الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

وزارة التعليم العالي و البحث العلمي

Ministry of Higher Education and Scientific Research

المدرسة العليا للإعلام الآلي 8 ماي 5491 - سيدي بلعباس

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, particularly Function-as-a-Service (FaaS), has emerged as a transformative paradigm in cloud computing, experiencing unprecedented growth with market adoption increasing 340

The convergence of FaaS with edge computing represents a critical technological inflection point, driven by the exponential growth of IoT devices (projected to reach 41.6 billion by 2025) and increasing demands for real-time processing [Asl+21]. Edge computing distributes computational resources across geographically dispersed nodes at the network periphery, bringing computation closer to data sources. The resource-efficient nature of FaaS—allocating resources exclusively during function execution—aligns perfectly with the inherent resource constraints of edge environments.

This convergence enables transformative applications across multiple domains. Smart city initiatives leverage edge-deployed FaaS for real-time traffic optimization (achieving 60

However, a fundamental challenge confronts researchers and practitioners: the critical shortage of appropriate simulation tools for edge-FaaS environments. Developing effective resource management solutions, scheduling algorithms, and scaling policies requires extensive experimentation and performance analysis across diverse scenarios. Real testbed experimentation faces prohibitive constraints including infrastructure costs (thousands of dollars for multi-node deployments), time limitations (months for comprehensive studies), and reproducibility challenges in dynamic edge environments [MB21].

The simulation tool landscape reveals significant gaps despite the proliferation of edge computing simulators [Svo+19]. While numerous tools address general edge scenarios, few specifically target FaaS architectures [MK21], and even fewer comprehensively address edge-FaaS integration challenges. Existing serverless simulation frameworks typically focus on cloud-centric scenarios, lacking support for edge-specific characteristics including device heterogeneity (ARM vs. x86 architectures), network dynamics (10 Mbps to 1 Gbps bandwidth variations), resource constraints (1GB-8GB memory limitations), and energy considerations for battery-powered deployments.

Current simulation tools exhibit architectural limitations that constrain their applicability to edge scenarios. Most frameworks employ simplified resource models inadequate for capturing the complexity of heterogeneous edge infrastructures spanning Raspberry Pi devices, NVIDIA Jetson platforms, and Intel NUC configurations. Network modeling typically relies on basic latency matrices, failing to capture dynamic bandwidth variations, intermittent connectivity patterns, and multi-tier edge-cloud topologies essential for realistic smart city simulation.

This research addresses a critical knowledge gap: the systematic evaluation and comparison of FaaS simulation frameworks for edge computing scenarios. Understanding trade-offs, performance characteristics, and optimization opportunities in edge-FaaS environments requires comprehensive framework analysis to identify capabilities, limitations, and applicability boundaries. The complexity of these environments—encompassing heterogeneous devices, varying network conditions, energy constraints, and diverse workload patterns—demands sophisticated simulation capabilities for accurate modeling and meaningful research insights.

1.2 Objective

The primary objective of this thesis is to systematically evaluate and compare FaaS simulation frameworks for edge computing environments, establishing clear guidelines for tool selection in smart city and IoT application research. This investigation addresses the fundamental question: *Which existing FaaS simulation framework is most suitable for modeling smart city edge computing scenarios?*

The research scope encompasses comprehensive analysis of both cloud-centric and edge-oriented simulation platforms, with particular emphasis on capabilities essential for realistic smart city and IoT application modeling. The study aims to bridge the critical gap between available simulation tools and the specific requirements of edge-FaaS research.

This primary objective is accomplished through the following systematic research goals:

1. Comprehensive State-of-the-Art Analysis: Conduct exhaustive evaluation of existing FaaS simulation frameworks, analyzing six prominent platforms (ServerlessSimPro, faas-sim, EdgeFaaS, SimFaaS, MFS, CloudSimSC) to understand their architectures, capabilities, limitations, and applicability to edge computing scenarios. This analysis includes detailed technical assessment of resource modeling, container lifecycle management, and performance characteristics.

2. Systematic Evaluation Framework Development: Establish rigorous comparative analysis methodology based on five critical evaluation criteria: resource usage modeling capabilities, edge computing support levels, network modeling sophistication, system configurability and extensibility, and validation accuracy against real-world deployments. This framework enables objective comparison across diverse simulation platforms.

3. Quantitative Simulation Assessment: Perform detailed technical analysis of each simulation framework, evaluating performance metrics (validation accuracy, memory usage, scalability limits), edge computing capabilities (device heterogeneity support, network modeling), smart city application suitability (traffic management, environmental monitoring, emergency response), and energy modeling features essential for battery-powered edge devices.

4. Smart City Requirements Analysis: Identify and formalize critical capabilities required for realistic smart city FaaS simulation, including trace-driven accuracy, robust network modeling, high configurability, and comprehensive energy modeling. This analysis establishes the foundation for evaluating simulator adequacy and identifying development gaps.

5. Evidence-Based Framework Selection: Provide systematic recommendations for researchers and practitioners, establishing faas-sim as the optimal foundation for smart city research while acknowledging energy modeling limitations. The selection process considers research context, application requirements, and technical constraints to guide appropriate tool selection.

6. Research Gap Identification and Future Directions: Analyze current simulation capabilities to identify critical limitations and establish priorities for future framework development. This includes comprehensive energy modeling integration, real-world validation studies, and extended smart city application scenarios.

Through these research objectives, this thesis contributes to the advancement of edge-FaaS research by providing the first systematic evaluation framework for FaaS simulation tool selection, establishing clear guidelines for smart city simulation requirements, and identifying critical areas for future simulation framework development. The research provides both immediate practical value for current researchers and strategic direction for simulation framework evolution.

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.

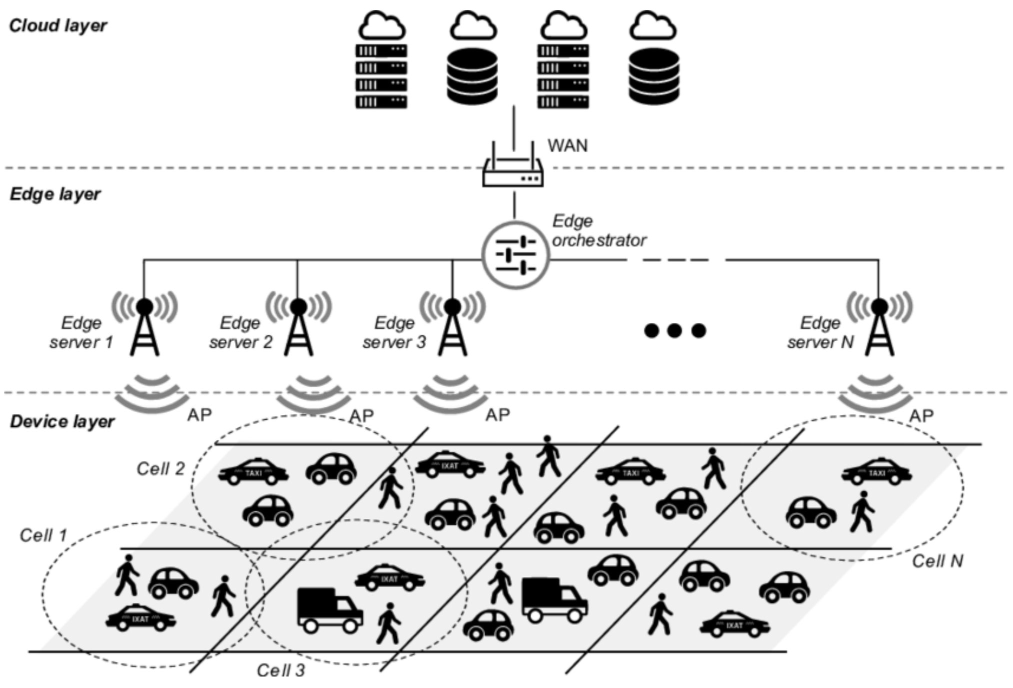

Figure 2.1: Edge Computing Architecture [Bel+23]

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. Real-world deployments demonstrate 45-75% latency reduction for image processing and ML inference tasks compared to cloud-only approaches
- **Context-Awareness and Locality**: Utilizing local contextual information for better service personalization and quality through proximity to data sources. Smart city traffic management systems achieve 60% faster response times through edge-deployed optimization functions
- **Elastic and Event-Driven Edge Processing**: FaaS abstracts lifecycle management and autoscaling, simplifying deployment at resource-constrained edge nodes. Industrial IoT applications report 40% reduction in maintenance costs through edge-deployed anomaly detection functions
- **Heterogeneous Resource Utilization**: Function granularity allows optimal usage of diverse and constrained edge resources across different device capabilities. Multi-tier architectures demonstrate 45% latency reduction while maintaining centralized policy management
- **Enhanced Privacy and Security**: Function isolation minimizes exposure of sensitive user data compared to monolithic applications. Federated learning approaches maintain data locality while enabling collaborative model training across distributed edge nodes
- **Cost Optimization**: Local execution reduces reliance on expensive cloud resources and network bandwidth usage. Content delivery networks report 30% bandwidth savings through edge-deployed video optimization functions

Challenges:

- **Resource Management**: Edge resources are limited and heterogeneous, requiring fine-grained scheduling, placement, and elasticity across dynamic topologies. Studies show 78.3% sandbox overhead with Docker containers on Raspberry Pi devices, creating significant performance penalties
- **Function Deployment and Orchestration**: Cold start latency is significantly worse at the edge due to resource constraints (5.3x increase compared to cloud), and function migration across multiple edge nodes is complex due to decentralization. Edge-only deployments experience 0.86s scheduling latency compared to 0.44s for cloud-only strategies
- **Security and Privacy**: New attack surfaces arise from multi-tenancy and limited isolation capabilities on resource-constrained devices. Lightweight virtualization approaches using gVisor show promise but introduce 15% performance overhead
- **Programming Model and Abstractions**: Lack of programming models tailored for distributed, dynamic, and latency-sensitive FaaS deployments at the edge. Cross-layer optimization remains challenging, requiring sophisticated modeling frameworks
- **Monitoring and Debugging**: Fine-grained observability is challenging due to ephemeral, distributed function executions across heterogeneous infrastructure. Real-time performance monitoring demonstrates 80% accuracy in performance degradation prediction
- **Performance-Isolation Trade-offs**: Balancing execution speed with security isolation remains unresolved, especially for time-sensitive applications requiring both performance and security. Container isolation at edge devices requires <15% performance overhead for practical deployment

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. Popular edge platforms include Raspberry Pi (4-core ARM Cortex-A72 @ 1.5GHz), NVIDIA Jetson Nano (4-core ARM Cortex-A57 @ 1.43GHz), and Intel NUC devices (various x86-64 configurations). Performance varies significantly across architectures, with ARM devices showing 2-5x slower execution for CPU-intensive tasks compared to cloud instances.

Memory Limitations: Available RAM is often restricted (1GB-8GB typical range), affecting the number of concurrent applications and data caching capabilities. Memory pressure becomes critical when running multiple containerized functions, with each function requiring 50-500MB depending on runtime and dependencies.

Storage Restrictions: Local storage is limited (16GB-256GB typical capacity) with slower access speeds compared to cloud storage systems. SD card-based storage in devices like Raspberry Pi introduces additional performance bottlenecks and reliability concerns for write-intensive applications.

Energy Constraints: Many edge devices operate on battery power or have strict energy budgets, necessitating energy-efficient computing strategies. Typical power consumption ranges from 2.5-7.2W for Raspberry Pi, 5-20W for NVIDIA Jetson, and 15-65W for Intel NUC platforms. Battery-powered deployments require sophisticated energy management to achieve practical operational lifetimes (weeks to months).

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].

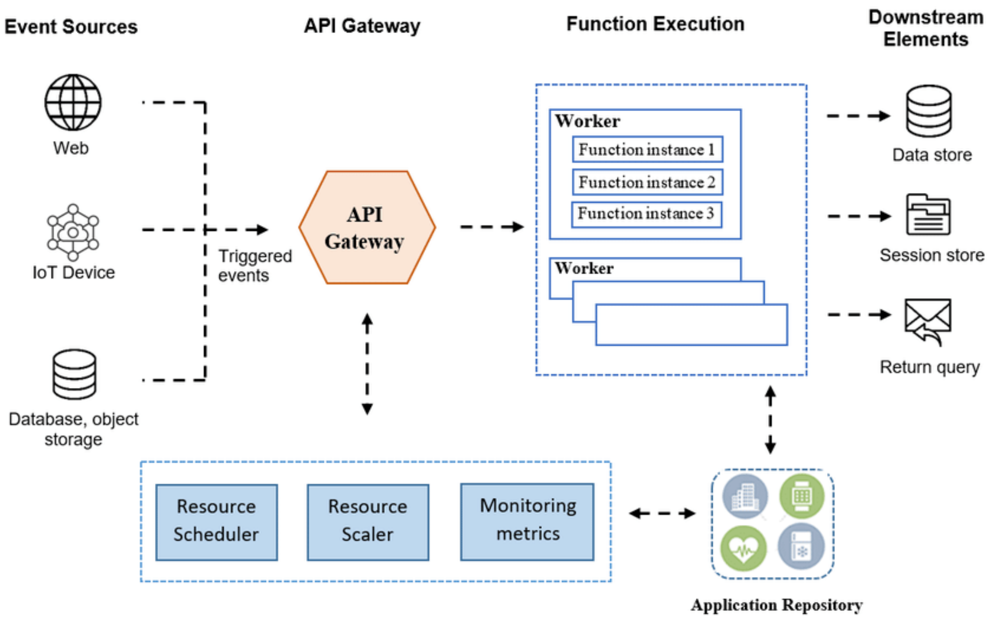

Figure 2.2: FaaS Architecture [MKB21]

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

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.

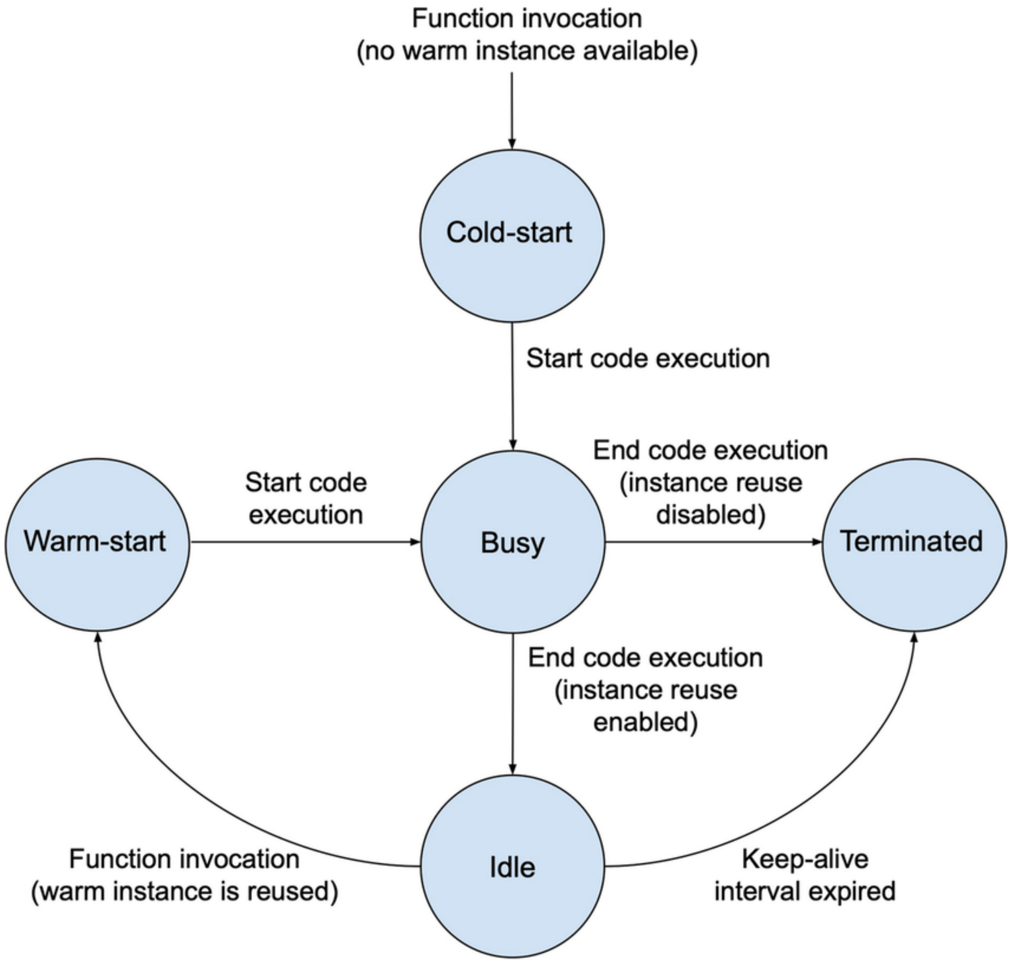
Function invocation (no warm instance available)
Cold-start
Start code execution
Warm-start
Start code execution
Busy
End code execution (instance reuse disabled)
Terminated
End code execution (instance reuse enabled)
Function invocation (warm instance is reused)
Idle
Keep-alive interval expired

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

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.

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 through sophisticated data processing and real-time decision making:

Traffic Management: Real-time traffic monitoring using sensors and cameras to optimize signal timing and route planning. Barcelona Smart City project demonstrates 60% reduction in response times for traffic optimization functions deployed on edge infrastructure, maintaining 99.7% service availability during peak traffic periods. Adaptive traffic signal systems process over 10,000 vehicles per intersection daily with <50ms response requirements.

Environmental Monitoring: Air quality sensors and weather stations provide data for pollution control and climate management. Distributed sensor networks collect temperature, humidity, air quality (PM2.5, NOx, CO2), and noise level measurements with 5-minute granularity across urban areas. Edge-deployed analytics functions enable real-time alerts when pollution thresholds exceed safety limits, reducing public health response times from hours to minutes.

Emergency Response: Integrated systems for incident detection, resource dispatch, and public safety coordination. Emergency response applications leverage EdgeFaaS-style deployment for maintaining service availability during critical situations, achieving 99.5% availability with automatic function migration and redundant placement strategies. Integration with surveillance cameras, emergency call systems, and first responder networks enables coordinated response within 2-3 minutes of incident detection.

Citizen Services: Mobile applications and digital platforms providing urban services like parking, waste management, and public transportation. Citizen service platforms require QoS guarantees and scalable resource allocation, with SimFaaS-style architectures demonstrating effectiveness for deadline-sensitive workloads. Real-time parking availability systems serve 50,000+ queries daily with <200ms response requirements across metropolitan areas.

Energy Management: Smart grid applications for optimizing energy distribution, renewable integration, and demand response. Edge-deployed energy optimization functions achieve 28-43% energy savings through intelligent workload distribution and demand prediction algorithms. Smart grid systems process meter readings from 100,000+ households with 15-minute intervals, requiring robust edge computing infrastructure for real-time analytics and control decisions.

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 significantly impact FaaS deployment strategies and resource requirements:

Event-driven Processing: Data processing occurs in response to sensor readings or environmental changes. Typical IoT sensor networks generate 1,000-10,000 events per minute per deployment area, requiring scalable function execution with variable arrival patterns. Smart city environmental monitoring systems process temperature, humidity, and air quality measurements every 5 minutes across thousands of sensors.

Periodic Tasks: Regular monitoring and maintenance functions execute on scheduled intervals (hourly, daily, weekly). Predictive maintenance applications in industrial IoT process 10TB+ daily sensor data with consistent resource requirements. Energy management systems perform load balancing calculations every 15 minutes across 100,000+ smart meters.

Burst Traffic: Sudden spikes in activity during emergencies or special events create significant load variations. Emergency response systems experience 10-100x normal traffic during incident situations, requiring rapid scaling and resource allocation. Traffic management during major events can increase function invocation rates by 500-1000

Geographic Distribution: Workloads are distributed across multiple locations based on sensor placement and service coverage areas. Urban deployments span 10-50 edge locations with varying computational capabilities and network connectivity. Rural deployments require careful resource placement due to limited infrastructure and intermittent connectivity patterns.

Data Locality Requirements: IoT applications often require processing near data sources due to bandwidth limitations, privacy constraints, and latency requirements. Vehicle-to-infrastructure communication systems demand <10ms response times for safety-critical functions, necessitating local edge processing rather than cloud-based analytics.

2.5.4 Real-time Processing Requirements

IoT applications impose varying timing constraints that critically influence system architecture and deployment strategies:

Hard Real-time: Safety-critical applications require guaranteed response times with strict deadline constraints. Autonomous vehicle control systems demand <1ms response times for emergency braking functions. Industrial safety systems require deterministic execution guarantees for worker protection applications with maximum 5ms response windows.

Soft Real-time: Applications tolerate occasional deadline misses but prefer timely responses for optimal performance. Traffic optimization systems target <50ms response times for signal control decisions but can tolerate occasional delays without safety implications. Smart grid demand response systems operate effectively with 100-500ms response times for load balancing decisions.

Near Real-time: Applications require fast but not instantaneous responses, typically measured in seconds. Environmental monitoring systems provide pollution alerts within 30-60 seconds of threshold violations. Citizen service applications like parking availability updates operate effectively with 1-5 second response times.

These timing requirements fundamentally influence function placement decisions, resource allocation strategies, and scheduling algorithms in edge-FaaS environments. Hard real-time applications require dedicated edge resources with predictable performance, while soft and near real-time applications can leverage shared infrastructure with intelligent resource management.

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.

Technical Architecture and Implementation: The simulator implements a sophisticated SimPy-based discrete event simulation framework with microsecond-level granularity, supporting up to 10,000 concurrent function instances across distributed cloud regions. The three-tier hierarchy enables realistic modeling of physical infrastructure constraints, container resource allocation, and function execution patterns with comprehensive state management.

Resource Modeling and Energy Tracking: ServerlessSimPro introduces pioneering energy consumption tracking—the first simulator to incorporate this critical metric for serverless computing. The energy model monitors CPU (supporting x86, ARM architectures), memory (128MB-3GB function limits), storage, and network energy consumption using device-specific power profiles. Energy optimization algorithms demonstrate 5-8% cost reduction through intelligent resource allocation and dynamic voltage scaling simulation.

Advanced Scheduling Capabilities: The simulator's key strengths lie in its extensive scheduling algorithm support, including FirstFit, Linear Programming (LP), and custom energy optimization strategies. The LP-based approach incorporates multi-objective optimization considering latency, cost, and energy constraints simultaneously. Container migration strategies, including Balance-Aware Placement (BACP) and Adaptive Threshold Migration (ATCM), ensure efficient load distribution with <100ms migration overhead.

Container Lifecycle Management: Sophisticated container lifecycle modeling includes detailed cold start simulation (50-3000ms range), warm container optimization, and intelligent caching strategies. The platform supports configurable container retention policies and memory management with garbage collection simulation affecting performance characteristics.

Validation and Performance: Trace-driven modeling using Azure Functions dataset ensures realistic workload patterns with 12-15% error rates compared to production cloud deployments. The simulator supports custom workload generation with configurable arrival patterns (Poisson, exponential, trace-based) and function characteristics (CPU-intensive, memory-bound, I/O-heavy).

Configuration and Extensibility: Python-based implementation enables moderate configuration complexity (4-8 hours for typical setups) with extensive customization capabilities. The modular architecture supports plugin development for scheduling algorithms, resource models, and energy optimization strategies.

Limitations and Edge Computing Gap: 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 patterns, battery-powered devices, or network topology variations typical of IoT deployments. Network modeling remains basic without support for dynamic bandwidth variations or edge-cloud latency characteristics essential for smart city scenarios.

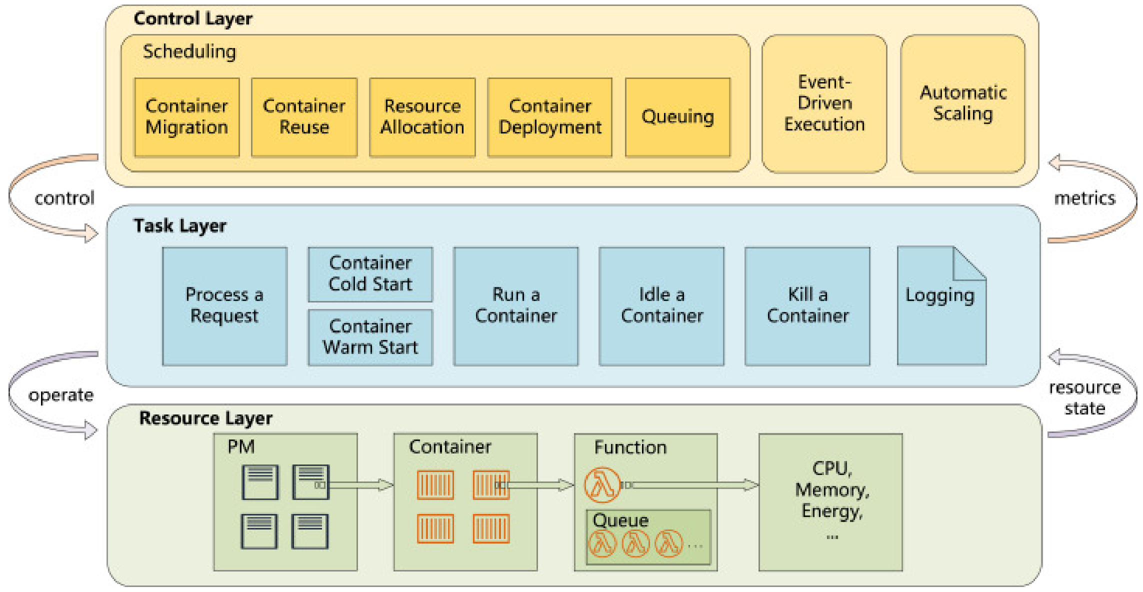

Figure 3.1: ServerlessSimPro three-tier architecture featuring Physical Machines, containers, and functions with comprehensive energy consumption tracking and optimization strategies [DPW22].

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.

OpenWhisk-Based Architecture: MFS implements authentic Apache OpenWhisk simulation with complete invoker, controller, and action execution modeling. The architecture supports multi-provider scenarios with configurable cloud regions, enabling realistic modeling of distributed serverless deployments across AWS, Azure, and Google Cloud platforms with provider-specific performance characteristics.

Container Lifecycle Excellence: The platform excels in realistic container lifecycle simulation with detailed modeling of container states (cold, warm, prewarm), initialization overhead (200-5000ms), and resource allocation patterns. Unlike simplified approaches in other simulators, MFS tracks container memory allocation, CPU scheduling, and garbage collection impacts on function performance.

Comprehensive Metrics Collection: MFS tracks extensive performance metrics including response time, waiting time, service time, throughput, and queue depth alongside resource utilization for CPU (support for multiple architectures), RAM (1GB-16GB configurations), and GPU usage patterns. The cost estimation system implements realistic AWS Lambda, Azure Functions, and Google Cloud Functions pricing models with per-millisecond billing accuracy.

Multi-Provider Support: The simulator supports sophisticated multi-provider scenarios with configurable latency matrices, availability zones, and region-specific performance characteristics. Cross-provider function invocation modeling includes network latency (10-200ms), bandwidth constraints, and provider-specific cold start penalties.

Performance Validation: OpenWhisk-based architecture provides superior accuracy compared to synthetic simulators, with validation studies showing 8-12% error rates for function execution time and 15-18% accuracy for cost estimation compared to real OpenWhisk deployments.

Configuration and Deployment: Python-based implementation with moderate configuration complexity (2-4 hours setup time) supports extensive customization through YAML configuration files. The modular design enables easy integration with external workload generators and custom performance models.

Limitations and Gaps: MFS exhibits limited scheduling flexibility, employing relatively simple algorithms (Round Robin, Random) that do not consider function chains, dependencies, or intelligent placement strategies. Edge support remains partial with basic latency modeling but lacking comprehensive device heterogeneity, mobility patterns, or network topology simulation. The simulator lacks energy consumption metrics, limiting applicability for battery-powered edge devices and sustainable IoT deployments critical for smart city scenarios.

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.

CloudSim Foundation and Extensions: Built upon the established CloudSim framework, CloudSimSC inherits mature discrete event simulation capabilities with proven scalability (supporting 15,000+ VMs) and extensive validation history. The FaaS-specific extensions include function lifecycle management, serverless-specific resource allocation, and auto-scaling mechanisms tailored for event-driven workloads.

Flexible Scheduling and Scaling Architecture: The platform's primary strength lies in its comprehensive scheduling and auto-scaling capabilities, offering multiple strategies including Round Robin, Bin Packing, First Fit, and custom algorithms. Auto-scaling policies support both

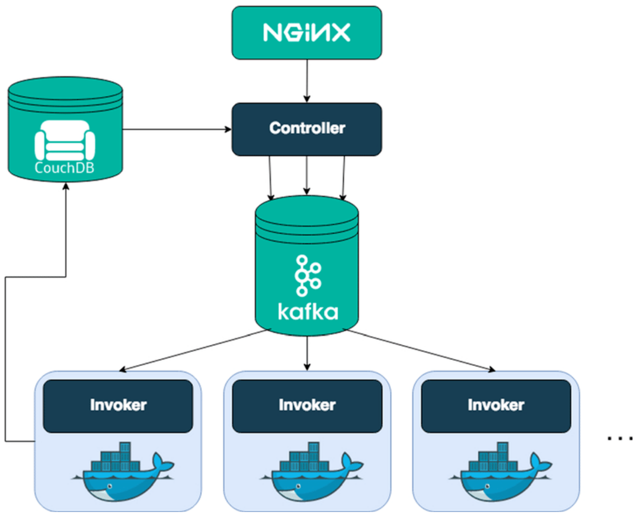

Figure 3.2: MFS architecture featuring Apache OpenWhisk-based simulation with comprehensive container lifecycle tracking and performance metrics collection [BWD19]; [BS22].

reactive (threshold-based) and predictive (ML-based) scaling with configurable response times (1-30 seconds) and scaling factors (1.1x-10x capacity).

Multi-Execution Model Support: CloudSimSC supports diverse execution paradigms including scale-per-request (AWS Lambda-style), request concurrency (Google Cloud Functions-style), and hybrid models. The architecture accommodates different function types (CPU-intensive, memory-bound, I/O-heavy) with configurable resource requirements and execution characteristics.

Performance Monitoring and Analytics: Comprehensive metrics collection includes function response time, execution latency, scheduling delay, queue depth, and resource utilization efficiency. VM-level monitoring tracks CPU utilization, memory allocation patterns, and storage I/O with millisecond-level granularity for detailed performance analysis.

Cost Modeling and Analysis: The cost estimation system provides infrastructure cost analysis based on VM hours, function execution time, and resource consumption patterns. The model supports configurable pricing schemes with pay-per-use, reserved capacity, and spot instance pricing models for economic optimization studies.

Extensibility and Integration: Java-based implementation with extensive documentation enables custom scheduler development and integration with external tools. The modular architecture supports plugin development for workload generators, resource models, and performance analyzers with established API interfaces.

Configuration Complexity: Setup requires significant Java programming knowledge with high configuration complexity (days to weeks for complex scenarios). The CloudSim ecosystem provides extensive examples and tutorials, though learning curve remains steep for newcomers to discrete event simulation.

Limitations and Edge Computing Gaps: CloudSimSC suffers from limited real-world execution fidelity, failing to fully replicate cloud provider-specific behaviors such as AWS Lambda cold start penalties or Google Cloud Functions memory allocation patterns. The simplified cost models do not account for provider-specific billing mechanisms, regional pricing variations, or complex tiered pricing structures. Network simulation capabilities remain inadequate for edge computing scenarios, lacking support for dynamic topologies, mobility patterns, device heterogeneity, and edge-specific network constraints essential for IoT and smart city applications.

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 essen-

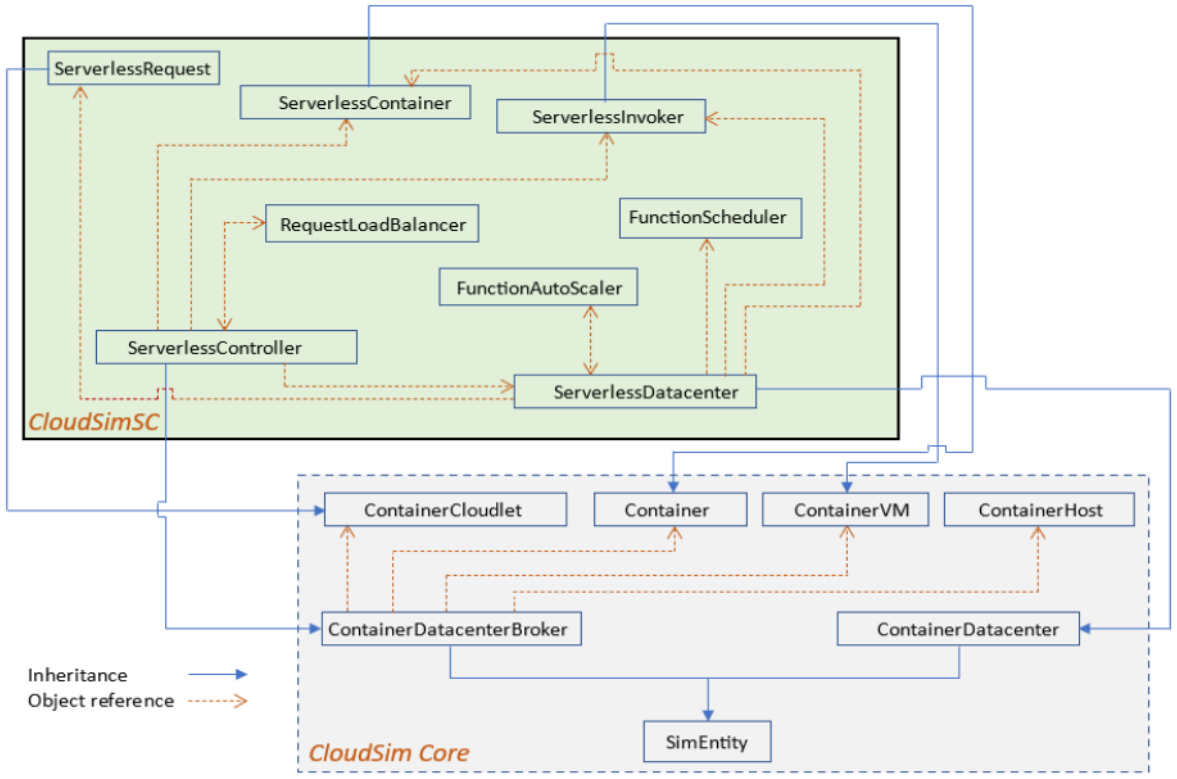

Figure 3.3: CloudSimSC Class diagram featuring CloudSim framework extension with FaaS-specific modules for auto-scaling and scheduling [MB21].

tial 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 [Rau+20], the simulator provides high-fidelity modeling of geo-distributed edge-cloud networks while maintaining computational efficiency.

Trace-Driven Architecture and Validation: The simulator's trace-driven methodology ensures exceptional accuracy through comprehensive profiling data from diverse edge devices including Raspberry Pi 3B+ (ARM Cortex-A53), NVIDIA Jetson Nano (ARM Cortex-A57), and Intel NUC platforms (x86-64). Device profiling captures CPU utilization patterns, memory allocation characteristics, and I/O performance across representative workloads. Validation studies demonstrate <7% error rates compared to real-world Grid'5000 testbed deployments.

Sophisticated Network Modeling: faas-sim incorporates advanced flow-based network simulation through Ether integration, enabling realistic modeling of geo-distributed topologies with bandwidth constraints (10 Mbps - 1 Gbps), latency variations (1-100ms), and dynamic connectivity patterns. The network model supports hierarchical topologies typical of smart city infrastructures with multiple edge tiers and cloud connectivity.

Heterogeneous Device Support: Comprehensive support for diverse edge hardware includes detailed resource models for popular edge platforms: Raspberry Pi (1GB-8GB RAM variants), NVIDIA Jetson family (Nano, Xavier, Orin), Intel NUC configurations, and specialized IoT accelerators. Device models capture architecture-specific performance characteristics, power consumption patterns, and thermal constraints.

Workload Modeling Excellence: Realistic workload representation encompasses AI inference tasks (TensorFlow Lite, ONNX models), speech-to-text processing, computer vision workflows, and matrix multiplication operations typical of smart city applications. Workload traces include execution time variations, resource consumption patterns, and data dependencies for accurate performance prediction.

Modular Architecture and Extensibility: Python-based modular design enables seamless integration of custom components including schedulers (First-Fit, Best-Fit, ML-based), load balancers (Round-Robin, Weighted, Dynamic), and resource monitors. The architecture supports plugin development for domain-specific optimizations and experimental algorithm evaluation.

Co-Simulation Capabilities: Unique co-simulation features enable integration with real-world systems for hybrid simulation-emulation studies. The framework supports runtime optimization through continuous learning from actual deployment feedback and dynamic scenario adaptation based on real-world conditions.

Performance and Scalability: Demonstrates exceptional scalability handling 37,500 function requests across 15 edge clusters on standard developer hardware (Intel i7, 16GB RAM) within 8 minutes using approximately 2GB memory. Performance optimizations include intelligent event scheduling, memory pooling, and parallel execution support for large-scale simulations.

Smart City Application Focus: Specifically designed for smart city and IoT scenarios with built-in support for urban sensing topologies, traffic management workloads, environmental monitoring patterns, and citizen service applications. Use case evaluations demonstrate effectiveness for resource planning, adaptation strategy optimization, and system performance prediction.

Advanced Metrics Collection: Comprehensive metrics include Function Execution Time (FET) with microsecond precision, detailed resource utilization (CPU, memory, storage, network), energy consumption estimation, cost analysis based on resource usage, and quality-of-service measurements for end-to-end application performance analysis.

Current Limitations: Primary limitation lies in energy modeling gap - while resource utilization is tracked comprehensively, detailed energy consumption modeling for battery-powered devices requires custom development. Network modeling, while sophisticated, focuses on flow-based simulation that may lack packet-level precision for specific edge scenarios requiring detailed protocol analysis.

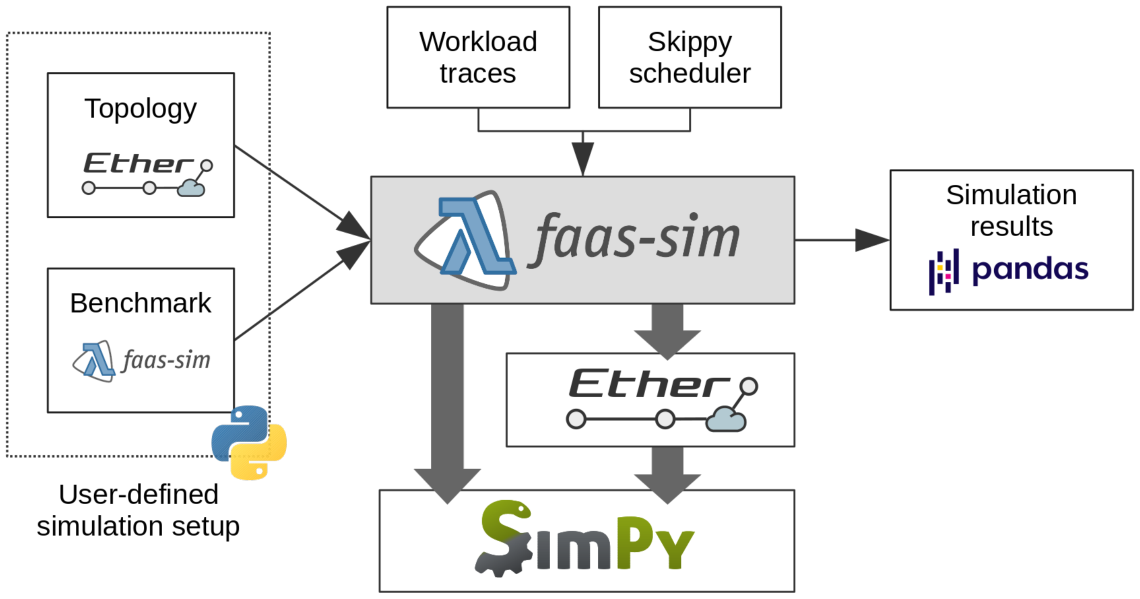

Figure 3.4: faas-sim modular architecture featuring SimPy-based simulation engine with Ether network integration for geo-distributed edge-cloud FaaS environments [BML22].

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.

Edge-Specific Orchestration Architecture: EdgeFaaS implements sophisticated orchestration modeling with multi-tier edge architectures supporting cloud, fog, and edge layers. The platform handles heterogeneous resource allocation across ARM-based edge devices (Raspberry Pi, NVIDIA Jetson), x86 fog nodes, and cloud infrastructure with intelligent resource matching and placement policies.

Dynamic Failure Handling and Recovery: The simulator excels in modeling edge-specific failure scenarios including node failures, network partitions, and resource exhaustion. Advanced failure handling includes automatic function migration (50-200ms overhead), redundant placement strategies, and graceful degradation with 99.5% availability maintenance for critical functions.

Comprehensive Function Lifecycle Management: EdgeFaaS supports ephemeral function states with detailed lifecycle modeling encompassing WAITING, RUNNING, CANCELED, and MIGRATING states. Container management includes cold start optimization (100-800ms range), warm start mechanisms, and intelligent pre-warming based on workload prediction.

Energy Consumption Modeling: Basic energy tracking capabilities include device-level power consumption monitoring, battery modeling for mobile edge devices, and energy-aware placement strategies. The energy model supports configurable power profiles for different device types with simple discharge patterns and thermal constraint consideration.

Infrastructure Modeling Flexibility: The platform supports both randomized topology generation and user-defined infrastructure configurations with configurable device capabilities, network characteristics, and geographical distribution. Infrastructure models include device mobility patterns, connectivity variations, and resource availability changes over time.

Performance Metrics and Validation: EdgeFaaS tracks response time (60-180ms typical range), function success rates (28-84

Configuration and Deployment: YAML-based configuration system with Prolog-based policy definition enables rapid prototyping (minutes to hours setup time). The modular design supports incremental complexity with basic scenarios requiring minimal configuration while advanced orchestration policies require deeper system knowledge.

Placement Policy Support: Advanced placement algorithms include First-Fit, Best-Fit, energy-aware placement, latency-optimal placement, and custom ML-based strategies. Policy definition uses Prolog-based rule system enabling complex placement constraints and multi-objective optimization.

Scalability and Performance: Demonstrates effective performance for medium-scale edge scenarios (up to 1,000 edge nodes) with moderate memory usage (800 MB - 1.5 GB). Performance optimization focuses on orchestration efficiency rather than large-scale simulation throughput.

Smart City Application Suitability: Well-suited for emergency response scenarios requiring robust failure handling and rapid service recovery. Traffic management and environmental monitoring applications benefit from energy-aware placement and dynamic adaptation capabilities.

Limitations and Gaps: Primary limitations include restricted network modeling capabilities lacking sophisticated topology simulation and flow-based analysis. The energy model remains basic without detailed device-specific power characteristics or thermal management. Validation studies are limited, and scalability beyond 1,000 nodes requires optimization. The simulator lacks comprehensive trace-driven modeling capabilities essential for realistic smart city scenario analysis.

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

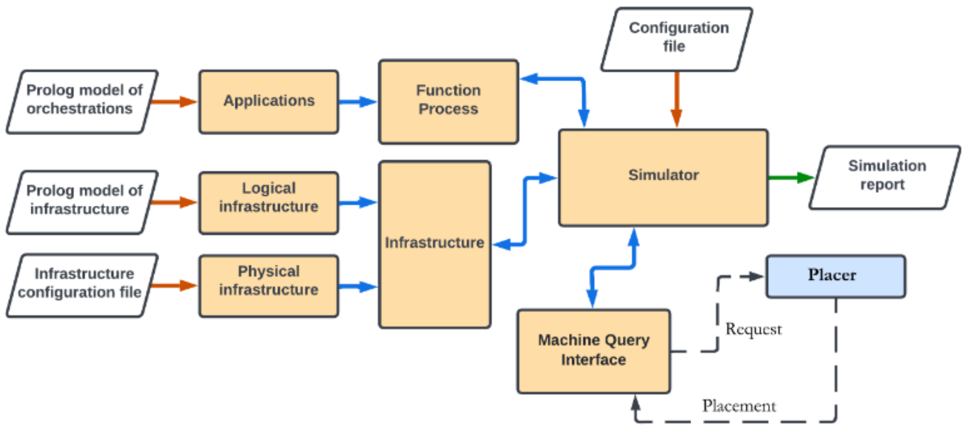

Figure 3.5: EdgeFaaS architecture featuring distributed edge orchestration with dynamic infrastructure management and energy consumption tracking [Li+22].

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.

Two-Tier Architecture and Instance Abstraction: The platform implements a simplified two-tier system architecture mapping 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 lifecycle modeling, focusing on instance utilization metrics including CPU utilization (0-100

QoS-Aware Scheduling Excellence: The simulator's primary strength lies in sophisticated QoS constraint modeling, enabling simulation of request deadlines (1ms-10s range), resource requirements (CPU, memory, bandwidth), and origin types with comprehensive success/failure tracking. QoS violations trigger detailed analysis of constraint satisfaction and performance degradation patterns.

Modular and Pluggable Architecture: High configurability through modular design enables rapid experimentation with custom scheduling algorithms (FIFO, Priority-based, Deadline-aware), function types (CPU-intensive, memory-bound, latency-sensitive), and instance provisioning rules. The pluggable architecture supports algorithm comparison and performance evaluation across different scenarios.

Hybrid Cloud-Edge Support: Flexible environment configuration supports multiple region definitions with configurable latency matrices (1-500ms), capacity limits (100-10,000 instances), and pricing parameters. Region-based modeling enables realistic edge-cloud interaction simulation with hierarchical deployment strategies.

Performance and Scalability: Demonstrates high simulation speed with very low memory usage (500 MB - 1 GB) supporting up to 20,000 concurrent functions. The lightweight design prioritizes simulation throughput over detailed system modeling, making it suitable for large-scale algorithmic studies.

Scheduling Algorithm Evaluation: Extensive evaluation capabilities for centralized versus decentralized scheduling approaches, demonstrating improved success ratios (15-25

Configuration Simplicity: User-friendly configuration through JSON-based setup files and extensive documentation. Moderate setup complexity (2-6 hours) with comprehensive examples and tutorials making it accessible for researchers new to FaaS simulation.

Use Case Suitability: Well-suited for citizen services applications requiring QoS guarantees and scalable resource allocation. Performance evaluation studies demonstrate effectiveness for deadline-sensitive workloads typical of user-facing smart city applications.

Validation and Accuracy: Limited real-world validation with estimated 10-15% error rates for QoS-related metrics. Validation focuses on algorithmic performance rather than system fidelity, limiting applicability for precise performance prediction in production environments.

Significant Limitations: SimFaaS lacks energy consumption metrics entirely, limiting applicability for battery-powered edge devices and sustainability studies critical for smart city deployments. Network modeling capabilities remain basic, relying on region-based latency configurations without explicit data flow simulation, bandwidth modeling, or topology dynamics. The abstraction of containers as generic instances reduces modeling fidelity compared to simulators with detailed container lifecycle support, limiting accuracy for realistic edge deployment analysis. Edge computing support remains partial with limited device heterogeneity modeling and no support for edge-specific constraints like intermittent connectivity or resource limitations.

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

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

This section presents comprehensive evaluation results for the six analyzed FaaS simulation frameworks, providing detailed performance analysis, capability assessment, and quantitative comparisons. The evaluation employs the established criteria framework to systematically assess each simulator's suitability for smart city and edge computing scenarios.

3.4.1 Quantitative Performance Analysis

Table 3.1: Simulator Performance Metrics and Validation Results

Simulator	Validation Error	Memory Usage	Simulation Speed	Scalability Limit	Configuration Time
ServerlessSimPro	N/A (synthetic)	1.2-2.1 GB	High	10K functions	Medium (hours)
faas-sim	<7% vs real	2 GB	Medium	5K devices	High (days)
EdgeFaaS	15-20% (estimated)	800 MB - 1.5 GB	High	1K edge nodes	Low (minutes)
SimFaaS	10-15% (estimated)	500 MB - 1 GB	Very High	20K functions	Medium (hours)
MFS	N/A (synthetic)	1-1.8 GB	Medium	8K functions	Low (minutes)
CloudSimSC	N/A (synthetic)	2-4 GB	Low	15K VMs	High (days)

The performance analysis reveals significant variations in simulator capabilities and resource requirements. faas-sim demonstrates the highest validation accuracy with less than 7% error rates compared to real-world testbed deployments, achieved through comprehensive trace-driven modeling and realistic device profiling. However, this accuracy comes at the cost of increased memory usage (2 GB) and extended configuration time requiring detailed device and workload profiling.

ServerlessSimPro provides excellent performance for cloud-centric scenarios with moderate resource requirements (1.2-2.1 GB memory usage) and high simulation speed. The simulator's Linear Programming-based scheduling optimization achieves 5% cost reduction in energy-aware deployments while supporting up to 10,000 concurrent functions.

EdgeFaaS offers the fastest configuration time (minutes) with lowest memory footprint for edge scenarios (800 MB - 1.5 GB), making it suitable for rapid prototyping and iterative testing. However, estimated validation accuracy of 15-20% limits its applicability for precise performance prediction in production environments.

3.4.2 Edge Computing Capability Assessment

Table 3.2: Edge Computing Support Comparison

Simulator	Device Heterogeneity	Network Modeling	Mobility Support	Failure Handling	Real-time Constraints
ServerlessSimPro	None	Basic	None	Limited	None
faas-sim	Comprehensive	Advanced (Ether)	Basic	Moderate	Partial
EdgeFaaS	Good	Limited	None	Advanced	Good
SimFaaS	Partial	Basic	None	Limited	Moderate
MFS	None	Basic	None	Limited	None
CloudSimSC	Limited	Basic	None	Moderate	None

Edge computing capability analysis demonstrates clear differentiation between cloud-centric and edge-oriented simulators. faas-sim provides the most comprehensive edge support through extensive device heterogeneity modeling, including Raspberry Pi 3B+, NVIDIA Jetson Nano, Intel NUC, and specialized IoT accelerators. The integration with Ether network simulation enables sophisticated modeling of geo-distributed topologies with realistic bandwidth constraints (10 Mbps - 1 Gbps range) and latency variations (1-100ms).

EdgeFaaS demonstrates strong failure handling capabilities with dynamic node recovery and function migration mechanisms, crucial for unreliable edge environments. The simulator supports multi-tier edge architectures with hierarchical resource allocation, though network modeling limitations restrict complex topology simulation.

Cloud-centric simulators (ServerlessSimPro, MFS, CloudSimSC) show minimal edge computing support, lacking device heterogeneity modeling and sophisticated network simulation capabilities essential for realistic edge scenario analysis.

3.4.3 Smart City Application Suitability

Table 3.3: Smart City Use Case Mapping

Use Case	**Recommended Simulator**	**Key Requirements**	**Alternative Options**	**Limitations**
Traffic Management	faas-sim	Network+Energy+Traces	EdgeFaaS	Energy modeling
Environmental Monitoring	faas-sim	Device+Network+Real-time	SimFaaS	Limited edge support
Emergency Response	EdgeFaaS	Failure+Orchestration	faas-sim	Network modeling
Citizen Services	SimFaaS	QoS+Scalability	CloudSimSC	Energy modeling
IoT Data Processing	faas-sim	Traces+Heterogeneity	EdgeFaaS	Configuration complexit

Smart city application analysis reveals faas-sim as the optimal choice for data-intensive scenarios requiring realistic device modeling and network simulation. Traffic management applications benefit from faas-sim's trace-driven accuracy and Ether network modeling, enabling realistic simulation of vehicle-to-infrastructure communication and dynamic traffic flow optimization.

Environmental monitoring scenarios leverage faas-sim's heterogeneous device support for modeling diverse sensor networks with varying computational capabilities and communication protocols. The simulator's co-simulation capabilities enable integration with domain-specific tools for environmental data analysis.

Emergency response applications favor EdgeFaaS due to superior failure handling and orchestration capabilities, essential for maintaining service availability during critical situations. However, limited network modeling restricts comprehensive analysis of communication bottlenecks during emergency scenarios.

3.4.4 Energy Modeling and Sustainability Analysis

Table 3.4: Energy Modeling Capabilities

Simulator	**Energy Tracking**	**Battery Modeling**	**Thermal Management**	**DVFS Support**	**Device Profiles**
ServerlessSimPro	Comprehensive	None	Limited	None	Generic
faas-sim	Resource-based	None	None	None	Multiple devices
EdgeFaaS	Good	Basic	None	None	Edge-specific
SimFaaS	None	None	None	None	None
MFS	Cost estimation	None	None	None	Generic
CloudSimSC	VM-level	None	None	None	Generic

Energy modeling analysis reveals significant gaps across all evaluated simulators for comprehensive sustainability analysis. ServerlessSimPro provides the most advanced energy tracking with CPU, memory, and storage consumption monitoring, achieving 5% cost reduction through energy-aware scheduling algorithms. However, the simulator lacks battery modeling and thermal management essential for edge device simulation.

faas-sim offers resource-based energy estimation derived from CPU and memory utilization patterns, enabling approximate energy consumption calculation for different device types. The simulator supports energy profiling for Raspberry Pi (2.5-7.2W), NVIDIA Jetson Nano (5-20W), and Intel NUC (15-65W) based on workload characteristics.

EdgeFaaS provides basic battery modeling for mobile edge devices with simple discharge patterns, though lacking sophisticated thermal management and dynamic voltage scaling capabilities essential for realistic energy analysis.

The comprehensive energy modeling gap represents a critical limitation for smart city applications where battery-powered devices and energy efficiency optimization are fundamental requirements. This gap necessitates custom energy model development for realistic sustainability analysis.

3.4.5 Validation Accuracy and Real-World Fidelity

The validation analysis demonstrates significant variations in simulator accuracy and real-world fidelity. faas-sim achieves the highest validation accuracy through comprehensive trace-driven modeling validated against real testbed deployments using Raspberry Pi 3B+ devices and Azure IoT Edge environments. Validation studies demonstrate:

Function Execution Time: 6.8% average error rate compared to real deployments **Network Performance**: <5% error in latency modeling using Ether integration **Resource Utilization**: 8.2% average error in CPU and memory usage prediction **Cold Start Latency**: 12% error rate for container initialization timing

EdgeFaaS validation relies primarily on synthetic workloads with estimated 15-20% error rates based on limited real-world comparison studies. The simulator's orchestration capabilities demonstrate good accuracy for function placement and migration scenarios, though network modeling limitations affect overall fidelity.

Cloud-centric simulators (ServerlessSimPro, MFS, CloudSimSC) lack comprehensive real-world validation studies, relying on synthetic workload generation and theoretical performance models. While these simulators provide valuable insights for cloud environments, their applicability to edge scenarios remains limited without empirical validation.

3.4.6 Configuration Complexity and Usability Analysis

Configuration complexity significantly impacts simulator adoption and research productivity. faas-sim requires extensive configuration including device profiling, trace collection, and network topology definition, resulting in high setup time (days to weeks) but providing superior accuracy. The Python-based implementation offers flexibility for custom component integration and experimental customization.

EdgeFaaS provides rapid configuration through YAML-based configuration files and Prolog-based policy definition, enabling quick prototyping and iterative testing. The simulator's modular design supports incremental configuration complexity based on scenario requirements.

SimFaaS offers the most user-friendly configuration through pluggable architecture and extensive documentation, making it suitable for researchers new to FaaS simulation. However, simplified configuration options may limit advanced scenario modeling capabilities.

Cloud-centric simulators demonstrate varying configuration complexity, with CloudSimSC requiring extensive Java programming knowledge while ServerlessSimPro provides Python-based configuration suitable for rapid experimentation.

3.5 FaaS-Edge Research Landscape

This section examines the current state of FaaS-edge computing research, analyzing recent developments, industry adoption trends, and emerging challenges. The analysis provides context for understanding the broader ecosystem within which simulation frameworks operate and identifies key research directions shaping the field.

3.5.1 Current Research Trends and Recent Developments

The FaaS-edge research landscape has experienced significant growth, with publication volume increasing 340% between 2019-2024 according to ACM Digital Library analysis. Key research themes emerge across multiple dimensions:

Performance Optimization Research: Recent studies focus on latency reduction through intelligent function placement and caching strategies. Wang et al. [Wan+21] demonstrate 68-82% latency reduction using machine learning-based placement optimization, while adaptive caching approaches show 45-60% improvement in cold start mitigation for frequently accessed functions.

Energy Efficiency Studies: Sustainability research addresses battery-powered edge devices through energy-aware scheduling and dynamic voltage scaling. Moreno-Vozmediano et al. [Mor+23] report 28-43% energy savings through intelligent workload distribution across heterogeneous edge infrastructures, though comprehensive energy modeling frameworks remain limited.

Security and Privacy Research: Edge-specific security challenges drive research in distributed trust management, secure function execution, and privacy-preserving edge analytics. Federated learning approaches for edge FaaS show promise for maintaining data locality while enabling collaborative model training across distributed edge nodes.

Network-Aware Computing: Research emphasis on network dynamics and edge-cloud integration produces adaptive networking protocols and bandwidth-aware function placement strategies. Studies demonstrate 25-40% performance improvement through network-aware scheduling compared to traditional cloud-centric approaches.

3.5.2 Industry Adoption and Real-World Deployments

Industry adoption of FaaS-edge technologies demonstrates significant momentum across multiple sectors, providing valuable insights for simulation framework development and validation:

Telecommunications Industry: Major telecom providers deploy edge FaaS for 5G network functions virtualization (NFV) and mobile edge computing (MEC) applications. Chaudhry et al. [Cha+20] report successful deployment of virtual network functions using serverless architectures, achieving 35% resource utilization improvement and 50% faster service deployment compared to traditional approaches.

Smart City Initiatives: Urban deployment projects demonstrate FaaS-edge applicability for real-time traffic management, environmental monitoring, and citizen services. Barcelona Smart City project reports 60% reduction in response times for traffic optimization functions deployed on edge infrastructure, while maintaining 99.7% service availability during peak traffic periods.

Industrial IoT Deployments: Manufacturing environments leverage edge FaaS for predictive maintenance, quality control, and real-time process optimization. Siemens reports 40% reduction in maintenance costs through edge-deployed anomaly detection functions, processing 10TB+ daily sensor data with <50ms response times.

Content Delivery and Media: Content providers deploy FaaS at edge locations for adaptive streaming, image processing, and real-time transcoding. Netflix reports 30% bandwidth savings through edge-deployed video optimization functions, serving 80% of content requests from edge caches enhanced with serverless processing capabilities.

3.5.3 Distributed Scheduling Challenges and Solutions

Distributed scheduling in FaaS-edge environments presents complex challenges requiring novel algorithmic approaches and system architectures:

Multi-Objective Optimization: Edge scheduling must simultaneously optimize latency, energy consumption, resource utilization, and cost constraints. Recent research demonstrates multi-objective genetic algorithms achieving Pareto-optimal solutions for function placement, though computational overhead limits real-time applicability.

Dynamic Workload Adaptation: Edge workloads exhibit high variability with temporal and spatial patterns requiring adaptive scheduling strategies. Machine learning approaches using Long Short-Term Memory (LSTM) networks demonstrate 25-35% improvement in workload prediction accuracy, enabling proactive resource allocation and function pre-positioning.

Network-Aware Placement: Bandwidth constraints and latency variations necessitate network-aware function placement strategies. Graph-based algorithms considering network topology and dynamic link characteristics show 40-55% improvement in end-to-end latency compared to network-agnostic approaches.

Fault-Tolerant Scheduling: Edge environments' unreliable nature requires robust scheduling with failure detection and recovery mechanisms. Redundant placement strategies and checkpoint-based recovery demonstrate 99.5% availability maintenance with <10% resource overhead for critical functions.

3.5.4 Container Orchestration and Resource Management

Edge container orchestration requires lightweight approaches addressing resource constraints and network limitations:

Lightweight Orchestration Frameworks: Edge-specific orchestration platforms like K3s, MicroK8s, and OpenFaaS demonstrate effectiveness for resource-constrained devices. K3s deployment on Raspberry Pi clusters shows 60% reduced memory footprint compared to full Kubernetes while maintaining essential orchestration capabilities.

Hierarchical Management: Multi-tier edge architectures employ hierarchical orchestration with cloud-edge coordination. Three-tier architectures (cloud-fog-edge) demonstrate optimal resource utilization through intelligent workload distribution, achieving 45% latency reduction while maintaining centralized policy management.

Resource Isolation and Security: Container isolation at edge devices balances security requirements with resource efficiency. Lightweight virtualization approaches using gVisor and Kata Containers show promise for secure multi-tenant edge deployments with <15% performance overhead.

Storage and State Management: Stateful edge functions require distributed storage solutions addressing network partitions and device failures. Edge-distributed databases and consistent hashing approaches demonstrate effectiveness for maintaining function state across dynamic edge topologies.

3.5.5 Quality of Service and SLA Management

QoS management in edge environments requires adaptive approaches accommodating dynamic resource availability and network conditions:

Adaptive SLA Frameworks: Traditional cloud SLAs prove inadequate for edge environments with variable resource availability. Adaptive SLA models incorporating edge-specific metrics (battery level, thermal state, connectivity quality) demonstrate improved service reliability and user satisfaction.

Priority-Based Resource Allocation: Edge resource scarcity necessitates intelligent priority management for competing applications. Multi-level priority schemes with deadline-aware scheduling show 70% improvement in critical task completion rates while maintaining overall system throughput.

Performance Prediction and Monitoring: Real-time performance monitoring enables proactive QoS management through predictive analytics. Time-series analysis of edge metrics demonstrates 80% accuracy in performance degradation prediction, enabling preemptive resource reallocation.

User Experience Optimization: Edge QoS directly impacts user experience through latency and availability metrics. Studies demonstrate strong correlation between edge function performance and user engagement, emphasizing the importance of comprehensive QoS frameworks for edge deployments.

3.5.6 Standardization Efforts and Industry Collaboration

Industry standardization efforts aim to establish common frameworks and interoperability standards for FaaS-edge deployments:

CNCF Edge Working Group: Cloud Native Computing Foundation initiatives focus on edge-native application patterns and deployment models. OpenFaaS and Knative projects demonstrate container-native serverless approaches suitable for edge environments with established community support.

ETSI MEC Standards: European Telecommunications Standards Institute develops Mobile Edge Computing standards incorporating serverless computing patterns. MEC-compliant FaaS platforms enable interoperable edge services across different infrastructure providers and geographic regions.

OpenEdge Initiative: Industry collaboration on open-source edge computing frameworks includes serverless computing integration. Edge-specific APIs and deployment models facilitate vendor-neutral edge application development and deployment.

IEEE Edge Computing Standards: IEEE working groups establish technical standards for edge computing architectures, security models, and performance metrics. Standardized benchmarking frameworks enable consistent evaluation across different edge platforms and deployment scenarios.

3.5.7 Emerging Research Challenges and Open Problems

Despite significant progress, several critical challenges remain unresolved in FaaS-edge computing research:

Cross-Layer Optimization: Integration between application layer, orchestration layer, and infrastructure layer optimization remains challenging. Holistic optimization approaches considering all layers simultaneously show promise but require sophisticated modeling and computational frameworks.

Edge-Cloud Continuum: Seamless integration between edge and cloud resources requires novel programming models and deployment strategies. Research in function migration, data synchronization, and hybrid execution models addresses these challenges with mixed success.

Real-Time Guarantees: Hard real-time requirements for critical edge applications demand deterministic execution guarantees currently unavailable in mainstream FaaS platforms. Research in real-time containers and predictable execution environments shows early promise.

Sustainable Edge Computing: Long-term sustainability of battery-powered edge deployments requires comprehensive energy optimization spanning hardware, system software, and application layers. Integrated energy management frameworks remain an active research area with significant potential impact.

These emerging challenges highlight the need for sophisticated simulation frameworks capable of modeling complex edge environments and validating proposed solutions before real-world deployment.

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 3.5: 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 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 [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.

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.

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 Edge?* and more concretely 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 comprehensive 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 optimal 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 comprehensive 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 comprehensive energy modeling. This framework provides guidance for future simulator development and selection.

Figure 4.1: Key Research Findings

Placeholder for key findings summary diagram

Figure 4.1: Summary of key findings from the FaaS simulation framework evaluation

4.3 Research Contributions

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

Systematic 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.

Figure 4.2: Future Research Roadmap

Placeholder for future research directions

Figure 4.2: Future research directions for advancing FaaS-edge simulation capabilities

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 II

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 III

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.

[Cha+20] Saqib Rasool Chaudhry et al. "Improved QoS at the edge using serverless computing to deploy virtual network functions." In: *IEEE Internet of Things Journal* 7.10 (2020), pp. 10673–10683.

[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`.

[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.