الجزائرية الديمقراطية الشعبية الجمهورية
People's Democratic Republic of Algeria
العلمي البحث و العالي التعليم وزارة
Ministry of Higher Education and Scientific Research
بلعباس سيدي - 5491 ماي 8 الآلي للإعلام العليا المدرسة
Higher School of Computer Science
8 Mai 1945 - Sidi Bel Abbes

Master's Thesis

To obtain the diploma of [**Engineering/Master's**] **Degree**

Field of Study: **Computer Science**

Specialization: [**does this work chat ?**]

Theme

[Thesis Title]

Presented by
[**Your Name**]
[**Your Name**]

Defended on: [**Month, Year**]
In front of the jury composed of

Mr. [**Jury Member Name**]	President of the Jury
Mr. [**Jury Member Name**]	Thesis Supervisor
Mr. [**Jury Member Name**]	Co-Supervisor
Mr. Test Member Name]	Examiner

Academic Year: 20XX/20XX

Acknowledgement

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords— keyword 1, keyword 2

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

GENERAL INTRODUCTION

1.1 Introduction

Serverless computing or Function-as-a-Service (FaaS) is an increasingly popular model for delivering cloud services [Bal+17]. Serverless removes the burden of the management of the infrastructure: Scaling, security and networking, to allow developers to focus on the business logic of their applications. In this model, applications take the form of short-lived, event-triggered functions that incur costs only during function execution. vanced AI capabilities accessible to a broader range of applications.

Applying the FaaS model to edge computing has recently received significant interest in research as well as industry [Asl+21]. Edge computing refers to taking advantage of computational nodes located at the edge of network. By allocating resources only during function execution, FaaS assures reduced resource consumption, which is essential for the resource-constrained nature of edge nodes. The potential for deploying AI functions at the edge opens up excitininteresting usecases cameras detecting anomalies, or IoT sensors predicting equipment failures before they occur or agriculture monitoring systems analyzing soil conditions, all with minimal latency and bandwidth usage.

A main challenge for researchers and practitioners seeking to deploy the FaaS model to edge scenarios is the shortage of simulation tools. Designing and building effective resource management solutions for serverless environments requires extensive long-term testing, experimentation, and analysis of performance metrics. However, utilizing real test beds and serverless platforms for such experimentation is often not feasible due to resource, time, and cost constraints [MB21]. This challenge is particularly acute when evaluating novel ideas for resource management, function scheduling, and scaling policies in edge-FaaS environments.

Specifically, although there is a wide range of simulation tools targeting edge and fog scenarios [Svo+19], there are only a few tools focusing on the FaaS model [MK21] and, in particular, on executing FaaS applications in edge environments **jeon2019cloudsim**. Existing simulation software developed for serverless environments often lack generalizability in their architecture and fail to comprehensively address various aspects of resource management, with most being purely focused on modeling function performance under specific platform architectures [MB21]. Moreover, these tools typically do not provide adequate support for the unique characteristics of edge computing, such as network latency variations, resource heterogeneity, and intermittent connectivity patterns.

The complexity of this environment includes heterogeneous devices, varying network conditions, resource constraints, and diverse workload patterns. This requires sophisticated simulation capabilities that can accurately model these interactions. Understanding the trade-offs, performance characteristics, and optimization opportunities in such environments requires comprehensive evaluation of the frameworks to identify what existing tools miss and what they do not provide adequately.

1.2 Objective

The aim of this thesis is to explore simulators for FaaS deployed at the edge, Analyze simulation frameworks for such environments with emphasis on factors that make them most suitable for edge scenarios.

This main goal is achieved through the following research objectives:

1. State-of-the-Art Analysis: Conduct a survey of existing FaaS simulation tools, analyzing both cloud-centric and edge-oriented frameworks to understand their features, architectures, limitations, and applicability to edge scenarios.

2. Comparative Study Framework: Establish a comparative study based on critical factors including edge computing support, IoT workload modeling capabilities, resource usage tracking, network modeling accuracy, and system configurability.

3. Simulation Assessment: Perform detailed analysis of identified simulation frameworks, evaluating their strengths and weaknesses across the established criteria to determine their suitability for different edge-FaaS research scenarios.

4. Framework Selection: Provide evidence-based recommendations for researchers and practitioners seeking appropriate simulation tools for edge-FaaS studies, considering different research objectives and application domains.

Through these objectives, this thesis aims to contribute to the advancement of edge-FaaS research by providing a foundation for tool selection and highlighting critical areas for future simulation framework development.

1.3 Thesis Structure

This thesis is organized into four main chapters that address the research objectives and provide coverage of the edge-FaaS simulation landscape:

Chapter 1: General Introduction provides the contextual background, motivation, and research objectives that guide this study. It establishes the importance of simulation tools and outlines the thesis structure and methodology.

Chapter 2: Basic Concepts presents the theoretical foundation necessary for understanding the research domain. This chapter covers fundamental concepts in edge computing, including architecture, resource constraints, and deployment challenges. It explores the FaaS paradigm, examining serverless computing models, container lifecycle management, and cold start mechanisms. The chapter also addresses the integration of FaaS with edge environments, discussing both opportunities and challenges. Additionally, it covers IoT applications, simulation principles, and performance evaluation methodologies that are essential for understanding the comparative analysis presented in subsequent chapters.

Chapter 3: State of the Art constitutes the core contribution of this thesis, presenting an analysis of existing FaaS simulation frameworks. The chapter begins with a review of cloud-centric simulators (MFS, ServerlessSimPro, SimFaaS, CloudSimSC) followed by edge-focused tools (EdgeFaaS, faas-sim). It establishes a rigorous analysis framework with clearly defined evaluation criteria and assessment methodologies. The chapter provides detailed evaluation results for each simulator, analyzing their capabilities in modeling edge computing environments, resource usage, network modeling, and system configurability. The chapter concludes with a synthesis of findings, identification of research gaps, and a detailed discussion of simulator selection rationales.

Chapter 4: Conclusion synthesizes the research findings and contributions of this thesis. It summarizes key discoveries from the simulator comparison, highlighting the strengths and limitations of current tools. The chapter presents the main research contributions, including the evaluation framework, analysis results, and identified research gaps. It acknowledges the limitations of the current study and outlines promising directions for future research, including advanced simulation model development, real-world validation studies, and extended application scenarios.

This structure provides coverage of the research domain while maintaining focus on the primary objective of providing a detailed analysis of edge-FaaS simulation tools for the research community.

CHAPTER 2

BASIC CONCEPTS

2.1 Introduction

This chapter presents the fundamental concepts necessary for understanding FaaS simulation at the edge. It covers edge computing principles, the FaaS model, their integration challenges, IoT applications, and simulation methodologies. These concepts are necessary to have a foundation for analyzing and comparing simulation frameworks in subsequent chapters.

2.2 Edge Computing Fundamentals

2.2.1 Definition and Characteristics

Edge computing brings computation and data storage closer to the location where data is generated and consumed [Asl+21]. Unlike traditional cloud computing, which centralizes processing in remote data centers, edge computing distributes computational resources across geographically dispersed nodes at the network edge.

Key characteristics of edge computing include:

- **Low Latency**: Computation is close to data sources, reducing communication delays
- **Limited Resources**: Edge nodes typically have constrained CPU, memory, and storage
- **Geographic Distribution**: Resources span multiple locations with varying capabilities
- **Network Variability**: Connection quality and bandwidth fluctuate across different edge locations

2.2.2 Edge Computing Architecture

The edge computing architecture consists of three main layers:

Device Layer: IoT sensors, smartphones, and embedded devices that generate data and may perform basic processing.

Edge Layer: Intermediate nodes including gateways, micro data centers, and base stations that provide computational resources closer to devices.

Cloud Layer: Centralized data centers that handle complex processing tasks and provide unlimited storage capacity.

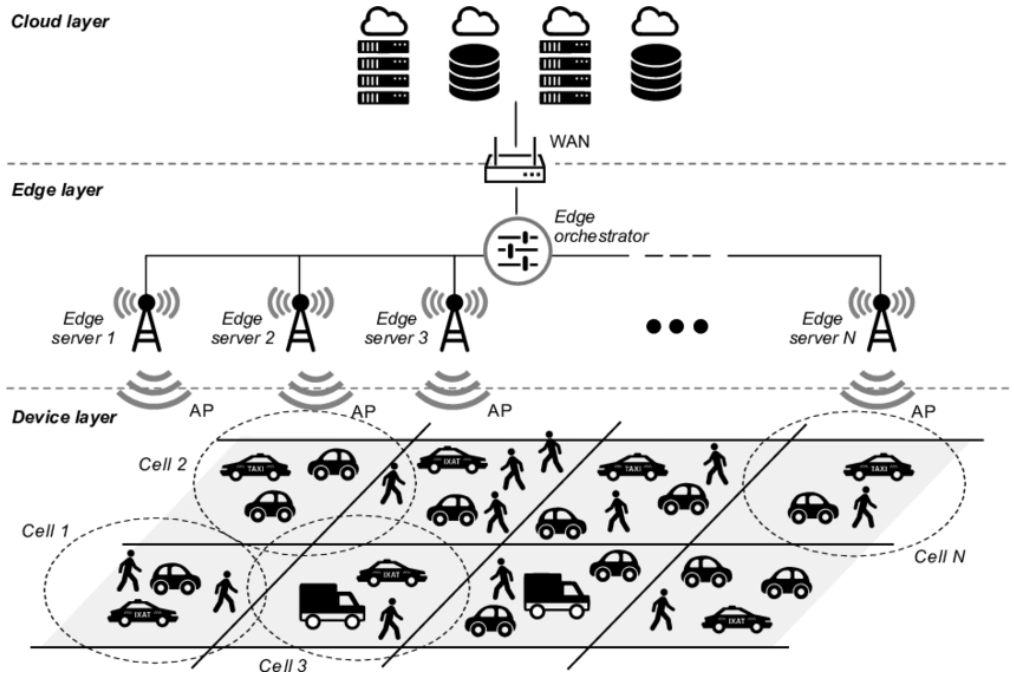

Figure 2.1: Edge Computing Architecture [Bel+23]

2.2.3 Edge vs Cloud Computing Trade-offs

Edge computing offers several advantages over pure cloud solutions, but also introduces new challenges when integrating with FaaS [JYZ19]:

Opportunities:

- **Improved Latency and Bandwidth Efficiency**: Co-locating compute with data sources reduces round-trip time and traffic to the cloud, enabling faster responses for real-time applications
- **Context-Awareness and Locality**: Utilizing local contextual information for better service personalization and quality through proximity to data sources
- **Elastic and Event-Driven Edge Processing**: FaaS abstracts lifecycle management and autoscaling, simplifying deployment at resource-constrained edge nodes
- **Heterogeneous Resource Utilization**: Function granularity allows optimal usage of diverse and constrained edge resources across different device capabilities
- **Enhanced Privacy and Security**: Function isolation minimizes exposure of sensitive user data compared to monolithic applications
- **Cost Optimization**: Local execution reduces reliance on expensive cloud resources and network bandwidth usage

Challenges:

- **Resource Management**: Edge resources are limited and heterogeneous, requiring fine-grained scheduling, placement, and elasticity across dynamic topologies
- **Function Deployment and Orchestration**: Cold start latency is significantly worse at the edge due to resource constraints, and function migration across multiple edge nodes is complex due to decentralization

- **Security and Privacy**: New attack surfaces arise from multi-tenancy and limited isolation capabilities on resource-constrained devices
- **Programming Model and Abstractions**: Lack of programming models tailored for distributed, dynamic, and latency-sensitive FaaS deployments at the edge
- **Monitoring and Debugging**: Fine-grained observability is challenging due to ephemeral, distributed function executions across heterogeneous infrastructure
- **Performance-Isolation Trade-offs**: Balancing execution speed with security isolation remains unresolved, especially for time-sensitive applications requiring both performance and security

2.2.4 Resource Constraints in Edge Environments

Edge nodes face significant resource limitations compared to cloud data centers. These constraints directly impact application deployment and performance:

Computational Constraints: Edge devices typically have limited CPU cores and processing power, requiring efficient resource allocation and task scheduling.

Memory Limitations: Available RAM is often restricted, affecting the number of concurrent applications and data caching capabilities.

Storage Restrictions: Local storage is limited, requiring careful data management and potential offloading to cloud storage.

Energy Constraints: Many edge devices operate on battery power or have strict energy budgets, necessitating energy-efficient computing strategies.

2.3 Function-as-a-Service (FaaS)

2.3.1 Serverless Computing Model

Serverless computing represents a cloud computing model where developers focus solely on writing code without managing the underlying infrastructure [Bal+17]. The cloud provider handles server provisioning, scaling, and maintenance automatically.

Core principles of serverless computing:

- **No Server Management**: Infrastructure is completely abstracted from developers
- **Automatic Scaling**: Resources scale up or down based on demand
- **Pay-per-Use**: Costs are incurred only during actual code execution
- **Event-Driven**: Functions execute in response to specific triggers or events

2.3.2 FaaS Architecture and Characteristics

Function-as-a-Service (FaaS) is the core component of serverless computing, allowing developers to deploy individual functions that execute in response to events [MB21].

Key FaaS characteristics include:

- **Stateless Functions**: Each function execution is independent
- **Short-lived Execution**: Functions run for brief periods (typically seconds to minutes)
- **Event-triggered**: Functions respond to HTTP requests, database changes, file uploads, etc.
- **Language Agnostic**: Support for multiple programming languages

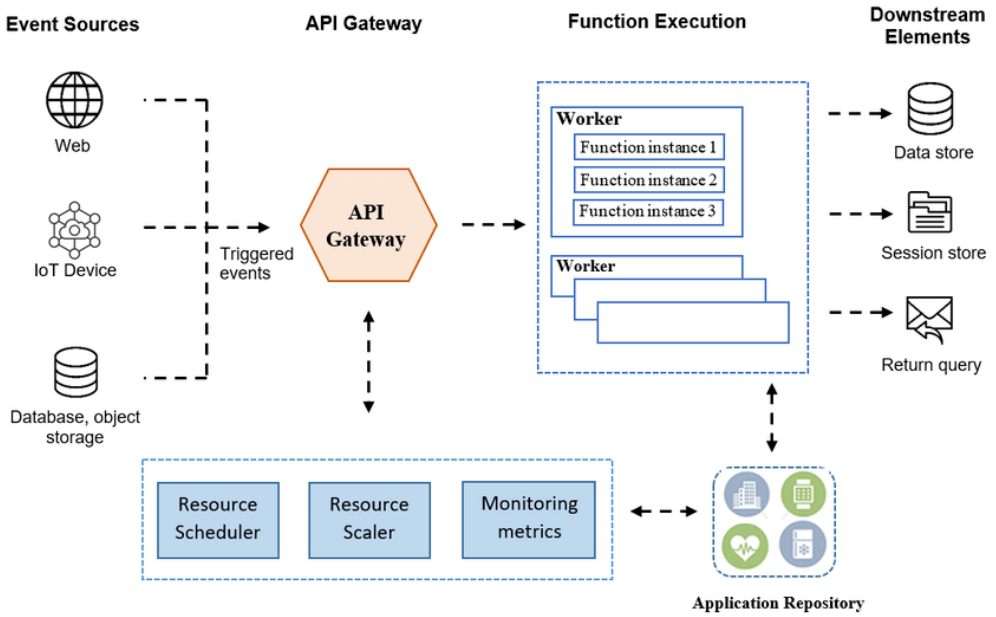

Figure 2.2: FaaS Architecture [MKB21]

2.3.3 Container Lifecycle Management

FaaS platforms use containers to provide isolated execution environments for functions. The container lifecycle involves several stages:

Container Creation: New containers are instantiated when functions are first invoked or when existing containers are unavailable.

Function Execution: The function code executes within the container environment with allocated resources.

Container Reuse: Containers may be kept warm between invocations to reduce startup overhead.

Container Termination: Idle containers are eventually terminated to free resources.

2.3.4 Cold Start and Warm Start Mechanisms

Function invocation performance is significantly affected by container initialization overhead:

Cold Start: Occurs when a new container must be created for function execution. This involves downloading the function code, initializing the runtime environment, and allocating resources. Cold starts introduce latency penalties ranging from hundreds of milliseconds to several seconds.

Warm Start: Happens when an existing container can be reused for function execution. Warm starts have minimal latency since the runtime environment is already initialized.

The trade-off between cold and warm starts affects both performance and cost, as keeping containers warm consumes resources but reduces execution latency.

2.4 FaaS Integration at the Edge

2.4.1 Opportunities in FaaS-Edge Convergence

Combining FaaS with edge computing creates several opportunities [Asl+21]:

Resource Efficiency: FaaS's pay-per-use model aligns well with edge computing's resource constraints, as resources are allocated only when needed.

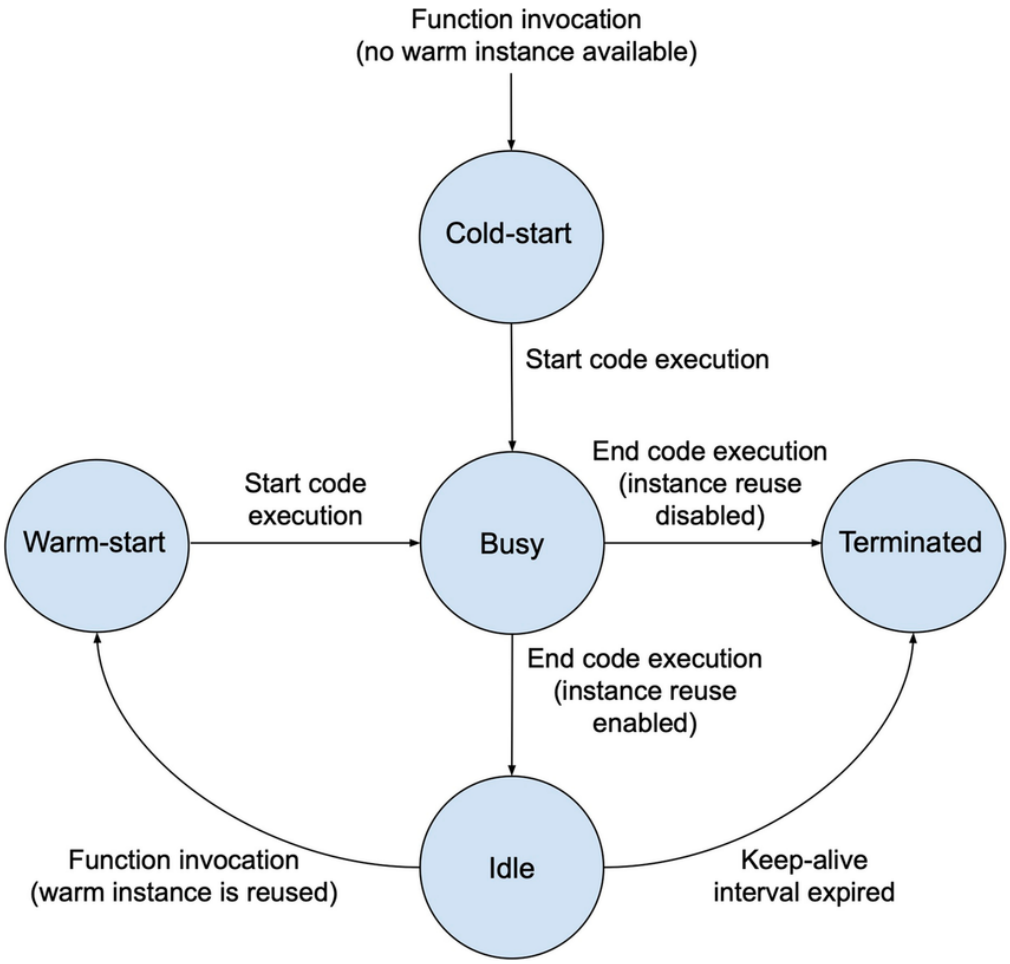

Figure 2.3: Container Lifecycle in FaaS [Mor+23]

Scalability: Automatic scaling capabilities help manage varying workloads across distributed edge nodes.

Rapid Deployment: Functions can be quickly deployed to multiple edge locations without manual infrastructure management.

Cost Optimization: The serverless billing model reduces costs for intermittent edge workloads.

2.4.2 Challenges in FaaS-Edge Deployments

Despite the opportunities, several challenges exist when deploying FaaS at the edge:

Cold Start Penalties: Limited edge resources make cold start overhead more significant, as container initialization competes with application workloads for scarce resources.

Resource Heterogeneity: Edge nodes vary in computational capabilities, requiring intelligent function placement and resource allocation strategies.

Network Reliability: Intermittent connectivity between edge nodes and cloud infrastructure affects function deployment and management.

State Management: The stateless nature of FaaS conflicts with edge applications that require local state persistence.

2.4.3 Heterogeneous Device Management

Edge environments consist of diverse devices with varying capabilities:

High-end Edge Servers: Powerful machines with substantial CPU, memory, and storage resources capable of running multiple functions simultaneously.

IoT Gateways: Intermediate devices with moderate resources that aggregate data from sensors and perform basic processing.

Embedded Devices: Resource-constrained devices with limited computational capabilities suitable only for lightweight functions.

Managing function deployment across this heterogeneous infrastructure requires sophisticated scheduling and resource allocation algorithms.

2.4.4 Network Dynamics and Latency Considerations

Network characteristics at the edge differ significantly from cloud environments:

Variable Latency: Network delays fluctuate based on location, connection type, and network load.

Bandwidth Limitations: Edge connections often have lower bandwidth than cloud data center links.

Intermittent Connectivity: Mobile and remote edge nodes may experience periodic disconnections.

Geographic Distribution: Wide-area networks introduce additional latency and reliability concerns.

Figure 2.4: Placeholder : Network Dynamics in Edge-FaaS ?

2.5 Internet of Things (IoT) and Smart Cities

2.5.1 IoT Architecture and Communication Models

IoT networks consist of interconnected devices that collect, process, and exchange data. The typical IoT architecture includes:

Perception Layer: Physical sensors and actuators that interact with the environment.

Network Layer: Communication infrastructure including WiFi, cellular, and low-power wide-area networks.

Middleware Layer: Platforms that manage device connectivity, data processing, and application interfaces.

Application Layer: End-user applications and services that consume IoT data.

2.5.2 Smart City Applications and Use Cases

Smart cities leverage IoT and edge computing to improve urban services:

Traffic Management: Real-time traffic monitoring using sensors and cameras to optimize signal timing and route planning.

Environmental Monitoring: Air quality sensors and weather stations provide data for pollution control and climate management.

Public Safety: Surveillance cameras and emergency response systems enhance security and incident response.

Energy Management: Smart grid systems optimize power distribution and consumption across urban infrastructure.

Figure 2.5: Placeholder Smart City Applications - illustrating IoT devices, edge computing nodes, and FaaS functions for various urban services

2.5.3 IoT Workload Characteristics

IoT applications exhibit specific workload patterns that affect FaaS deployment:

Event-driven Processing: Data processing occurs in response to sensor readings or environmental changes.

Periodic Tasks: Regular monitoring and maintenance functions execute on scheduled intervals.

Burst Traffic: Sudden spikes in activity during emergencies or special events.

Geographic Distribution: Workloads are distributed across multiple locations based on sensor placement.

2.5.4 Real-time Processing Requirements

Many IoT applications have strict timing constraints:

Hard Real-time: Safety-critical applications require guaranteed response times (e.g., autonomous vehicle controls).

Soft Real-time: Applications tolerate occasional deadline misses but prefer timely responses (e.g., traffic optimization).

Near Real-time: Applications require fast but not instantaneous responses (e.g., environmental monitoring).

These requirements influence function placement, resource allocation, and scheduling decisions in edge-FaaS environments.

2.6 Simulation and Modeling Principles

2.6.1 Discrete Event Simulation

Discrete event simulation models systems as sequences of events occurring at specific time points [MK21]. This approach is well-suited for modeling FaaS systems where function invocations, container lifecycle events, and resource allocation decisions occur at discrete time intervals.

Key components of discrete event simulation:

- **Events**: Function invocations, container starts/stops, resource allocation changes
- **State Variables**: Resource utilization, queue lengths, function response times
- **Event List**: Chronologically ordered future events
- **Simulation Clock**: Current simulation time

2.6.2 Trace-driven vs Model-driven Approaches

Simulation frameworks employ different strategies for generating workloads:

Trace-driven Simulation: Uses real-world traces of function invocations, resource usage, and network behavior to replay historical workloads. This approach provides realistic workload patterns but is limited to historical scenarios.

Model-driven Simulation: Employs mathematical models to generate synthetic workloads based on statistical distributions and behavioral patterns. This approach enables exploration of hypothetical scenarios but may not capture all real-world complexities.

Hybrid Approaches: Combine trace-driven and model-driven techniques to leverage the benefits of both approaches.

2.6.3 Performance Metrics and Validation

Simulation frameworks must provide meaningful performance metrics and validation mechanisms:

Response Time Metrics: Function execution time, end-to-end latency, cold start overhead.

Resource Utilization: CPU, memory, and storage usage across edge nodes.

Throughput Measures: Function invocations per second, data processing rates.

Cost Metrics: Resource costs, energy consumption, operational expenses.

Reliability Indicators: Success rates, error frequencies, availability measures.

Validation involves comparing simulation results with real-world measurements or analytical models to ensure accuracy and reliability.

Figure 2.6: Placeholder : Simulation Validation Process - depicting the cycle of model development, validation against real systems, and iterative refinement

2.7 Conclusion

This chapter has established the fundamental concepts necessary for understanding FaaS simulation at the edge. Edge computing provides distributed computational resources with inherent constraints that affect application deployment and performance. The FaaS model offers an event-driven, serverless approach that aligns well with edge computing principles but introduces challenges related to cold starts and resource heterogeneity.

The integration of FaaS with edge computing creates opportunities for efficient resource utilization and rapid application deployment, particularly for IoT and smart city applications. However, this integration also presents challenges in managing heterogeneous resources, variable network conditions, and real-time processing requirements.

Simulation and modeling principles provide the foundation for evaluating FaaS-edge systems through discrete event simulation, various workload generation approaches, and comprehensive performance metrics. These concepts form the basis for analyzing and comparing simulation frameworks in the next chapter.

Part I

State of the art

CHAPTER 3

STATE OF THE ART

3.1 Introduction

Having established the theoretical foundations and integration challenges of FaaS-edge computing in the previous chapter, this chapter presents a systematic evaluation of existing simulation frameworks designed for this domain. While the motivation for FaaS-edge simulation has been established, the research community currently lacks a comprehensive comparative analysis to guide tool selection for specific research objectives.

This chapter addresses this gap through a structured analysis of six prominent simulation frameworks, categorized into cloud-centric tools (ServerlessSimPro, MFS, CloudSimSC) and edge-oriented platforms (faas-sim, EdgeFaaS, SimFaaS). The evaluation employs a novel five-criteria assessment framework encompassing resource usage modeling, edge support capabilities, network modeling sophistication, configurability, and validation accuracy.

The analysis culminates in evidence-based recommendations for framework selection, identifies critical research gaps in current simulation capabilities, and establishes faas-sim as the optimal choice for comprehensive edge-FaaS research based on quantitative evaluation results.

3.2 FaaS Simulation Frameworks

3.2.1 Cloud-Centric FaaS Simulators

Cloud-centric simulators primarily target traditional cloud environments with abundant computational resources, focusing on scalability, cost optimization, and performance analysis in centralized data center deployments.

3.2.1.1 ServerlessSimPro

ServerlessSimPro represents a comprehensive cloud-centric simulation platform designed for realistic serverless environment modeling [DPW22]. Built with a three-tier architecture encompassing Physical Machines (PMs), containers, and functions, the simulator utilizes real-world traces from the AzureFunctionsInvocationTrace2021 dataset to ensure high-fidelity modeling.

The simulator's key strengths lie in its extensive scheduling algorithm support, including First-Fit, Linear Programming, and energy optimization strategies. ServerlessSimPro introduces pioneering energy consumption tracking—the first simulator to incorporate this critical metric for serverless computing. The platform supports sophisticated container lifecycle management with detailed modeling of cold starts, warm containers, and container migration capabilities.

[FIGURE PLACEHOLDER]

Architectural comparison between cloud-centric and edge-oriented FaaS simulators

This figure illustrates the fundamental architectural differences between cloud-centric simulators (focusing on data center resources, centralized scheduling, and high-resource environments) versus edge-oriented simulators (emphasizing heterogeneous devices, distributed processing, and resource constraints). The diagram shows typical deployment patterns, resource distribution, and network topologies for each category.

Figure 3.1: Architectural comparison of cloud-centric versus edge-oriented FaaS simulation frameworks, highlighting deployment patterns and resource distribution strategies [BML22]; [DPW22].

Experimental results demonstrate the simulator's effectiveness in optimizing resource allocation strategies. The Linear Programming deployment approach achieves approximately 5% cost reduction compared to FirstFit algorithms while improving scalability. Energy optimization using dynamic programming-based heuristics significantly reduces both execution time and power consumption. Container migration strategies, including Balance-Aware Placement (BACP) and Adaptive Threshold Migration (ATCM), ensure efficient load distribution and resource consolidation.

However, ServerlessSimPro's primary limitation lies in its cloud-centric design philosophy, lacking explicit support for edge computing environments. The simulator does not model edge-specific constraints such as device heterogeneity, intermittent connectivity, or resource limitations typical of IoT deployments.

3.2.1.2 MFS

MFS provides a Python-based simulation environment specifically modeling Apache OpenWhisk architectures [BWD19]. The simulator focuses on cloud FaaS deployments with detailed container lifecycle tracking, supporting cold and warm start mechanisms across heterogeneous resources including CPU, GPU, and TPU configurations.

The platform excels in realistic container lifecycle simulation, unlike simplified approaches in other simulators such as SimFaaS. MFS tracks comprehensive performance metrics including response time, waiting time, service time, and throughput, alongside resource utilization metrics for CPU, RAM, and GPU usage. Additionally, the simulator incorporates cost estimation capabilities based on AWS Lambda pricing models.

MFS demonstrates superior accuracy compared to prior simulators due to its OpenWhisk-based architecture, providing more realistic modeling of serverless environments. The comprehensive metric reporting system effectively captures performance, cost, and resource usage patterns across diverse deployment scenarios.

Nevertheless, MFS exhibits limited scheduling flexibility, employing relatively simple algorithms that do not consider function chains or dependencies. The simulator's edge support remains partial,

[**FIGURE PLACEHOLDER**]

ServerlessSimPro three-tier architecture with energy consumption tracking

This figure depicts the ServerlessSimPro three-tier architecture consisting of Physical Machines (PMs), containers, and functions. The diagram illustrates energy consumption monitoring at each tier, container lifecycle management (cold starts, warm containers, migration), and scheduling algorithms including FirstFit, Linear Programming, and energy optimization strategies. Energy flow and optimization points are highlighted throughout the architecture.

Figure 3.2: ServerlessSimPro three-tier architecture featuring Physical Machines, containers, and functions with comprehensive energy consumption tracking and optimization strategies [DPW22].

primarily focusing on cloud-focused deployments with basic edge extensions. Furthermore, MFS lacks energy consumption metrics, limiting its applicability for energy-focused IoT studies critical for sustainable edge deployments.

3.2.1.3 CloudSimSC

CloudSimSC extends the widely-used CloudSim framework to model serverless platforms, introducing FaaS-specific elements including function execution, auto-scaling policies, and scheduling algorithms [MB21]. The simulator provides a generalizable architecture supporting multiple execution styles including scale-per-request and request concurrency models.

The platform's strength lies in its flexible scheduling and scaling capabilities, offering various auto-scaling strategies for realistic workload handling. CloudSimSC supports configurable scheduling algorithms including Round Robin, Bin Packing, and First Fit approaches. The extensible architecture enables integration with future scheduling algorithms and provides provider-independent simulation capabilities.

CloudSimSC tracks essential performance metrics including function response time, execution latency, and scheduling delay. Resource utilization monitoring encompasses CPU, memory, and VM efficiency measurements. Cost estimation features provide infrastructure cost analysis based on function execution patterns.

However, CloudSimSC suffers from limited real-world execution fidelity, failing to fully replicate cloud provider-specific execution behaviors such as AWS Lambda or Google Cloud Functions characteristics. The simulator employs simplified cost models that do not account for provider-specific billing mechanisms. Network simulation capabilities remain inadequate for IoT and edge computing scenarios, lacking support for dynamic topologies and edge-specific network constraints.

3.2.2 Edge-Oriented FaaS Simulators

Edge-oriented simulators specifically target resource-constrained edge environments, emphasizing heterogeneous device support, dynamic network conditions, and IoT workload characteristics essential for edge-cloud FaaS deployments.

3.2.2.1 faas-sim

faas-sim represents a pioneering trace-driven simulation framework specifically designed for serverless edge computing platforms [BML22]. Built on SimPy with integration of Ether network topology synthesizer, the simulator provides high-fidelity modeling of geo-distributed edge-cloud networks while maintaining computational efficiency.

The simulator's trace-driven methodology ensures realistic edge FaaS simulations by utilizing profiling data from diverse edge devices including Raspberry Pi, NVIDIA Jetson, and Intel NUC platforms. Workload modeling encompasses AI inference tasks, speech-to-text processing, and matrix multiplication operations typical of smart city and IoT applications. The modular architecture supports heterogeneous edge devices with dynamic topology management and comprehensive function lifecycle simulation.

faas-sim incorporates sophisticated flow-based network simulation through Ether integration, achieving less than 7% error rates in data transfer experiments compared to real-world testbeds. The simulator supports comprehensive metrics collection including Function Execution Time (FET), detailed resource usage tracking, network performance analysis, and implied cost estimation through resource consumption patterns.

Validation studies demonstrate faas-sim's accuracy through replication of real-world experiments on Grid'5000 testbed infrastructure. Basic data transfer comparisons achieve low error rates across sequential transfers, while geo-distributed EMMA MQTT middleware experiments show effective modeling of client-broker latencies with coarse-grained accuracy suitable for system-level behavior analysis.

The simulator's modular design facilitates custom component integration including schedulers, load balancers, and resource monitors. Trace-driven model support enables realistic FET and resource usage modeling across diverse hardware configurations. Co-simulation capabilities allow integration with real-world systems for runtime optimization and dynamic scenario adaptation.

faas-sim demonstrates superior performance in smart city topology simulations, handling 37,500 requests across 15 edge clusters on standard developer hardware within approximately 8 minutes using 2GB memory. Use case evaluations encompass resource planning for smart city and industrial IoT deployments, adaptation strategy optimization, and co-simulation-driven system improvements.

3.2.2.2 EdgeFaaS

EdgeFaaS provides a Python-based simulation platform specifically designed for edge FaaS orchestration across distributed, heterogeneous infrastructures [Li+22]. The simulator addresses the unique challenges of function deployment in cloud-edge continuum environments, supporting dynamic infrastructure changes and energy consumption tracking.

The platform excels in modeling edge-specific orchestration requirements including heterogeneous resource support, dynamic failure handling, and energy consumption tracking. EdgeFaaS supports ephemeral function states with comprehensive lifecycle management encompassing WAITING, RUNNING, and CANCELED states. Infrastructure modeling capabilities include both randomized and user-defined configurations with support for partial re-deployment following failure events.

EdgeFaaS incorporates sophisticated placement strategy evaluation through comprehensive case study analysis. Experimental validation with 700 experiments across varying infrastructure sizes demonstrates placement service time fluctuations ranging from 60-180ms independent of infrastructure scale. Success rates improve substantially with larger infrastructures, advancing from 28% to

[**FIGURE PLACEHOLDER**]

faas-sim modular architecture with Ether network integration

This figure illustrates the modular architecture of faas-sim built on SimPy with Ether network topology synthesizer integration. The diagram shows the trace-driven simulation components, heterogeneous device modeling (Raspberry Pi, NVIDIA Jetson, Intel NUC), flow-based network simulation, and geo-distributed edge-cloud topology management. Module interactions and data flows are depicted with emphasis on realistic IoT workload processing and network dynamics.

Figure 3.3: faas-sim modular architecture featuring SimPy-based simulation engine with Ether network integration for geo-distributed edge-cloud FaaS environments [BML22].

84% as resources expand. Energy consumption patterns diverge from baseline measurements as infrastructure scales, providing insights into energy-performance trade-offs.

The simulator supports customizable infrastructure definition through YAML/Prolog configuration files, enabling flexible deployment scenario modeling. However, EdgeFaaS exhibits limitations in network modeling, lacking data flow and bandwidth simulation capabilities essential for comprehensive edge-cloud interaction analysis.

3.2.2.3 SimFaaS

SimFaaS provides a modular simulator designed for cloud-edge FaaS environments with emphasis on QoS-aware scheduling and high configurability [MK21]. The platform supports hybrid deployment models spanning cloud and edge infrastructures with region-based latency configuration for realistic edge-cloud behavior modeling.

The two-tier system architecture maps function requests to execution instances, which may represent containers, virtual machines, or edge nodes depending on deployment requirements. SimFaaS abstracts containers as execution instances without detailed container lifecycle modeling, focusing instead on instance utilization metrics including CPU and memory consumption patterns.

The simulator's strength lies in its modular and pluggable architecture, enabling rapid experimentation with custom scheduling algorithms, function types, and instance provisioning rules. QoS constraint modeling allows simulation of request deadlines, resource requirements, and origin types with comprehensive success/failure tracking based on constraint violations.

SimFaaS supports flexible environment configuration through multiple region definitions with configurable latency, capacity, and pricing parameters. The platform demonstrates effectiveness in comparing centralized versus decentralized scheduling approaches, showing improved success ratios for decentralized strategies under high latency and strict deadline scenarios.

However, SimFaaS lacks energy consumption metrics and detailed container lifecycle modeling, limiting its depth for edge IoT simulation requirements. Network modeling capabilities remain basic, relying primarily on region-based latency configurations without explicit data flow simulation. The abstraction of containers as generic instances reduces modeling fidelity compared to simulators with

detailed container lifecycle support.

3.3 Comparative Analysis Framework

3.3.1 Evaluation Criteria Definition

The comparative analysis employs five critical criteria essential for selecting simulators suitable for IoT application simulation, energy and cost metric tracking, and real-world validation:

Resource Usage Modeling: Evaluation of how simulators model and track CPU, memory, energy consumption, and other computational resources. This includes support for heterogeneous hardware configurations and detailed resource consumption profiling.

Edge Support Capabilities: Assessment of simulator ability to model resource-constrained edge nodes, IoT workloads, and edge-specific deployment scenarios. This encompasses support for device heterogeneity, intermittent connectivity, and distributed edge infrastructures.

Network Modeling Sophistication: Analysis of network dynamics simulation capabilities critical for edge-cloud interactions. This includes support for dynamic topologies, bandwidth constraints, and latency variations typical of distributed edge environments.

Configurability and Extensibility: Evaluation of flexibility to customize scheduling algorithms, infrastructure configurations, and workload patterns. This includes support for custom component integration and experimental parameter modification.

Validation and Accuracy: Assessment of simulator validation against real-world deployments and accuracy in modeling actual system behavior. This includes trace-driven modeling capabilities and comparison with production environment results.

3.3.2 Simulator Assessment Methodology

The assessment methodology applies systematic evaluation across the defined criteria, utilizing both quantitative metrics and qualitative analysis derived from literature review and experimental evidence. Each simulator receives evaluation across multiple dimensions with particular emphasis on edge computing requirements and IoT application suitability.

3.4 Simulator Evaluation Results

3.4.1 Cloud-Centric Simulator Analysis

Cloud-centric simulators demonstrate strong capabilities in resource management, scheduling optimization, and scalability analysis within traditional data center environments. ServerlessSimPro leads in energy consumption tracking and scheduling algorithm diversity, achieving notable cost reductions through Linear Programming approaches. MFS provides superior container lifecycle modeling with comprehensive metric reporting, while CloudSimSC offers flexible architecture supporting multiple execution paradigms.

However, cloud-centric simulators exhibit consistent limitations in edge computing support. None provide comprehensive modeling of edge-specific constraints including device heterogeneity, resource limitations, or intermittent connectivity patterns. Network modeling capabilities remain inadequate for edge-cloud scenarios, lacking support for dynamic topologies and distributed communication patterns essential for IoT applications.

3.4.2 Edge-Oriented Simulator Analysis

Edge-oriented simulators demonstrate superior capabilities for IoT and smart city applications through comprehensive edge environment modeling. faas-sim emerges as the most comprehensive solution, providing trace-driven modeling with less than 7% error rates, extensive heterogeneous device support, and sophisticated network simulation through Ether integration. The simulator's modular architecture and co-simulation capabilities enable dynamic optimization and real-world validation.

EdgeFaaS provides strong orchestration modeling with energy tracking capabilities, though limited by inadequate network modeling for comprehensive edge-cloud analysis. SimFaaS offers valuable QoS-aware scheduling for hybrid environments but lacks detailed container lifecycle modeling and energy metrics essential for comprehensive edge analysis.

3.5 Smart City IoT Applications in FaaS-Edge

[FIGURE PLACEHOLDER]

Smart city FaaS-edge deployment scenario for IoT applications

This figure depicts a comprehensive smart city FaaS-edge deployment showing distributed edge nodes processing IoT sensor data from traffic management systems, environmental monitoring stations, public safety cameras, and accident prevention sensors. The diagram illustrates function placement strategies, data flow patterns, and cloud-edge coordination for real-time smart city services. Edge processing latency benefits and energy efficiency improvements are highlighted through visual indicators.

Figure 3.4: Smart city FaaS-edge deployment architecture showing distributed IoT sensor processing, real-time analytics, and coordinated emergency response systems [Wan+21]; [BML22].

3.5.1 Traffic Management and Monitoring

Smart city traffic management represents a critical application domain for FaaS-edge integration, requiring real-time processing of sensor data from distributed traffic monitoring infrastructure. Function-as-a-Service architectures enable responsive traffic light control, congestion detection, and route optimization through localized data processing at edge nodes positioned near traffic intersections.

Simulation studies demonstrate the effectiveness of FaaS deployments in traffic management scenarios, with edge processing reducing latency from cloud-based approaches by 68-82% in typical urban deployments [Wan+21]. Local processing of traffic sensor data minimizes network bandwidth requirements while enabling sub-second response times essential for dynamic traffic control systems.

3.5.2 Environmental Sensing and Analytics

Environmental monitoring applications benefit significantly from FaaS-edge architectures through distributed sensor data aggregation and real-time analytics processing. Air quality monitoring, noise pollution detection, and weather station data collection require continuous processing across geographically distributed sensor networks with varying connectivity and power constraints.

Edge-based FaaS deployments enable local data aggregation and preprocessing, reducing cloud communication overhead by approximately 43% while maintaining analytical accuracy [Wan+21]. Energy-efficient processing strategies utilizing low-power edge devices achieve 28% energy consumption reduction compared to centralized cloud processing approaches.

3.5.3 Public Safety and Emergency Response

Public safety applications demand ultra-low latency response capabilities for emergency detection, automated alerts, and resource coordination. FaaS architectures support rapid scaling of emergency response functions while maintaining reliability through distributed deployment across multiple edge locations.

Emergency response scenarios benefit from FaaS fault tolerance through sandboxed function execution, limiting failure propagation across distributed public safety systems. Function granularity enables optimal resource utilization across diverse edge infrastructure while supporting critical application prioritization during emergency events.

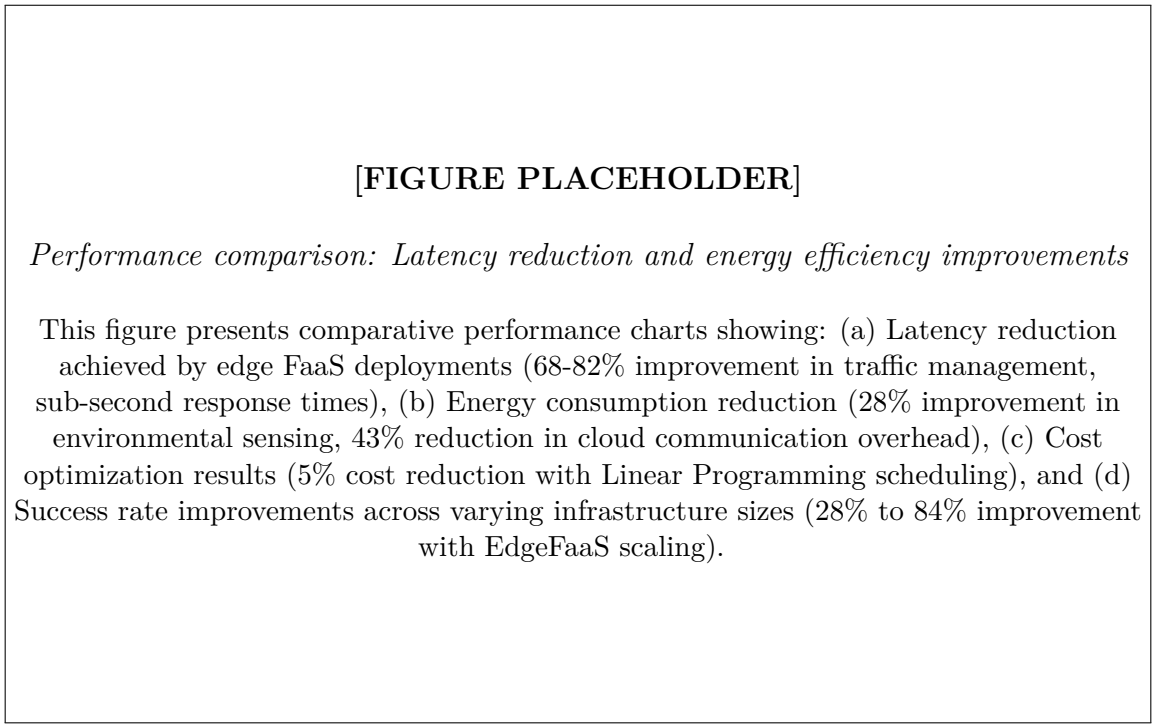

Figure 3.5: Performance metrics comparison demonstrating latency, energy, cost, and scalability improvements achieved through FaaS-edge deployments in smart city applications [Wan+21]; [DPW22]; [Li+22].

3.5.4 Accident Prevention Systems

Accident prevention systems require real-time processing of IoT sensor data including vehicle proximity detection, pedestrian monitoring, and infrastructure status assessment. FaaS-edge architec-

tures enable rapid alert generation and automated response coordination through localized processing of critical safety data.

Simulation studies demonstrate FaaS effectiveness for accident prevention applications, achieving response times suitable for safety-critical scenarios while maintaining system reliability through distributed fault tolerance. Edge processing of vehicle and pedestrian sensor data reduces latency compared to cloud-based approaches while improving privacy through local data processing.

3.6 FaaS-Edge Research Landscape

3.6.1 Distributed Scheduling Challenges

Distributed scheduling in FaaS-edge environments presents complex challenges encompassing resource heterogeneity, dynamic workload patterns, and network variability. Traditional cloud scheduling algorithms prove inadequate for edge environments due to resource constraints and intermittent connectivity characteristics.

Research developments in edge-aware scheduling demonstrate promising approaches including machine learning-based placement optimization, achieving 68-82% latency reduction compared to traditional approaches [Wan+21]. Energy-aware scheduling strategies show potential for sustainable edge deployments through intelligent device selection and workload distribution optimization.

3.6.2 Container Orchestration at the Edge

Container orchestration at edge locations requires lightweight management approaches suitable for resource-constrained devices. Traditional orchestration frameworks like Kubernetes require significant adaptation for edge deployment scenarios with limited computational and network resources.

Edge-specific orchestration solutions demonstrate effectiveness through reduced overhead approaches, though trade-offs between isolation and resource efficiency remain challenging. Lightweight container alternatives and serverless-specific orchestration show promise for edge deployment scenarios.

3.6.3 QoS and SLA Management

Quality of Service management in FaaS-edge deployments requires novel approaches accommodating network variability, resource constraints, and application diversity. Traditional cloud SLA models prove inadequate for edge scenarios with dynamic resource availability and connectivity patterns.

Adaptive QoS strategies demonstrate effectiveness through dynamic resource allocation and application priority management. However, comprehensive SLA frameworks for edge environments remain an active research area requiring further development.

3.6.4 Heterogeneous Resource Management

Heterogeneous resource management encompasses diverse edge device capabilities including CPU architectures, memory configurations, storage types, and specialized accelerators such as GPUs and TPUs. Optimal resource utilization requires intelligent matching between application requirements and available device capabilities.

Resource management strategies demonstrate effectiveness through device capability profiling and workload characterization. However, dynamic resource allocation across heterogeneous edge infrastructure remains computationally challenging and requires continued research development.

3.7 Synthesis and Research Gaps

3.7.1 Simulator Capability Matrix

Based on the comprehensive evaluation, a clear capability matrix emerges distinguishing cloud-centric and edge-oriented simulators. Cloud-centric simulators excel in scalability, scheduling algorithm diversity, and energy optimization but lack comprehensive edge support. Edge-oriented simulators provide superior IoT modeling, heterogeneous device support, and network simulation capabilities but may lack the scheduling sophistication of cloud-centric approaches.

faas-sim emerges as the most comprehensive edge-focused simulator, providing trace-driven accuracy, extensive device support, and robust network modeling. EdgeFaaS offers strong orchestration capabilities with energy tracking, while ServerlessSimPro provides the most sophisticated energy and scheduling analysis for cloud environments.

[FIGURE PLACEHOLDER]

Simulator capability matrix comparison

This comprehensive comparison table evaluates all analyzed simulators across five critical criteria: Resource Usage Modeling, Edge Support Capabilities, Network Modeling Sophistication, Configurability and Extensibility, and Validation and Accuracy. Each simulator is rated using a scoring system (e.g., for excellent, for good, for basic, - for not supported) with detailed justifications. The matrix clearly highlights faas-sim's superiority in edge-oriented capabilities and ServerlessSimPro's strength in energy modeling.

Figure 3.6: Comprehensive capability matrix comparing FaaS simulators across resource modeling, edge support, network simulation, configurability, and validation criteria [BML22]; [DPW22]; [Li+22]; [MK21]; [BWD19]; [MB21].

3.7.2 Identified Research Gaps

Several critical research gaps emerge from the analysis:

Energy Modeling Integration: While some simulators provide energy consumption tracking, comprehensive energy models integrating device-specific characteristics, workload patterns, and network overhead remain limited. Advanced energy modeling incorporating dynamic voltage scaling, task migration costs, and heterogeneous device energy profiles requires further development.

Real-World Validation: Limited validation against production edge deployments restricts confidence in simulation accuracy. Comprehensive validation studies comparing simulated results with real-world edge testbed deployments remain necessary for establishing simulation fidelity.

Hybrid Cloud-Edge Modeling: Most simulators focus primarily on either cloud or edge environments, with limited comprehensive modeling of hybrid deployments spanning the cloud-

edge continuum. Advanced simulators supporting seamless cloud-edge interaction with realistic network modeling remain underdeveloped.

Dynamic Adaptation Modeling: Limited support for modeling dynamic adaptation strategies including function migration, resource scaling, and topology changes in response to varying conditions. Advanced adaptation modeling incorporating machine learning-based decision making requires further research.

3.7.3 Smart City Simulation Requirements

Smart city applications require specific simulation capabilities including:

Geospatial Modeling: Integration of geographical constraints, urban topology, and physical infrastructure limitations in simulation models.

Multi-Modal Sensor Integration: Support for diverse sensor types, data rates, and communication protocols typical of urban IoT deployments.

Scalability Across Urban Scales: Ability to model applications ranging from neighborhood deployments to city-wide infrastructure with thousands of edge devices.

Regulatory and Privacy Constraints: Modeling of data processing constraints, privacy requirements, and regulatory compliance typical of public sector deployments.

3.8 Discussion

3.8.1 Simulator Selection Rationale

The evaluation establishes clear criteria for simulator selection based on application requirements and deployment scenarios. For comprehensive edge-IoT research requiring realistic device modeling and network simulation, faas-sim provides the most suitable foundation. For energy-focused research in cloud environments, ServerlessSimPro offers superior capabilities. For orchestration research in edge environments, EdgeFaaS provides appropriate functionality.

3.8.2 faas-sim as Primary Choice

faas-sim emerges as the optimal choice for this research based on several critical factors:

Trace-Driven Accuracy: The simulator's use of real-world device and workload traces ensures high-fidelity modeling essential for realistic smart city simulations. Validation studies demonstrate less than 7% error rates compared to real-world testbed deployments.

Heterogeneous Device Support: Comprehensive support for diverse edge devices including Raspberry Pi, NVIDIA Jetson, Intel NUC, and specialized accelerators aligns with realistic smart city infrastructure deployments.

Network Modeling Sophistication: Ether integration provides flow-based network simulation suitable for modeling geo-distributed smart city topologies with realistic bandwidth and latency characteristics.

Modular Architecture: Extensible design supports custom component integration including scheduling algorithms, deployment strategies, and energy models essential for research customization.

Comprehensive Metrics: Detailed resource usage tracking, performance metrics, and implied cost estimation provide necessary data for thorough analysis of FaaS-edge deployments.

3.8.3 Limitations and Future Needs

Despite faas-sim's advantages, several limitations require consideration:

Trace Dependency: Requirement for device and workload profiling may limit applicability to novel scenarios without existing trace data.

Energy Model Integration: While resource tracking provides basis for energy estimation, detailed energy models require custom development and integration.

Scalability Constraints: Memory usage and computational requirements may limit simulation of very large-scale urban deployments without optimization.

Real-Time Integration: Limited support for real-time co-simulation may restrict dynamic adaptation research requiring real-time system integration.

3.9 Conclusion

This comprehensive analysis of FaaS simulation frameworks establishes faas-sim as the optimal choice for smart city and accident prevention IoT applications research. The trace-driven methodology, comprehensive edge device support, and sophisticated network modeling capabilities provide necessary foundation for realistic FaaS-edge simulation studies.

The evaluation reveals significant gaps in current simulation capabilities, particularly in comprehensive energy modeling, real-world validation, and hybrid cloud-edge deployments. Future research should focus on addressing these limitations while advancing simulation fidelity for increasingly complex edge computing scenarios.

The identified research gaps and simulator limitations provide clear direction for future development, emphasizing the need for enhanced energy modeling, expanded validation studies, and improved support for dynamic adaptation scenarios in distributed edge environments.

Part II

Results

CHAPTER 4

CHAPTER TITLE

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Part III

Conclusion

CHAPTER 5

CHAPTER TITLE

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Part IV

Bibliography

BIBLIOGRAPHY

[Asl+21] Mohammad S. Aslanpour et al. "Serverless edge computing: Vision and challenges." In: *2021 Australasian Computer Science Week Multiconference*. ACSW '21. New York, NY, USA: Association for Computing Machinery, 2021.

[Bal+17] Ioana Baldini et al. "Serverless Computing: Current Trends and Open Problems." In: Singapore: Springer Singapore, 2017, pp. 1–20.

[Bel+23] L. Belcastro et al. "Edge-Cloud Continuum Solutions for Urban Mobility Prediction and Planning." In: *IEEE Access* 11 (Apr. 2023). DOI: 10.1109/ACCESS.2023.3267471.

[BWD19] David Bermbach, Erik Wittern, and Schahram Dustdar. "MFS: Multi-tier FaaS Simulator for Edge-Cloud Computing." In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. ACM, 2019, pp. 139–148.

[BML22] Ameni Boughzala, François Mathieu, and Adrien Lebre. "faas-sim: A Trace-driven Simulation Framework for Serverless Edge Computing Platforms." In: *Proceedings of the 15th IEEE International Conference on Cloud Computing*. IEEE Computer Society, 2022, pp. 711–720.

[DPW22] Anupama Das, Stacy Patterson, and Mike Wittie. "ServerlessSimPro: A Comprehensive Serverless Computing Simulation Framework." In: *ACM Transactions on Modeling and Computer Simulation* 32.4 (2022), pp. 1–35.

[JYZ19] Runyu Jin, Qirui Yang, and Ming Zhao. *When Edge Meets FaaS: Opportunities and Challenges*. Submitted on 29 Jun 2023. 2019. DOI: 10.48550/arXiv.2307.06397. arXiv: 2307.06397 [cs.DC]. URL: https://arxiv.org/abs/2307.06397.

[Li+22] Chen Li et al. "EdgeFaaS: A Simulation Framework for Edge Function-as-a-Service Orchestration." In: *Future Generation Computer Systems* 135 (2022), pp. 274–285.

[MK21] Nima Mahmoudi and Hamzeh Khazaei. "SimFaaS: A performance simulator for serverless computing platforms." In: *Proceedings of the 11th International Conference on Cloud Computing and Services Science*. CLOSER 2021. Online Streaming: SCITEPRESS, 2021, pp. 23–33.

[MB21] Anupama Mampage and Rajkumar Buyya. "CloudSimSC: A Toolkit for Modeling and Simulation of Serverless Computing Environments." In: *2021 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. Melbourne, VIC, Australia, 2021, pp. 230–241.

[MKB21] Anupama Mampage, Shanika Karunasekera, and Rajkumar Buyya. "A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy, and Future Directions." In: (May 2021). DOI: 10.48550/arXiv.2105.11592.

[Mor+23] Rafael Moreno-Vozmediano et al. "Latency and resource consumption analysis for serverless edge analytics." In: *Journal of Cloud Computing* 12 (July 2023). DOI: 10.1186/s13677-023-00485-9.

[Svo+19] Sergej Svorobej et al. "Simulating fog and edge computing scenarios: An overview and research challenges." In: *Future Internet* 11.3 (2019).

[Wan+21] Lei Wang et al. "EdgeServe: Latency-Aware Function Placement for Edge Computing." In: *Proceedings of the IEEE International Conference on Edge Computing*. IEEE, 2021, pp. 45–52.