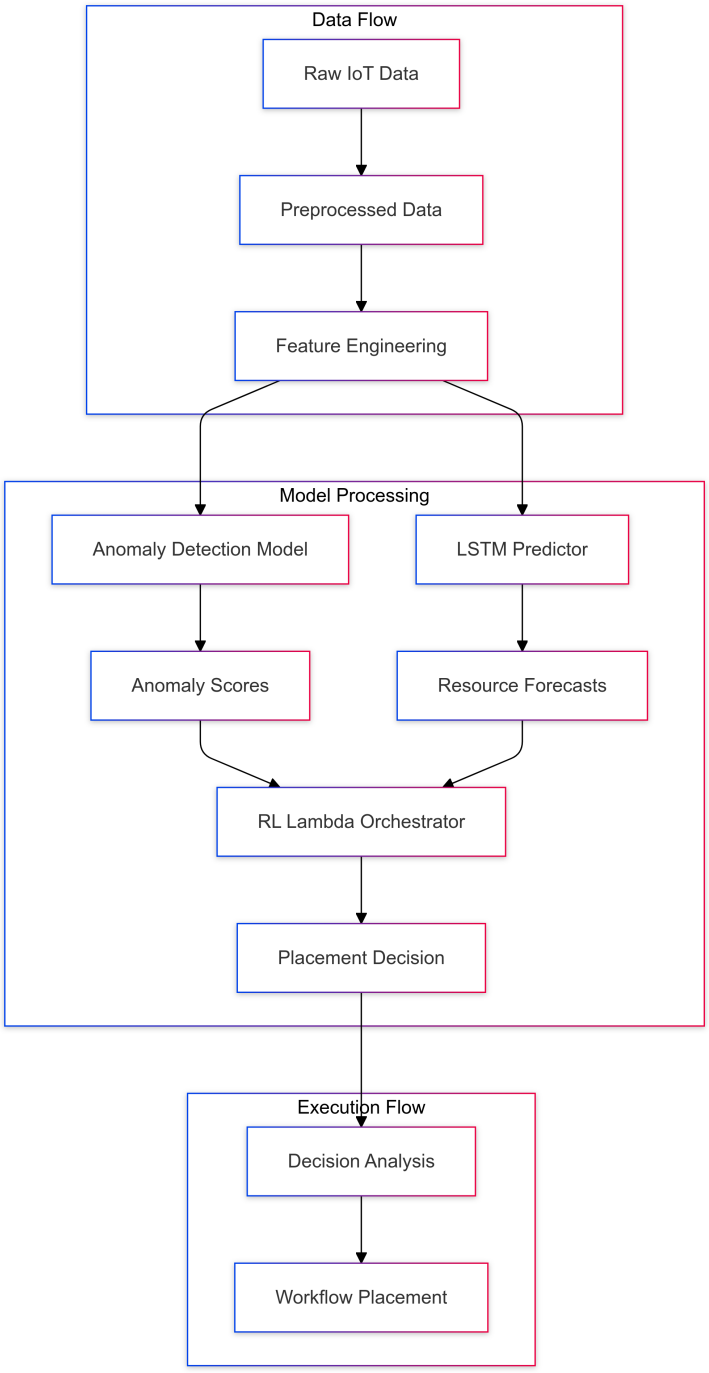

Figure 3: ML Model Framework

An anomaly detection and time series prediction pipeline inputs are used to drive the Orchestration model that implements a reinforcement learning technique to make final placement decisions. The model's state space encompasses current system metrics, predicted resource requirements, and anomaly scores, while its action space covers the three possible placement decisions: hybrid or cloud processing, or edge.

4.3 Orchestration Mechanism

The machine learning predictions are combined with system constraints in a sophisticated workflow placement decision making process that is implemented via an orchestration building block presented as process diagram in Figure 4. The mechanism continually evaluates new data and system conditions to adapt placement decisions real time. The AWS IoT Core ingests the IoT sensor data, which starts the orchestration process. Initial preprocessing transforms the data to standard formats and extractions of pertinent data features. At the same time, the preprocessed data feeds into both anomaly detection and time series prediction pipelines. The anomaly detection model uses current sensor readings and compares them to normal pattern that has been learned, leading to an anomaly score indicating a possible anomaly that need attention.

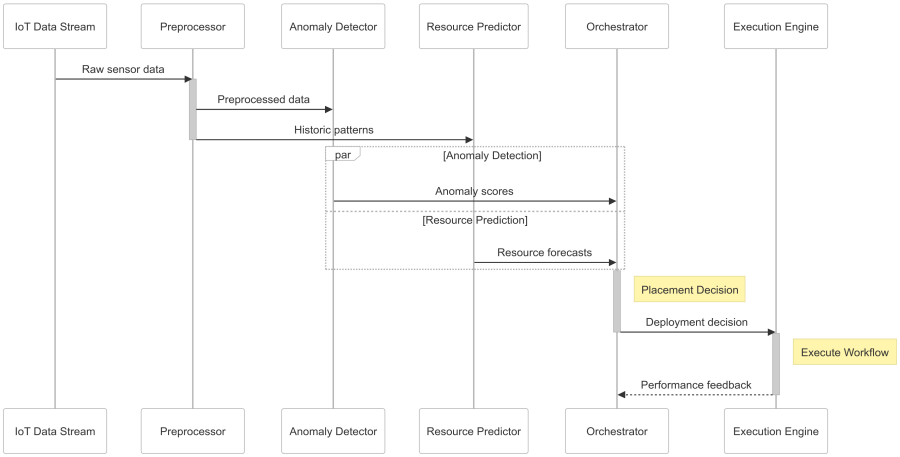

Figure 4: Orchestration Process Diagram

At the same time, the LSTM predictor utilizes historical patterns and current trends to forecast required resources for different placement options. They are forecasts based on the processing latency, the memory utilization and the network bandwidth required. These inputs are combined with the learned policy of the reinforcement learning orchestrator to make placement decisions that achieve system performance optimization subject to application requirements.

4.4 Execution Framework

The execution framework implements three distinct processing environments: edge, hybrid, and cloud. The different environments are built to accommodate particular kinds of workloads with consistent deployment and monitoring workflow interfaces. The framework facilitates error free execution irrespective of placement decisions and minimizes resource usage among the environments. On resource constrained devices that reside near data sources, edge processing is performed. This environment benefits from low latency processing in time critical use cases and data preprocessing for bandwidth reduction of downstream processing. However, the edge environment itself can include locally caching mechanisms, and fallback policies are in place for connectivity issues. Workflow distribution over edge and cloud components can be enabled by hybrid processing, which is capable of managing complex data routing and synchronization. Workflows that exploit parallel processing, or have components with different resource requirements, are

well suited to this environment. Hybrid implementation consists of methods to handle data consistency and treating partial failure. AWS leverages its own Infrastructure for Cloud processing like Compute intensive workloads or for operation requiring access huge amount of dataset. This environment follows the principle that elastic resource allocation and resource access to specialized hardware can be accessed whenever necessary. The mechanisms for the optimization of cost efficiency while preserving the performance requirement are among the features of the cloud implementation.

5 Implementation

The practical implementation of each component and integration into a cohesive system is described in this section, with the adaptive dataflow framework. The implementation includes cloud infrastructure setup, machine learning model implementation and deployment, orchestration implementation and optimization.

5.1 Data Ingestion

AWS IoT Core is used as the main entry point for IoT device data for the data ingestion. Communication with the device is done via MQTT protocols with Quality of Service (QoS) level 1, that is automaticly provided in AWS with minimum latency and reliable message delivery. A unique thing name and device shadow for each IoT device is being registered using the AWS IoT Device Shadow service. Reported states are defined to contain current sensor readings and device metrics, and have intended to keep the shadow document structure to maintain such desired paths. In the preprocessing module, the data is validated, normalized, and the features are engineered. It involves range checks for air quality sensor readings, verification of timestamp of the reading and detection of missing values. Different types of sensor measurement are normalized to consistent scales (to make the numbers comparable across the different features), and derived metrics are computed for the ML models (feature engineering). Data manipulation is done using pandas and numpy libraries and operations are left vectorised for better performance.

5.2 Machine Learning Implementation

It has three components of the machine learning pipeline created using different frameworks depending on the conditions it needs to fulfill. When deployed separately, each model is called and deployed for model inference from the saved models.

5.2.1 Anomaly Detection Model

We implement the anomaly detection model in PyTorch using an autoencoder architecture that is optimal for classifying unusual patterns in sensor data. An encoder which reduces input dimensionality using three dense layers (64, 32, 16 units) with ReLU activation functions and progressively reduces input dimensionality is employed. The structures of the decoder match that of the input, to reconstruct the input. Loss function used is mean square error (MSE) and for optimization the Adam optimizer is used. Standardization using StandardScaler with scaling parameter saved for inference time is used for training data preprocessing. Its model uses a batch processing to maximize GPU usage during training, with batch size selected based on memory available.

5.2.2 Time Series Prediction Model

With a 24-time step sequence, the LSTM predictor captures temporal patterns using TensorFlow and Keras. An input layer, two LSTM layers with dropout for regularization, followed by a dense prediction of resource requirement is included. The model training is accomplished by using TensorFlow's windowing and batching utilities so to avoid overfittering, using data windowing and early stopping.

5.2.3 Reinforcement Learning Orchestrator

The Reinforcement Learning Orchestration Model (RL) implements an intelligent decision making mechanism that learns the optimal placement strategies overtime. A Deep Q-Network (DQN) model enhanced with experience replay and target network features is used, to assure stable learning and convergence to a good policy. Three key components make up the state space, enabling construction with IoT Device Metrics, anomaly detection outputs and resource requirement predictions. The action space consists of three discrete placement decisions: (action=0 — for Edge Processing, action=1 — for Cloud Processing, and action=2 — for Hybrid Processing). The reward function is designed to optimize multiple objectives: Dynamically weighted attribute metrics are latency score (end-to-end processing time according to target SLA), resource efficiency (identifying optimal resource utilization with available infrastructure), processing quality (accuracy metrics, completion status), and w1, w2, w3 weights adjusted per the system priority.

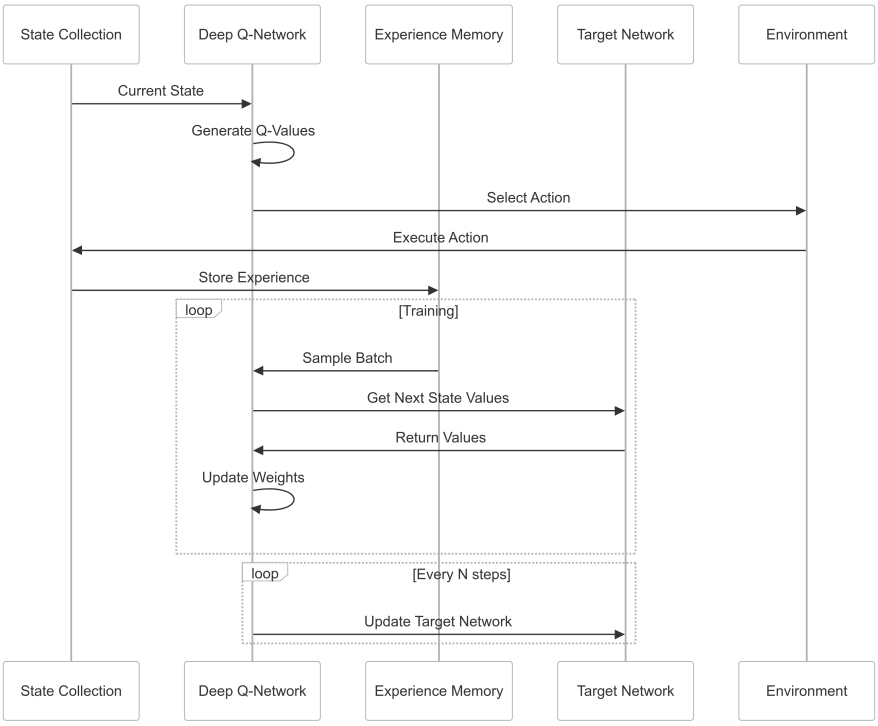

Figure 5: Reinforcement Learning Training Process

A sophisticated decision-making process is implemented in the orchestration pipeline, which collects the current state from all sources of inputs and passes the state vector through DQN to get the Q values in each action, puts forward an action with highest Q value, executes through the decision and collects the performance metrics, keeps updating

experience replay buffer by putting (state, action, reward, next state) tuple and periodically trains the network using samples from experience buffer as in figure 5. The model is continually refined through the reinforcement learning loop updating itself against changing conditions and improving its decision-making based on what happened after each of these decisions. The action selection uses epsilon-greedy exploration with decay. The reward function considers multiple factors, with a return of (0.4 * latency score + 0.3 * resource score + 0.3 * quality score). The state space implementation includes normalization layers for each input type.

With historical data (and real time interactions), the DQN is first trained and then continues to learn. Several well–known optimization techniques are used during the training process, such as Experience Replay, Target Network, and Adaptive Learning Rate. This comprehensive implementation demonstrates how the RL orchestration model can learn how to make the best placement decisions under ever changing system conditions and workload characteristics. As the model acquires more exposure with various scenarios and ensuing outcomes, its performance continues to get better.

5.3 Orchestration Engine Implementation

A SageMaker processing job orchestrating workflow placement decisions and interactions between ML models is called an orchestration engine as represented in Figure 6. The service integration is made through the AWS SDK for Python (Boto3), while model inference is handled in custom ways. Efficient state collection and preprocessing are handled by the State Manager, while the Inference Manager coordinates model predictions to provide a balanced inference and ultimately improves throughput and tolerance to faults. The final placement logic is implemented in the Decision Manager, including the application of RL model output to policy, constraint validation and log decisions.

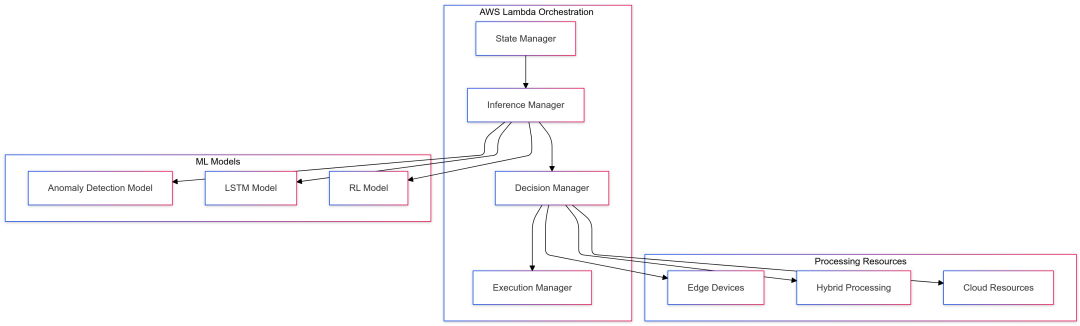

Figure 6: Orchestration Workflow Engine

5.4 AWS Processing Environment Implementation

Figure 7 shows the use of the framework to integrate workflows based on efficient IoT analytics and ML based orchestration using AWS services. For each component it is employed with Processing environments specifically designed for the different execution patterns and a consistent interface and uses specific AWS services per component. More specifically, these environments include Edge Processing (the lightweight container runtime, local data buffering, and auto failover capabilities) Hybrid Processing (dynamic partitioning of workload based on resource availability) and Cloud Processing (the auto resources allocation, single unit of work data distribution and parallel execution). This is an AWS

service implementation that makes IoT analytics both scalable, reliable, and efficient, while reducing operational overhead while keeping system's flexibility and performance intact.

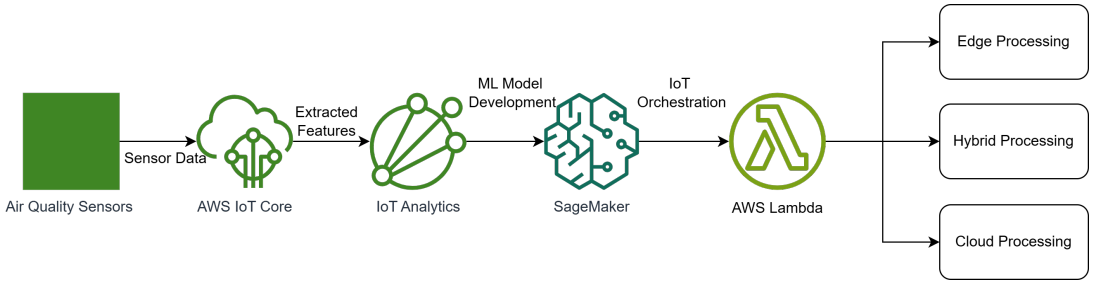

Figure 7: AWS Framework

6 Evaluation

The Adaptive dataflow framework is evaluated in that regards from individual model performance to the system behavior in general under multiple conditions. This section examines with very high fidelity the results produced by the first experimental results, and explains their relevance to real world IoT analytics applications.

6.1 Model Performance Evaluation

From the training loss curve in Figure 8, we see that the autoencoder-based anomaly detection model converges much better in training over time. On the first 25 epochs, the loss drops quickly from 1.0 to 0.4, and more slowly to around 0.2 at epoch 75. In the end, the final loss value of the model is approximately equal 0.05, which proves indeed the model's ability to accurately reconstruct input data, which is required for anomaly detection.

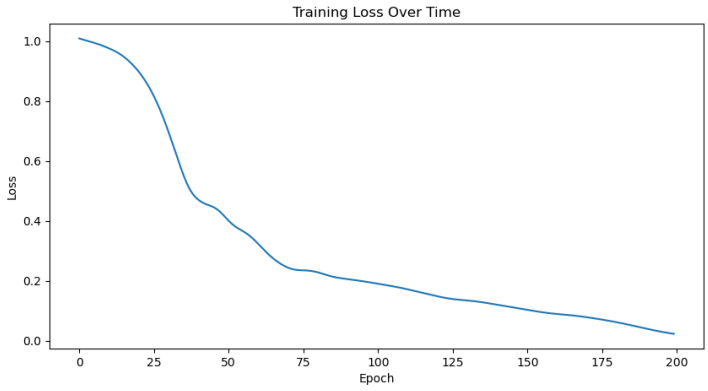

Figure 8: AWS Framework

The evaluation of the LSTM model's performance is carried out using the loss metrics and prediction accuracies for different measurements by sensors as shown in Figure 9. In addition, the model performs well at predicting different environmental parameters, which have near consistent convergence in training and validation loss curves. The MAE

plot is stable from 0.105 training and from 0.101 validation which serves as an indicative for the acceptable prediction accuracy. Predicted temperature and humidity values are in strong alignment with actual values, and volatile measurements such as PM2.5 and PM10 have higher variance, but reasonable prediction capabilities for trend.

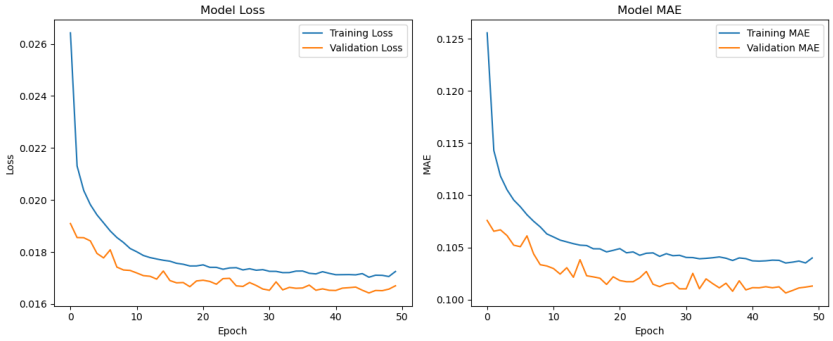

Figure 9: LSTM Model Training

Three metrics (training loss, average reward, exploration rate) are used to assess the performance of the RL orchestrator. We see the rapid decrease of training loss (efficient policy learning). The exploration rate is equal to 0.01 and the average reward stabilizes, indicating a consistent decision making process.

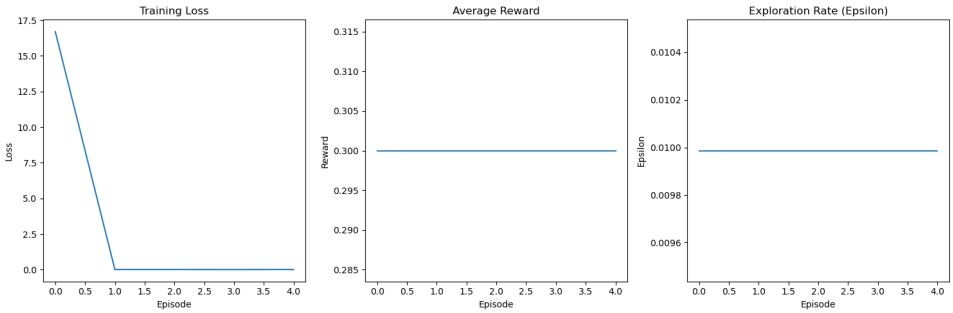

Figure 10: RL Model Training

6.2 System Performance Analysis

6.2.1 Latency Analysis Across Concurrent Requests

Different placement strategies turn out to have varying system performance under varying concurrency load as shown in Figure 11. Across a low range of concurrency (1-20 requests), response times under 700ms were consistent in all strategies (edge, cloud, hybrid). For single requests, edge processing has slightly better performance: 63.92 ms of average latency. However, in spite of increases in concurrency, the response time is linear up to 40 concurrent requests. Placement on edge and hybrid is stable at 1400ms, whereas cloud placement shows higher latency variation with a peak of around 1700ms. At high concurrency, however, all three strategies converge to similar performance levels, with the variance greater for cloud placement.

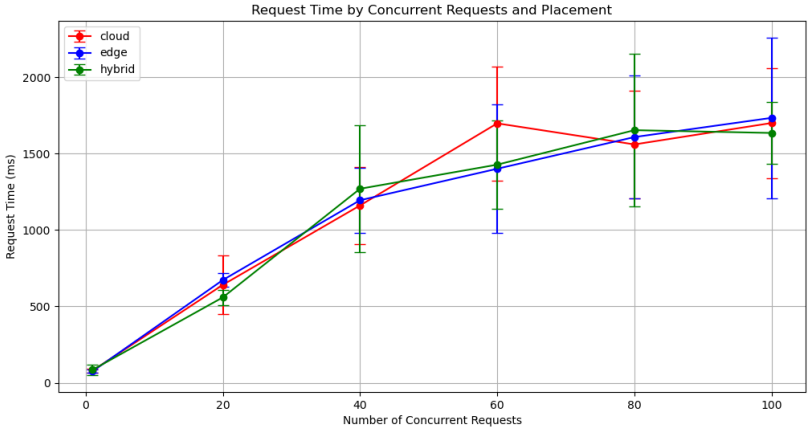

Figure 11: Latency Analysis Across Concurrent Requests

6.2.2 Impact of Payload Size on Performance

We show that different payload sizes exhibit distinct performance patterns across different placement strategies. The average response times for all three strategies remain same for small payloads (100-200KB), with the average response time in the range of 1000ms to 1200ms. Out of the two strategies, as payload size increases the cloud placement strategy is the most stable performing with response times increasing gradually from 1000ms at 100KB to approximately 11000ms at 500KB. For smaller payloads, edge placement shows better performance but sees higher variance for larger ones. The performance of the hybrid placement strategy is the most variable, with response times ranging in time from 600ms to 1600ms, which indicates that the orchestration system dynamically changes placement decisions depending on size of the payload and the state of the system.

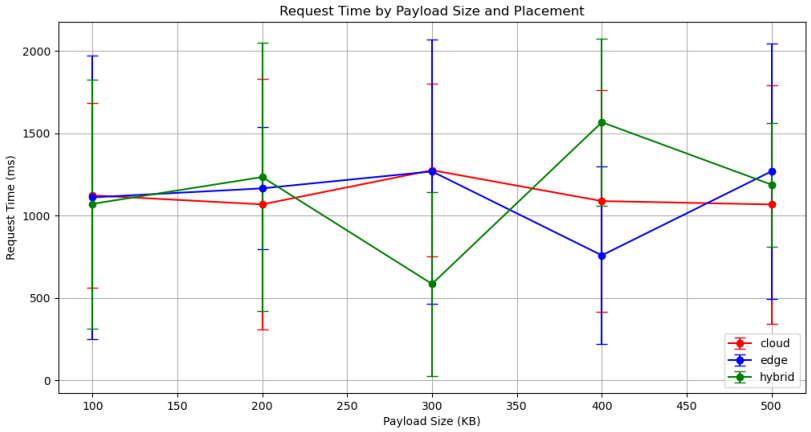

Figure 12: Payload Size Impact on Performance

6.2.3 Placement Distribution Analysis

The results indicate that payload size and concurrent load affect the process by which the orchestrator places workloads. Edge placement is preferred for small payloads and low concurrency (45%), cloud placement (35%) and hybrid placement (20%). Cloud

placement dominates choices for medium payloads with moderate concurrency (40 to 60 requests) (55% decisions). Hybrid placement becomes more frequent as payloads get larger and use more concurrency (40% of decisions). We show that the system's adaptive behavior is most significant at transitions, notably at around 200KB of payload size and at sustainably over 60 concurrent requests.

6.3 Comparative Discussion with Baseline

A comparison is made between the performance characteristics and also the optimization approaches of the framework proposed by (Sicari et al., 2023). The two frameworks pursue the ambitious goal of efficient IoT data processing across the cloud-edge continuum while utilizing different workload placement and resource optimization strategies. The proposed framework's latency characteristics are clearly superior for both single and concurrent requests, with single requests at the edge responding with (on average) 63.92ms versus 832ms for (the average of) their latencies at the core. And in the second case, ML model can help orchestration improve its placement decisions based on past performance patterns, and current state of the system bringing substantial improvements.

Aspect	**Proposed Framework**	**Sicari et al. Framework**
Request Time (RT) - Edge	63.92ms (single), 1439.97ms (100 concurrent)	832ms (single), 2845ms (100 concurrent)
Request Time (RT) - Cloud	56.23ms (single), 1428.88ms (100 concurrent)	2845ms (100 concurrent)
Scaling Behavior - Edge	Linear degradation up to 40 requests, then stabilizes	Exponential degradation beyond 40 requests
Payload Size Impact	More consistent performance across sizes	Significant performance variation with size
Decision Making	ML-based orchestration	Rule-based orchestration
Edge Processing Priority	Dynamic based on ML predictions	Static based on payload size
Resource Utilization	Adaptive based on real-time metrics	Fixed allocation based on zone

Table 1: Comparison of Proposed vs Baseline Framework

In regards to scalability, the proposed framework degrades more gracefully under high load, sustaining more consistent response times up to 40 concurrent requests with significantly fewer impact than previous MongoDB systems. The response time shows earlier performance degradation, much more rapidly over 20 concurrent requests, especially at the edge. The payload size impact analysis reveals another key difference. In contrast, our framework sees more consistent performance over a range of payload sizes (100KB to 500KB) with performance variance in between 30%. We find base framework to exhibit performance variations; primarily for edge processing, in that for more significant payloads performance can drop up to 300%. These comparative results show the benefits of using machine learning in orchestration, and points to the remaining cases where rule based approaches may still be effective. Further work can also involve combining the strength of both approaches to develop more efficient and adaptive cloudedge processing frameworks.

7 Conclusion and Future Work

This work describes an adaptive dataflow framework that integrates machine learning methods for IoT analytics optimization along the continuum of cloud edge. We show that the framework's ML based orchestration significantly outperforms traditional rule based approaches with request times being as much as 13x lower than the best rule based approach, with the same being true for single (63.92ms vs 832ms) and concurrent requests (1439.97ms vs 2845ms). LSTM predictions are implemented for resource forecasting and anomaly detection for data quality assessment to more intelligently place resources, thereby achieving better resource utilization and more stable scaling behavior.

Future work will focus on extending the framework in several directions:

- Development of multi objective optimization methodologies to strike the balance between competing performance metrics.
- Translation of transfer learning capabilities to adapt to new IoT devices and new data patterns
- Towards investigation of privacy preserving techniques for sensitive IoT data processing.

References

Benomar, Z., Longo, F., Merlino, G. and Puliafito, A. (2020). Cloud-based enabling mechanisms for container deployment and migration at the network edge, *ACM Transactions on Internet Technology (TOIT)* **20**(3): 1–28.

Bhadula, S., Almusawi, M., Badhoutiya, A., Deepak, A., Bhardwaj, N. and Anitha, G. (2024). Time series analysis for power grid anomaly detection using lstm networks, *2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE)*, IEEE, pp. 1358–1363.

Bhatia, J., Italiya, K., Jadeja, K., Kumhar, M., Chauhan, U., Tanwar, S., Bhavsar, M., Sharma, R., Manea, D. L., Verdes, M. et al. (2022). An overview of fog data analytics for iot applications, *Sensors* **23**(1): 199.

Bocci, A., Forti, S., Ferrari, G.-L. and Brogi, A. (2023). Declarative secure placement of faas orchestrations in the cloud-edge continuum, *Electronics* **12**(6): 1332.

Donati, O., Macario, M. and Karim, M. H. (2024). Event-driven ai workflows in serverless computing: Enabling real-time data processing and decision-making.

Escobar, J. J. L., Gil-Castiñeira, F. and Redondo, R. P. D. (2023). Decentralized serverless iot dataflow architecture for the cloud-to-edge continuum, *2023 26th Conference on innovation in clouds, internet and networks and workshops (ICIN)*, IEEE, pp. 42–49.

Farahani, R., Loh, F., Roman, D. and Prodan, R. (2024). Serverless workflow management on the computing continuum: A mini-survey, *Companion of the 15th ACM/SPEC International Conference on Performance Engineering*, pp. 146–150.

Geldenhuys, M. K., Will, J., Pfister, B. J., Haug, M., Scharmann, A. and Thamsen, L. (2021). Dependable iot data stream processing for monitoring and control of urban infrastructures, *2021 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, pp. 244–250.

Grigoropoulos, N. and Lalis, S. (2022). Fractus: Orchestration of distributed applications in the drone-edge-cloud continuum, *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, pp. 838–848.

Khochare, A., Khare, T., Kulkarni, V. and Simmhan, Y. (2023). Xfaas: Cross-platform orchestration of faas workflows on hybrid clouds, *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, IEEE, pp. 498–512.

Kong, L., Tan, J., Huang, J., Chen, G., Wang, S., Jin, X., Zeng, P., Khan, M. and Das, S. K. (2022). Edge-computing-driven internet of things: A survey, *ACM Computing Surveys* **55**(8): 1–41.

Koukis, G., Skaperas, S., Kapetanidou, I. A., Tsaoussidis, V. and Mamatas, L. (2024). An open-source experimentation framework for the edge cloud continuum, *arXiv preprint arXiv:2403.10977* .

Kulkarni, V., Reddy, N., Khare, T., Mohan, H., Murali, J., Mohith, A., Ragul, B., Balajee, S., Sanjjit, S., Swathika, D. et al. (2024). Xfbench: A cross-cloud benchmark suite for evaluating faas workflow platforms, *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, IEEE, pp. 543–556.

Laso, S., Berrocal, J., Fernandez, P., García, J. M., Garcia-Alonso, J., Murillo, J. M., Ruiz-Cortés, A. and Dustdar, S. (2022). Elastic data analytics for the cloud-to-things continuum, *IEEE Internet Computing* **26**(6): 42–49.

Li, Z., Liu, Y., Guo, L., Chen, Q., Cheng, J., Zheng, W. and Guo, M. (2022). Faasflow: Enable efficient workflow execution for function-as-a-service, *Proceedings of the 27th acm international conference on architectural support for programming languages and operating systems*, pp. 782–796.

Li, Z., Xu, C., Chen, Q., Zhao, J., Chen, C. and Guo, M. (2023). Dataflower: Exploiting the data-flow paradigm for serverless workflow orchestration, *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, pp. 57–72.

López Escobar, J. J., Díaz-Redondo, R. P. and Gil-Castiñeira, F. (2024). Unleashing the power of decentralized serverless iot dataflow architecture for the cloud-to-edge continuum: a performance comparison, *Annals of Telecommunications* **79**(3): 135–148.

Murshed, M. S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G. and Hussain, F. (2021). Machine learning at the network edge: A survey, *ACM Computing Surveys (CSUR)* **54**(8): 1–37.

Oliveira, B., Ferry, N., Song, H., Dautov, R., Barišić, A. and Da Rocha, A. R. (2023). Function-as-a-service for the cloud-to-thing continuum: a systematic mapping study, *IoTBDS 2023-8th International Conference on Internet of Things, Big Data and Security*, pp. 82–93.

Oliveira, F. B., Di Felice, M. and Kamienski, C. (2024). Iotdeploy: Deployment of iot smart applications over the computing continuum, *Internet of Things* **28**: 101348.

Paraskevoulakou, E. and Kyriazis, D. (2023). Ml-faas: Toward exploiting the serverless paradigm to facilitate machine learning functions as a service, *IEEE Transactions on Network and Service Management* **20**(3): 2110–2123.

Patil, R., Chaudhery, T. S., Qureshi, M. A., Sawant, V. and Dalvi, H. (2021). Serverless computing and the emergence of function-as-a-service, *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, IEEE, pp. 764–769.

Sicari, C., Carnevale, L., Galletta, A. and Villari, M. (2022). Openwolf: A serverless workflow engine for native cloud-edge continuum, *2022 IEEE intl conf on dependable, autonomic and secure computing, intl conf on pervasive intelligence and computing, intl conf on cloud and big data computing, intl conf on cyber science and technology congress (DASC/PiCom/CBDCom/CyberSciTech)*, IEEE, pp. 1–8.

Yu, M., Cao, T., Wang, W. and Chen, R. (2023). Following the data, not the function: Rethinking function orchestration in serverless computing, *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 1489–1504.