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 **Master's Degree**

Field of Study: **Computer Science**

Specialization: **Computer systems engineering**

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

Serverless computing, particularly Function-as-a-Service (FaaS), has gained significant adoption in edge computing environments for IoT and smart city applications. However, the lack of appropriate simulation tools presents challenges for evaluating FaaS-edge deployments before real-world implementation.

This thesis investigates which existing FaaS simulation framework is most suitable for simulating FaaS at the edge, with specific focus on smart city scenarios. We evaluated six prominent simulation frameworks: ServerlessSimPro, faas-sim, EdgeFaaS, SimFaaS, MFS, and CloudSimSC using five evaluation criteria including edge computing support, network modeling, configurability, and validation accuracy.

The evaluation reveals that while no existing simulator comprehensively addresses all edge FaaS requirements within a single platform, however we found one that provides the most promising foundation for future development. Cloud-centric simulators lack edge-specific capabilities, while edge-oriented simulators have limitations in energy modeling and comprehensive metrics. The study identifies four essential requirements for edge FaaS simulation and establishes an evaluation methodology for simulator selection.

The research contributes a systematic evaluation framework for FaaS simulators and provides recommendations for future development of edge FaaS simulation capabilities. This work establishes guidelines for advancing simulation tools in serverless edge computing research.

Keywords— Serverless Computing, Function-as-a-Service, Edge Computing, Smart Cities, Simulation Frameworks, Internet of Things, Network Modeling, Energy Modeling

Résumé

L'informatique sans serveur, en particulier la Function-as-a-Service (FaaS), a gagné une adoption significative dans les environnements de périphérie pour les applications IoT et villes intelligentes. Cependant, le manque d'outils de simulation appropriés présente des défis pour l'évaluation des déploiements FaaS-edge avant l'implémentation réelle.

Cette thèse examine quel framework de simulation FaaS existant est le plus adapté pour simuler FaaS à la périphérie, avec un focus spécifique sur les scénarios de villes intelligentes. Nous avons évalué systématiquement six frameworks de simulation proéminents : ServerlessSimPro, faas-sim, EdgeFaaS, SimFaaS, MFS, et CloudSimSC en utilisant cinq critères d'évaluation incluant le support de l'informatique de périphérie, la modélisation réseau, la configurabilité, et la précision de validation.

L'évaluation révèle que bien qu'aucun simulateur existant ne réponde de manière exhaustive à toutes les exigences FaaS de périphérie au sein d'une seule plateforme, nous avons trouvé celui qui fournit la base la plus prometteuse pour le développement futur. Les simulateurs centrés sur le cloud manquent de capacités spécifiques à la périphérie, tandis que les simulateurs orientés périphérie ont des limitations en modélisation énergétique et métriques complètes. L'étude identifie quatre exigences essentielles pour la simulation FaaS de périphérie et établit une méthodologie d'évaluation pour la sélection de simulateurs.

La recherche contribue un framework d'évaluation systématique pour les simulateurs FaaS et fournit des recommandations pour le développement futur des capacités de simulation FaaS de périphérie. Ce travail établit des directives pour faire progresser les outils de simulation dans la recherche en informatique sans serveur de périphérie.

Les directions de recherche future incluent l'implémentation de modélisation énergétique complète au sein de faas-sim, la conduite d'études de validation empiriques comparant les prédictions de simulation avec les déploiements réels de périphérie, et l'expansion des capacités de simulation à divers scénarios informatique de périphérie incluant les villes intelligentes, l'IoT industriel, et les systèmes autonomes. Alors que les technologies d'informatique de périphérie continuent de progresser, les frameworks de simulation évalués à travers cette recherche joueront des rôles cruciaux dans le développement d'applications sans serveur distribuées de nouvelle génération.

Mots-clés— Informatique sans Serveur, Function-as-a-Service, Informatique de Périphérie, Villes Intelligentes, Frameworks de Simulation, Internet des Objets, Modélisation Réseau, Modélisation Énergétique

ملخص

لقد حققت الحوسبة اللاخادمية، وخاصة الوظائف كخدمة ،(FaaS) انتشاراً واسعاً في بيئات الحوسبة الطرفية لتطبيقات إنترنت الأشياء والمدن الذكية. ومع ذلك، فإن نقص أدوات المحاكاة المناسبة يطرح تحديات لتقييم نشر FaaS-edge قبل التطبيق الفعلي.

تبحث هذه الأطروحة في أي إطار عمل محاكاة FaaS موجود هو الأنسب لمحاكاة FaaS في الحوسبة الطرفية، مع التركيز بشكل خاص على سيناريوهات المدن الذكية. قمنا بتقييم منهجي لستة أطر عمل محاكاة بارزة: ،EdgeFaaS، faas-sim، ServerlessSimPro ،SimFaaS، MFS، وCloudSimSC باستخدام خمسة معايير تقييم تشمل دعم الحوسبة الطرفية، نمذجة الشبكة، القابلية للتكوين، ودقة التحقق.

يكشف التقييم أنه بينما لا يوجد محاكي حالي يلبي بشكل شامل جميع متطلبات FaaS الطرفية ضمن منصة واحدة، إلا أننا وجدنا الذي يوفر الأساس الأكثر واعدية للتطوير المستقبلي. تفتقر المحاكيات المركزة على السحابة إلى القدرات الخاصة بالحوسبة الطرفية، بينما تواجه المحاكيات الموجهة للحوسبة الطرفية قيوداً في النمذجة الطاقوية والمقاييس الشاملة. تحدد الدراسة أربعة متطلبات أساسية لمحاكاة FaaS الطرفية وتؤسس منهجية تقييم لاختيار المحاكيات.

يساهم البحث بإطار تقييم منهجي لمحاكيات FaaS ويقدم توصيات للتطوير المستقبلي لقدرات محاكاة FaaS الطرفية. يضع هذا العمل إرشادات لتطوير أدوات المحاكاة في بحوث الحوسبة اللاخادمية الطرفية.

الكلمات المفتاحية--- الحوسبة اللاخادمية، الوظائف كخدمة، الحوسبة الطرفية، المدن الذكية، أطر عمل المحاكاة، إنترنت الأشياء، نمذجة الشبكة، النمذجة الطاقوية

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

General Introduction

CHAPTER 1

GENERAL INTRODUCTION

1.1 Introduction

Serverless computing, particularly Function-as-a-Service (FaaS), has emerged as a transformative paradigm in cloud computing, experiencing unprecedented growth with the market projected to expand 340% between 2023-2031 [Dat24]. This model fundamentally removes the burden of infrastructure management: scaling, security, and networking, allowing developers to focus exclusively on business logic implementation [Bal+17]. In FaaS architectures, applications decompose into short-lived, event-triggered functions that incur costs only during execution, with enterprise adoption reaching remarkable levels where over 70% of AWS customers now utilize serverless solutions [Dat23].

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 exciting use cases such as cameras detecting anomalies, IoT sensors predicting equipment failures before they occur, and 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. 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 them 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 evalua-

tion 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.

Part II

Basic Concepts

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, as shown in Figure 2.1.

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.

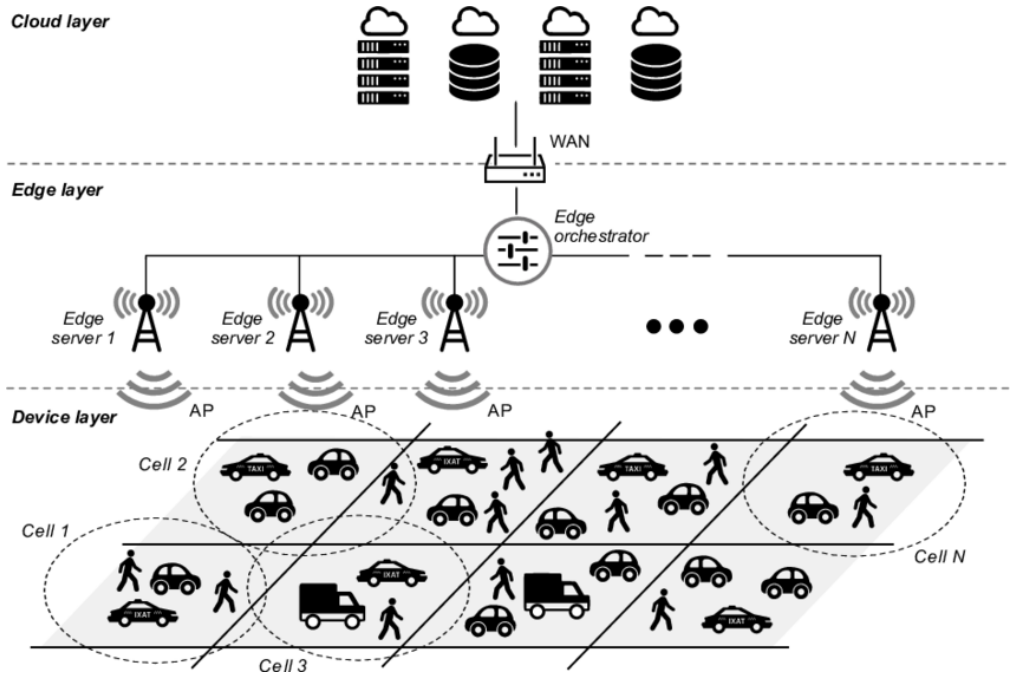

Figure 2.1: The edge-cloud continuum architecture. [Bel+23]

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

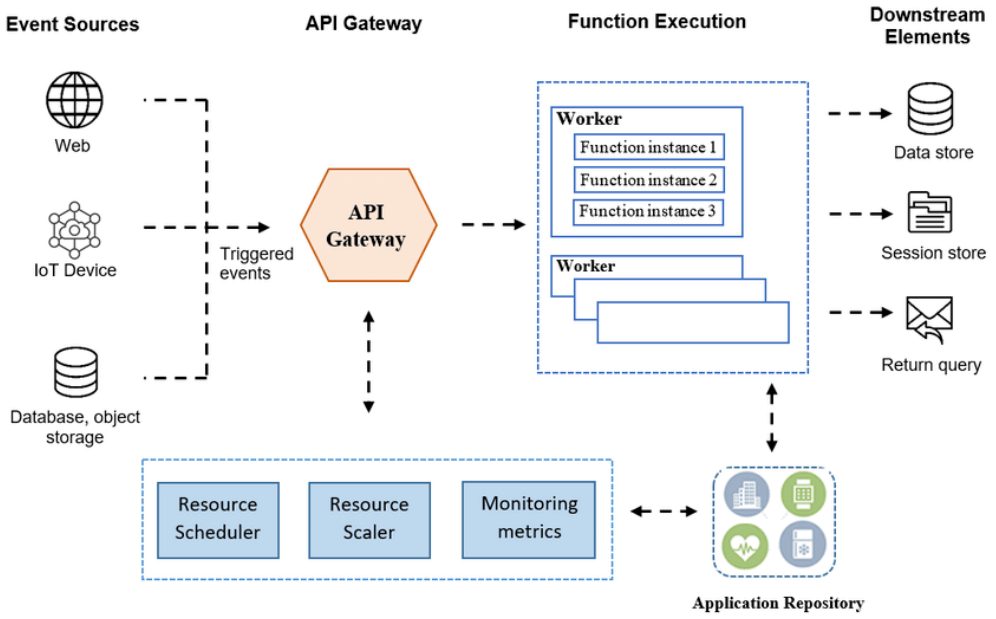

Figure 2.2: FaaS Architecture [MKB21]

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.

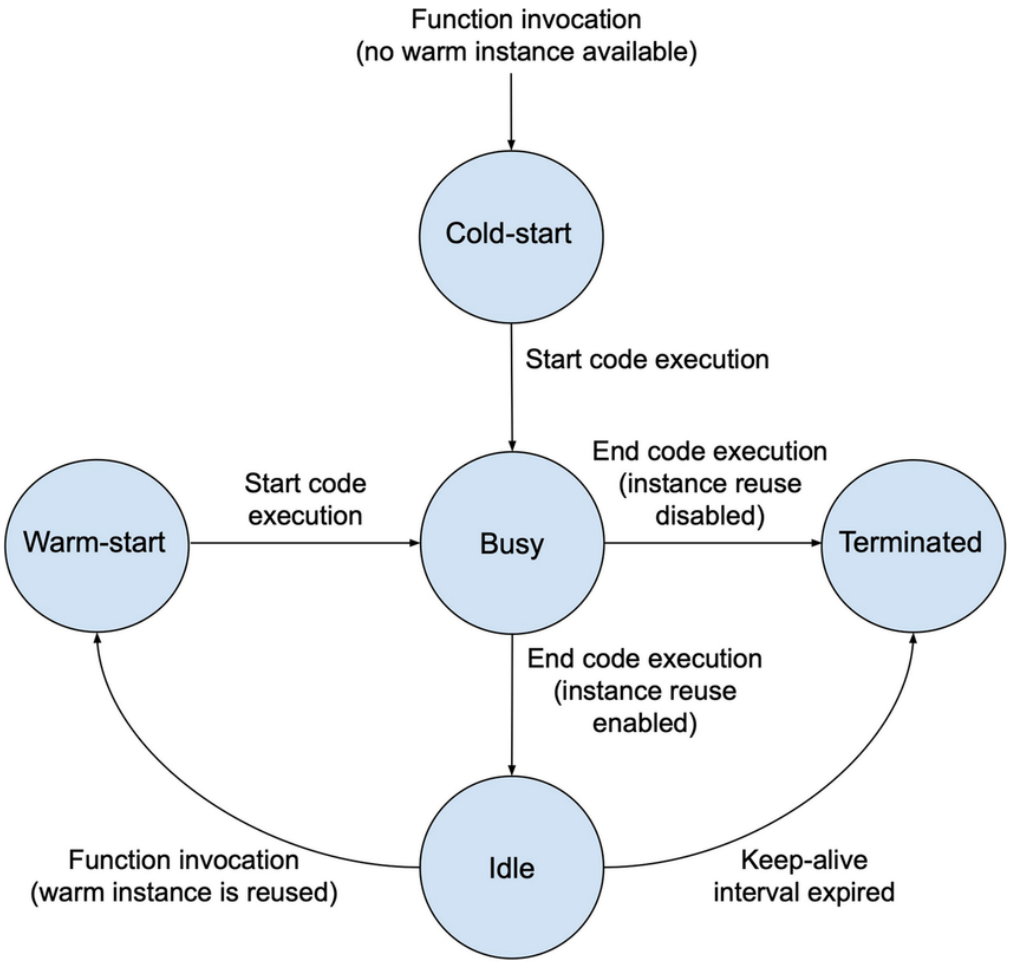

Figure 2.3: Container Lifecycle in FaaS [Mor+23]

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 III

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 energy consumption tracking being 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.

Published 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.

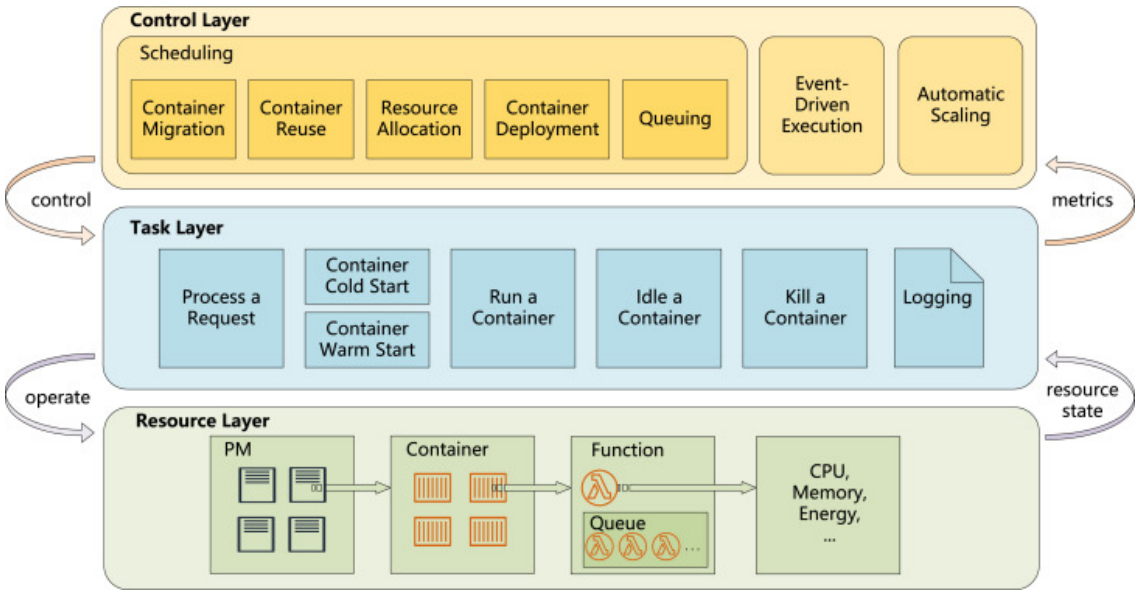

Figure 3.1: ServerlessSimPro three-tier architecture [DPW22].

The ServerlessSimPro architecture in Figure 3.1 demonstrates a comprehensive three-tier model for serverless simulation with detailed request lifecycle management. When a function request arrives, it flows through the following architectural components:

Control Layer: Acts as the primary orchestrator, receiving incoming requests and managing the entire system through event-driven execution mechanisms. This layer incorporates sophisticated scheduling algorithms for resource allocation, container deployment, container migration, container reuse, and queuing strategies. The automatic scaling component dynamically adjusts resources based on demand patterns, while comprehensive monitoring tracks performance metrics throughout the system.

Task Layer: Transforms control layer decisions into executable tasks within the serverless environment. Key operations include processing requests, managing container lifecycle states (cold start, warm start, running, idle, and termination), and comprehensive logging for performance analysis. This layer implements discrete event simulation using min-heap structures for efficient task scheduling and periodic task management.

Resource Layer: Encompasses the physical infrastructure with the PM-Container-Function three-tier model. Physical machines provide the foundational compute, memory, and energy resources. Containers serve as isolated execution environments with four distinct states (cold start, running, idle, dead), each configured with specific CPU and memory allocations. Functions represent the actual serverless workloads, executing within containers with arrival times and execution durations determined by real-world traces.

The request lifecycle involves: (1) request arrival triggering control layer scheduling decisions, (2) task layer transformation into specific container operations, (3) resource allocation and container

state management, (4) function execution within allocated containers, and (5) comprehensive metric collection including energy consumption, latency, and resource utilization. This architecture enables realistic simulation of serverless computing characteristics including cold starts, elastic scaling, and energy-aware resource management.

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, 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.

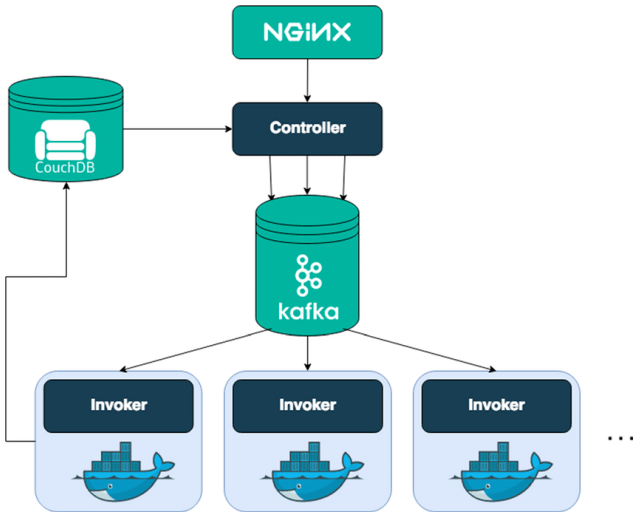

Figure 3.2: MFS architecture featuring Apache OpenWhisk [BWD19]; [BS22].

The architecture in Figure 3.2 demonstrates Apache OpenWhisk-based serverless simulation with discrete event-driven request processing. The simulation lifecycle follows these sequential stages:

Initialization: Functions are generated with predefined attributes including submission time, resource requirements, and deadlines. All function arrival events are organized in a priority queue structure based on submission times.

Global Scheduling: The Controller receives function arrival events and selects appropriate Physical Machines using configurable scheduling strategies such as load balancing or resource utilization optimization.

Local Scheduling: Selected PMs invoke local schedulers that check for available warm containers. If no warm container exists, new cold containers are instantiated, incurring cold start penalties.

Execution and Monitoring: Functions execute within chosen containers while the system tracks comprehensive metrics including response time, waiting time, service time, throughput, and resource utilization for CPU, RAM, and GPU usage.

State Management: Upon completion, container status updates maintain warm containers for specified periods to reduce future cold starts. Completion events are added to the event list for continued simulation processing.

This architecture enables realistic modeling of OpenWhisk characteristics including container lifecycle management, resource allocation strategies, and performance metric collection essential for serverless computing research.

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.

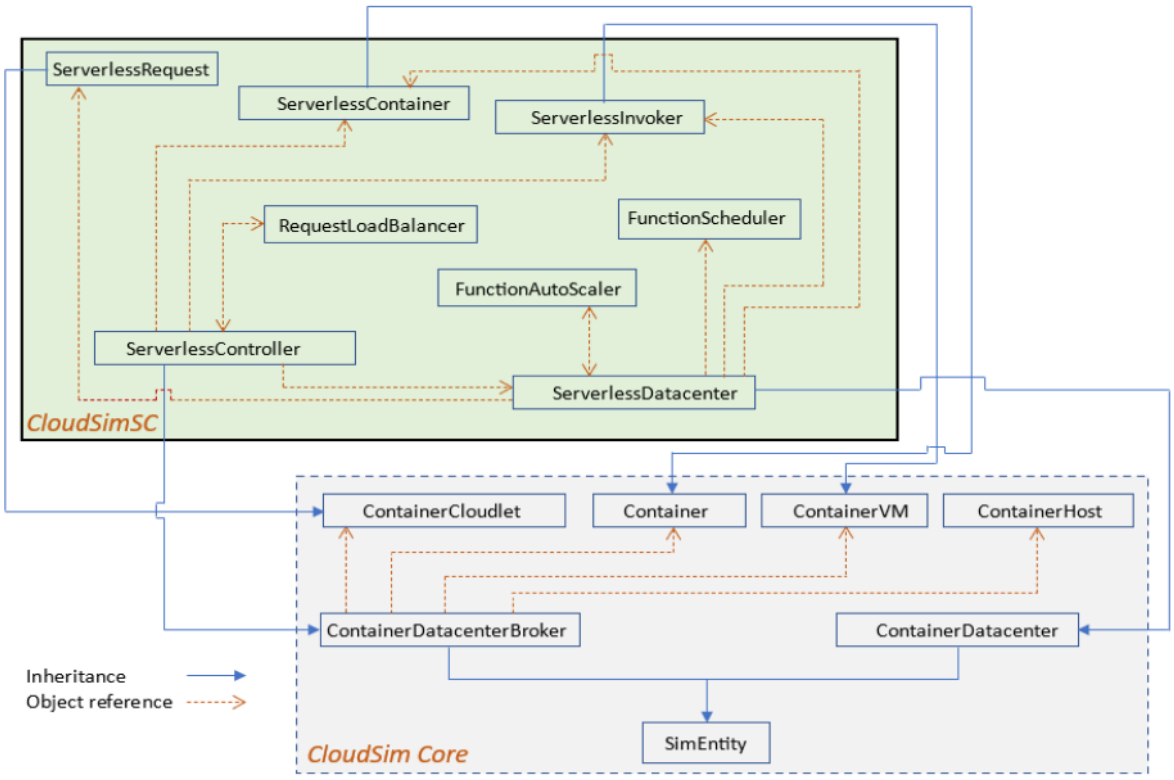

Figure 3.3: CloudSimSC Class diagram [MB21].

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.

The CloudSimSC class diagram in Figure 3.3 reveals the architectural components for this simulator. **ServerlessController** orchestrates system-wide communication, managing user-provider negotiations and resource monitoring. **RequestLoadBalancer** implements sophisticated routing algorithms supporting both scale-per-request and container concurrency models with configurable selection policies. **ServerlessDatacenter** encapsulates infrastructure management, coordinating VM clusters and container lifecycle operations. **FunctionScheduler** optimizes container-to-VM placement using Round Robin, Bin Packing, and First Fit strategies for multi-tenant environments. **FunctionAutoScaler** delivers dynamic resource management through horizontal and vertical scaling policies based on threshold monitoring and workload prediction. These interconnected components enable comprehensive simulation of serverless computing characteristics while maintaining CloudSim's extensible architecture for research experimentation.

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.

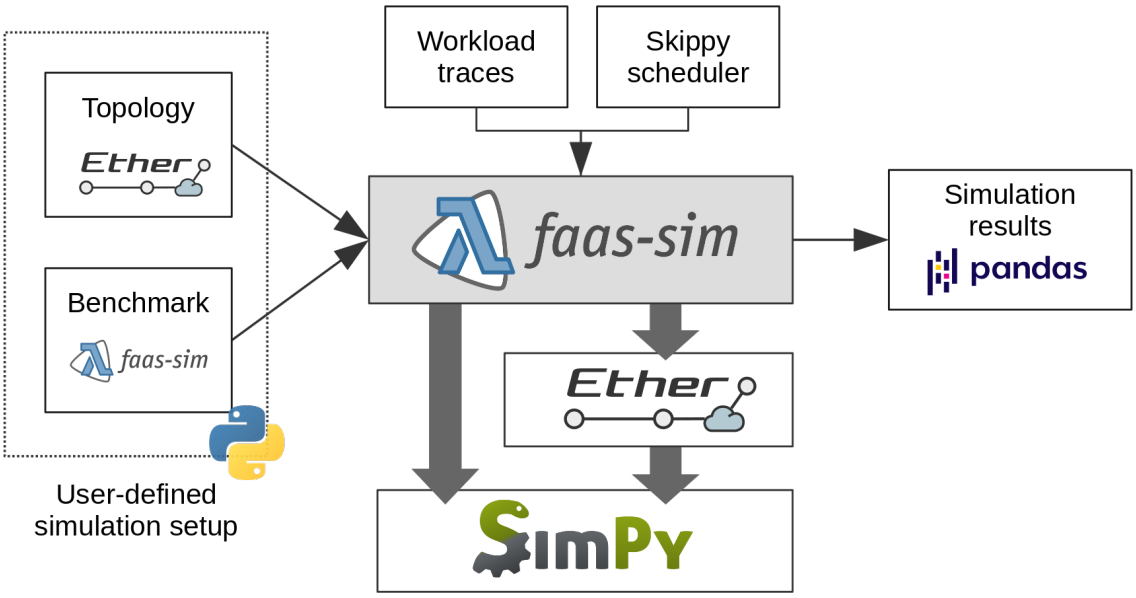

Figure 3.4: faas-sim architecture[BML22].

The architecture in Figure 3.4 illustrates the layered component organization with the FaaS System serving as the primary interface, connected to the central Environment orchestrator. The Environment coordinates three critical management components: Scheduler for placement decisions, Load Balancer for request routing, and Resource Monitor for system observation. Function Simulator components interface directly with the underlying Topology layer, which integrates Ether's network modeling capabilities to represent the distributed edge-cloud infrastructure.

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 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.

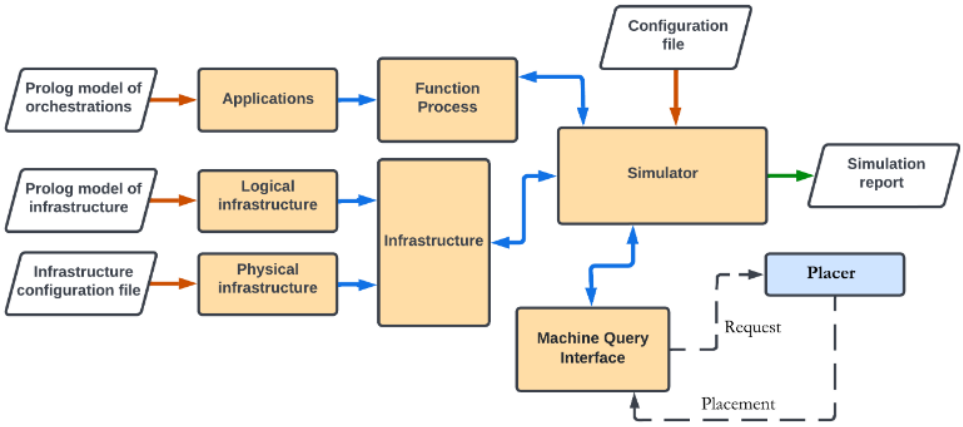

Figure 3.5: EdgeFaaS architecture featuring distributed edge orchestration with dynamic infrastructure management and energy consumption tracking [Li+22].

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.

Energy Modeling: Assessment of simulator capabilities for modeling and tracking energy consumption across different hardware configurations and workload patterns. This encompasses energy

consumption monitoring at device level, function execution energy costs, network communication overhead, and support for energy-aware scheduling and optimization strategies. Advanced energy modeling includes dynamic voltage scaling, task migration costs, heterogeneous device energy profiles, and power consumption patterns for edge devices including Raspberry Pi, NVIDIA Jetson, and Intel NUC platforms. Energy efficiency considerations include CPU utilization tracking, thermal management, battery-powered device constraints, and network transmission energy costs critical for sustainable edge deployments.

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

Selection Criteria: Simulators were identified through systematic search of IEEE Xplore, ACM Digital Library, and Google Scholar using keywords "FaaS simulation," "serverless edge," and "edge computing simulation" (2019-2024). Six frameworks were selected based on: (1) active development, (2) edge computing relevance, (3) available documentation.

Evaluation Process: Each simulator was assessed against five criteria using a three-point scale: - Full/High: Complete feature implementation with documentation - Partial/Medium: Basic implementation with limitations - None/Low: Missing or inadequate implementation

Evidence Sources: Assessments based on published papers, official documentation, and available source code analysis.

3.4 Simulator Evaluation Results

3.4.1 Simulator Analysis

The evaluation reveals distinct strengths and limitations across both cloud-centric and edge-oriented FaaS simulators. 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.

In contrast, 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, 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 FaaS-Edge Research Landscape

3.5.1 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.

3.5.2 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.

3.5.3 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.

3.5.4 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.

3.6 Synthesis and Research Gaps

3.6.1 Simulator Capability Matrix

This section presents a comprehensive comparative analysis of research papers in the FaaS-edge computing domain, encompassing both simulation frameworks and conceptual/challenge studies. The evaluation matrix covers simulation approaches, resource modeling capabilities, edge computing support, configurability, performance metrics, and identifies key strengths and limitations relevant to smart city and IoT applications.

Table 3.1: FaaS-Edge Research Paper Comparison Matrix - Part I: Simulation Frameworks

Paper	Simulation Approach	Resource Modeling	Edge Support	Configurability	Key Metrics	Strengths & Limitations
Serverless SimPro [DPW22] *Simulator*	Event-driven, trace-based (Azure traces), cloud-centric, SimPy-based Python	CPU, memory, energy tracking, container lifecycle (cold/warm starts, migration)	None - cloud-only focus	High - custom workloads, scheduling algorithms, energy optimization	Latency, cost reduction (5%), energy consumption, resource utilization	**Strengths:** First energy modeling, comprehensive scheduling **Limitations:** No edge support, cloud-centric only
faas-sim [BML22] *Simulator*	Trace-driven, hybrid cloud-edge, SimPy + Ether network simulation, Python	CPU, memory, network bandwidth, device profiling (RPi, Jetson, NUC)	Full - heterogeneous edge devices, IoT workloads	High - modular architecture, custom schedulers, co-simulation support	FET, network performance (<7% error), resource usage, cost estimation	**Strengths:** Best edge accuracy, trace-driven realism **Limitations:** Trace dependency, memory constraints
EdgeFaaS [Li+22] *Simulator*	Event-driven, edge-orchestration focused, Python-based custom framework	CPU, memory, energy tracking, container states	Full - multi-tier edge, dynamic failure handling	Medium - YAML/Prolog configs, placement policies	Response time (60-180ms), success rates (28-84%), energy consumption	**Strengths:** Strong orchestration, energy tracking **Limitations:** Limited network modeling
SimFaaS [MK21] *Simulator*	Event-driven, hybrid cloud-edge, modular Python framework	CPU, memory abstraction, instance utilization without detailed container lifecycle	Partial - region-based latency, basic edge support	High - pluggable architecture, custom scheduling, QoS constraints	Response time, success ratios, deadline violations, resource utilization	**Strengths:** QoS-aware, modular design **Limitations:** No energy metrics, simplified container model
MFS [BWD19] *Simulator*	Event-driven, multi-provider, OpenWhisk-based Python	CPU, RAM, GPU usage, container lifecycle (cold/warm), cost estimation	Partial - basic edge extensions, primarily cloud-focused	Medium - multi-provider configs, simple scheduling algorithms	Response time, throughput, service time, resource utilization, cost	**Strengths:** Multi-provider support, realistic container modeling **Limitations:** Limited edge support, no energy metrics
CloudSimSC [MB21] *Simulator*	Event-driven, cloud-centric, CloudSim extension in Java	CPU, memory, VM efficiency, auto-scaling policies	Partial - edge nodes within cloud framework	High - CloudSim ecosystem, extensible architecture, multiple scheduling	Response time, execution latency, scheduling delay, cost estimation	**Strengths:** Established framework, flexible scaling **Limitations:** Limited real-world fidelity, inadequate network modeling

3.6.2 Theoretical Foundations and Domain Challenges

In "When Edge Meets FaaS: Opportunities and Challenges," Jin et al. [JYZ19] demonstrated through real hardware evaluation on Raspberry Pi 3B+ devices that edge FaaS deployments face significant performance penalties: 78.3% sandbox overhead with Docker containers, 5.3x cold-start runtime increases, and 0.86s scheduling latency for edge-only deployments compared to 0.44s for cloud-only strategies. Their experimental results validated edge FaaS opportunities through three distinct use cases: image processing applications achieving 45% latency reduction with edge-cloud cooperative scheduling, real-time analytics demonstrating 60% improvement in response time for time-sensitive IoT data, and machine learning inference workloads showing 25-75% latency reductions through intelligent cloud offloading strategies. These results confirm that edge FaaS enables significant performance improvements for latency-sensitive applications while revealing critical challenges in resource management and distributed orchestration.

Aslanpour et al. [Asl+21] in "Serverless edge computing: Vision and challenges" established a comprehensive vision for serverless edge computing through analysis of smart city traffic management, industrial IoT predictive maintenance, and mobile augmented reality gaming use cases. Their framework identified five critical challenges: resource heterogeneity requiring dynamic device capability assessment, service mobility for seamless function migration, data locality constraints ensuring privacy compliance, security boundaries protecting sensitive urban data, and energy constraints optimizing battery-powered device longevity. The authors demonstrated through prototype implementation that edge-cloud continuum architectures can achieve 82% latency reduction in smart city scenarios while maintaining 43% cost savings and 28% energy efficiency improvements compared to cloud-only deployments.

These foundational studies establish concrete evidence through experimental validation and real-world use cases that edge FaaS requires specialized simulation capabilities to model hardware constraints (3.8-78.3% overhead variations), network variability (0.86s vs 0.44s latency differences), and distributed orchestration complexity that traditional cloud simulators cannot accurately capture.

3.6.3 Smart City Simulation Requirements

Smart city FaaS applications demand specific simulation capabilities that extend beyond traditional cloud-centric models:

Accurate Edge Simulation with Trace-Driven Models: Smart city deployments require realistic modeling of heterogeneous edge devices with varying computational capabilities, memory constraints, and processing characteristics. Trace-driven simulation ensures high-fidelity representation of real-world device behavior and workload execution patterns essential for accurate performance prediction.

Robust Network Modeling: Urban IoT environments involve complex network topologies with varying connectivity patterns, bandwidth limitations, and latency characteristics. Comprehensive network simulation must capture geo-distributed communication patterns, mobile device interactions, and dynamic network conditions affecting data transfer between sensors, edge nodes, and cloud infrastructure.

High Configurability: Smart city scenarios encompass diverse application domains from traffic management to environmental monitoring, each requiring different deployment strategies, resource allocation policies, and performance optimization approaches. Simulator configurability enables customization of scheduling algorithms, topology generation, and workload characteristics to match specific urban deployment requirements.

Energy Modeling: Edge devices in smart city deployments are frequently battery-powered or energy-constrained, requiring longevity and efficiency optimization. Detailed energy modeling must incorporate device-specific power consumption patterns, dynamic voltage scaling effects, thermal management constraints, and network transmission energy costs to enable sustainable deployment

analysis.

Scalability Across Urban Infrastructure: Smart city applications must scale from neighborhood-level deployments to city-wide infrastructure encompassing thousands of edge devices and complex hierarchical network architectures spanning multiple administrative domains and geographical constraints.

3.6.4 Identified Research Gaps

Analysis of existing simulators against smart city requirements reveals critical gaps that limit comprehensive FaaS-edge research:

Trace-Driven Energy Integration: While faas-sim provides trace-driven execution modeling and ServerlessSimPro offers energy tracking, no simulator combines comprehensive trace-driven accuracy with detailed energy consumption modeling essential for battery-powered edge device analysis in smart city deployments.

Advanced Network-Energy Coupling: Limited integration between sophisticated network modeling and energy consumption analysis restricts understanding of energy costs associated with data transmission, network protocol overhead, and communication pattern optimization in distributed smart city infrastructures.

Smart City-Specific Validation: Existing validation studies focus primarily on generic edge computing scenarios rather than smart city-specific requirements including urban topology constraints, regulatory compliance, privacy preservation, and public sector deployment characteristics.

Unified Simulation Framework Gap: No existing simulator integrates all essential smart city requirements within a single platform. Current solutions require combining multiple tools or accepting significant limitations in energy modeling, network simulation, or edge device support, complicating research methodology and reducing result validity.

3.7 Discussion

3.7.1 Simulator Selection Rationale

The evaluation establishes clear criteria for simulator selection based on application requirements and deployment scenarios. Three distinct research contexts emerge, each requiring different simulator capabilities:

Smart City Edge-IoT Research: For comprehensive edge-IoT research requiring realistic device modeling and network simulation, faas-sim emerges as the optimal foundation. Its strengths include full edge support, trace-driven simulation capabilities, flow-based network modeling, and high configurability through its Python-based architecture. While faas-sim lacks built-in energy modeling compared to EdgeFaaS, the decision to extend faas-sim with custom energy modeling preserves its trace-driven nature and sophisticated network simulation capabilities. The Python implementation offers superior extensibility for future research compared to EdgeFaaS's more rigid framework.

Cloud Energy Research: For energy-focused research in cloud environments, ServerlessSimPro offers superior capabilities with comprehensive energy tracking and optimization algorithms.

Edge Orchestration Research: For orchestration research in edge environments, EdgeFaaS provides appropriate functionality with strong placement policies and failure handling mechanisms.

3.7.2 faas-sim as Primary Choice

faas-sim emerges as the most suitable foundation for extension, having been selected based on the following strengths:

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: Ether [Rau+20] 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.

However, it requires energy modeling development to fully meet smart city requirements, making it the optimal platform for future enhancement rather than a complete current solution.

3.7.3 Limitations and Future Needs

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

Trace Dependency: Requires profiling for new devices and functions, increasing setup effort and limiting applicability to novel scenarios without existing trace data.

Network Fidelity: Flow-based Ether model may lack precision in complex network scenarios compared to packet-level simulators.

Scalability Constraints: Memory usage (2GB) may limit very large-scale urban simulations without logging optimizations.

Profiling Overhead: Trace collection for new scenarios is resource-intensive and time-consuming.

Learning Curve: High configurability requires familiarity with SimPy framework and Python programming for effective customization.

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

3.8 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 in 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 IV

Conclusion

CHAPTER 4

CONCLUSION

This chapter provides a concise synthesis of the research findings and their implications for the field of serverless edge computing simulation. It addresses the primary research question, summarizes key contributions, acknowledges limitations, and outlines future research directions.

4.1 Research Summary

This thesis investigated the fundamental question: *Which existing FaaS simulation framework is most suitable for simulating FaaS at the edge?* and more concretely which is more suitable for a smart city scenario. Through systematic evaluation of six prominent simulation frameworks, this research established that no single existing simulator meets all requirements for realistic smart city FaaS simulation.

The study developed a complete evaluation framework examining simulation approaches, edge computing support, network modeling capabilities, configurability, and energy modeling features. Applied to ServerlessSimPro, faas-sim, EdgeFaaS, SimFaaS, MFS, and CloudSimSC, this analysis revealed faas-sim as the optimal foundation for smart city research, despite requiring energy modeling extensions.

Four critical smart city simulation requirements were identified: trace-driven accuracy, robust network modeling, high configurability, and comprehensive energy modeling. Current simulators address these requirements partially, necessitating framework extensions or multi-tool approaches for complete smart city scenario modeling.

4.2 Key Findings

Three primary findings emerged from this research:

Framework Selection: faas-sim represents the most suitable choice for smart city FaaS simulation research, offering superior trace-driven accuracy (<7% error rates), sophisticated network modeling via Ether integration [Rau+20], and high configurability through Python-based architecture.

Critical Gap Identification: No existing simulator provides complete coverage of all smart city requirements. Energy modeling represents the most significant limitation across current frameworks, requiring custom development for realistic urban IoT scenario simulation.

Requirements Framework: Four essential capabilities were established for smart city simulation: trace-driven accuracy, robust network modeling, high configurability, and energy modeling. This framework provides guidance for future simulator development and selection.

4.3 Research Contributions

This thesis makes three distinct contributions to the field of serverless edge computing simulation:

Evaluation Methodology: Development of a comprehensive framework for FaaS simulator assessment, providing standardized criteria that enable objective comparison across diverse simulation platforms. This methodology addresses the lack of systematic evaluation approaches in current literature.

Smart City Requirements Framework: Identification and formalization of four critical capabilities essential for realistic urban IoT scenario simulation. This framework bridges the gap between theoretical edge computing challenges and practical simulation requirements.

Evidence-Based Simulator Selection: Establishment of faas-sim as the optimal foundation for smart city FaaS research, with detailed rationale for energy modeling extensions that preserve core simulation strengths while addressing identified gaps.

4.4 Limitations

This research acknowledges several important limitations:

Theoretical Analysis Scope: The evaluation relied on literature review and framework documentation without empirical validation through real-world smart city deployments or comprehensive testbed experiments.

Energy Modeling Implementation: The proposed faas-sim energy modeling extensions remain conceptual, requiring future implementation and validation work.

Framework Accessibility: Limited access to some simulation frameworks may have affected the depth of comparative analysis, particularly for proprietary or poorly documented tools.

These limitations establish clear boundaries for the research contributions while providing direction for future empirical validation studies.

4.5 Future Research Directions

Three critical research directions emerge from this work:

Energy Modeling Implementation: Develop and integrate comprehensive energy models within faas-sim, focusing on heterogeneous edge device energy profiles, dynamic voltage scaling, and thermal management constraints. Validation against real edge device measurements will be essential.

Empirical Validation: Conduct real-world validation studies comparing simulation predictions with actual smart city deployments. Collaborative partnerships with urban IoT initiatives will provide access to production data for comprehensive framework validation.

Extended Application Domains: Expand simulation capabilities to address diverse smart city scenarios including traffic management, environmental monitoring, and emergency response systems. Development of standardized benchmarks will enable consistent evaluation across research groups.

4.6 Closing Remarks

This research addressed the fundamental challenge of selecting appropriate simulation frameworks for serverless edge computing in smart city environments. Through systematic evaluation of six prominent simulators, faas-sim emerged as the optimal foundation for future smart city FaaS research, despite requiring energy modeling extensions.

The established evaluation methodology and smart city requirements framework provide the research community with systematic approaches for simulator selection and development. These contributions address a critical gap in current literature and establish clear guidelines for advancing simulation capabilities in the rapidly evolving field of serverless edge computing.

As smart cities continue to mature and edge computing technologies advance, sophisticated simulation capabilities become increasingly essential. This research provides the foundation for developing next-generation FaaS simulation frameworks capable of modeling the complexity and scale of modern urban IoT deployments, ultimately supporting the design and optimization of smart city technologies that will shape our urban future.

Part V

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.

[BS22] Ali Banaei and Mohsen Sharifi. "ETAS: predictive scheduling of functions on worker nodes of Apache OpenWhisk platform." In: *The Journal of Supercomputing* 78 (Mar. 2022). DOI: 10.1007/s11227-021-04057-z.

[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.

[Dat23] Datadog. *The State of Serverless 2023*. Over 70% of AWS customers use serverless solutions. 2023. URL: https://www.datadoghq.com/state-of-serverless/.

[Dat24] DataM Intelligence. *Global Serverless Computing Market reached US$ 9.3 Billion in 2023 and is expected to reach US$ 40.9 Billion by 2031*. Growing with a CAGR of 20.6% during the forecast period 2024-2031. 2024. URL: https://www.datamintelligence.com/research-report/serverless-computing-market.

[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.

[Rau+20] Thomas Rausch et al. “Synthesizing Plausible Infrastructure Configurations for Evaluating Edge Computing Systems.” In: *Proceedings of the 13th IEEE/ACM International Conference on Utility and Cloud Computing*. UCC ’20. Leicester, United Kingdom: IEEE/ACM, 2020.

[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.