ComFaaS: A Dynamic Approach to Edge and Cloud Computing with Function-as-a-Service

Jaden Jinu Lee
Computer Science & Information Systems Department
Minnesota State University Moorhead
Moorhead, Minnesota, USA
jinu.lee@go.minnesota.edu

Judah J. Nava*
Computer Science & Information Systems Department
Minnesota State University Moorhead
Moorhead, Minnesota, USA
judah.nava@go.mnstate.edu

Hanku Lee
Computer Science & Information Systems Department
Minnesota State University Moorhead
Moorhead, Minnesota, USA
hanku.lee@mnstate.edu

Abstract

The rapid evolution of cloud and edge computing has redefined how data-intensive applications are developed and deployed, with Function-as-a-Service (FaaS) playing a pivotal role in this transformation. FaaS provides a serverless model where functions are executed in response to specific events, offering developers automatic scalability, high availability, and reduced infrastructure management overhead. The latest release of ComFaaS brings substantial improvements in flexibility, scalability, security, and ease of use. It introduces a dynamic architecture that enables FaaS applications to be added and executed at runtime, without the need to modify the core system, streamlining feature integration and enhancing scalability. ComFaaS also includes dynamic load balancing, which intelligently distributes workloads between edge and cloud environments, ensuring that tasks always benefit from the most efficient computing resources available. This hybrid approach allows edge and cloud computing to complement each other, resulting in optimized performance tailored to the specific needs of each application. The fully functional release of ComFaaS now delivers a powerful and adaptable solution for modern FaaS deployments, offering a secure and scalable platform for both cloud and edge environments.

CCS Concepts

• **Distributed computing methodologies**; • **Concurrent computing methodologies**; • **Parallel computing methodologies**;

Keywords

Edge Computing, FaaS, Event-Driven, Cloud Computing, Serverless Computing

ACM Reference Format:

Jaden Jinu Lee, Judah J. Nava, and Hanku Lee. 2024. ComFaaS: A Dynamic Approach to Edge and Cloud Computing with Function-as-a-Service. In *2024 the 9th International Conference on Cloud Computing and Internet of Things (CCIOT) (CCIOT 2024), November 01–03, 2024, HaNoi, Vietnam.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3704304.3704312

*Judah Nava (judah.nava@go.minnesota.edu) and Hanku Lee (hanku.lee@mnstate.edu) are the corresponding authors.

CCIOT 2024, November 01–03, 2024, HaNoi, Vietnam

ACM ISBN 979-8-4007-1716-1/24/11
https://doi.org/10.1145/3704304.3704312

1 Introduction

The evolution of cloud and edge computing has significantly impacted the way data-intensive applications are developed and deployed. Function-as-a-Service (FaaS) [1] has emerged as a core component of this shift, offering a serverless computing model that enables functions to execute in response to specific events. With FaaS, developers benefit from scalability, high availability, and streamlined infrastructure management, allowing them to focus on application logic while resources are dynamically allocated based on demand.

Cloud computing has long been the dominant platform for FaaS, providing centralized resources, scalability, and simplified deployment through services like AWS Lambda [2] and Google Cloud Functions. However, this centralization introduces latency challenges as data often must travel considerable distances from edge devices to remote cloud data centers, limiting the responsiveness of real-time applications.

Edge computing [3] tackles these limitations by moving computation closer to the data source. By processing data locally at the network edge, edge computing reduces latency and improves real-time performance. This is particularly valuable for applications in domains such as the Internet of Things (IoT), smart cities, and autonomous systems, where immediate data processing is essential for effective operations.

As IoT devices generate increasing volumes of data, the constraints of cloud-based FaaS become more apparent, especially in latency-sensitive use cases. Industries like construction, which require real-time data analysis, demonstrate the need for localized edge solutions. To address this, we developed ComFaaS—a system that evaluates and compares FaaS implementations on both cloud and edge infrastructures.

The latest version of ComFaaS introduces several significant improvements over previous iterations [4, 5], enhancing its flexibility, versatility, security, and ease of use. The system has undergone a complete reimplementation, transitioning to a dynamic architecture that allows FaaS programs to be added and executed at runtime, without modifying the core system. This upgrade simplifies the integration of new features and boosts scalability. Additionally, ComFaaS now supports Python virtual environments, which isolate Python instances and dependencies, preventing conflicts between different FaaS applications. This isolation ensures that multiple applications can run concurrently without interference, addressing

both version control and security concerns. These enhancements make ComFaaS a more versatile, scalable, and secure platform for deploying FaaS in both cloud and edge environments.

This paper marks the official release of ComFaaS version 1.0.0, a significant achievement in the integration of edge computing with Function-as-a-Service (FaaS). This release introduces essential bug fixes, improved stability, and optimized performance, making ComFaaS a more reliable and efficient solution for deployment in production environments. With this version, ComFaaS becomes a robust, scalable system designed for seamless use in real-world edge computing scenarios. The package is available for download on the Parallel Solvit website (https://parallelsolvit.com/research-and-publications/) [6], with an alternative option provided on GitHub site (https://github.com/judahn02/ComFaaS) [7].

This paper evaluates the effectiveness of cloud and edge computing for FaaS applications, offering guidance on when to leverage each platform. The remainder of the paper is structured as follows: Section 2 explores related work, Section 3 details the ComFaaS architecture and setup, Section 4 presents benchmark results and analysis, and Section 5 concludes with a summary of findings.

2 Related Work

2.1 Status of Serverless Computing and Function-as-a-Service(FaaS) in Industry and Research

The Workshop on Serverless Computing (WoSC 2017) played a pivotal role in shaping the understanding of serverless computing during that period. Fox's report [8] not only provided a comprehensive overview of the state of serverless technologies at the time but also introduced a clear framework for key concepts such as Function-as-a-Service (FaaS), edge computing, and event-driven computing. By thoroughly detailing these concepts, the workshop offered valuable insights into how these technologies were evolving and their potential applications in modern computing environments. For the development of this project, the WoSC report was an indispensable reference, helping to inform decisions and guiding the technical approach. Its relevance to serverless computing made it a foundational resource that contributed significantly to the direction and structure of the work undertaken.

2.2 FaaS execution models for edge applications

The study of FaaS execution models for edge applications [5] emphasizes the importance of utilizing advanced implementations of FaaS services to ensure that the research remains relevant and impactful for the reader. The work presented in [9] offers valuable insights into multi-edge environments, demonstrating how edge nodes can communicate and efficiently distribute and manage functions. In particular, the paper explores three specific models—PureFaaS, StateProp, and StateLocal—that provide distinct methods for implementing FaaS services across multiple edge nodes. While these models offer significant contributions, our research shifts focus to investigating the interaction between cloud and edge environments over varying physical distances. We specifically analyze how these spatial factors influence performance from the perspective of edge users, adding a unique dimension to the ongoing study of FaaS in edge computing.

2.3 Balancing local vs. remote state allocation for micro-services in the cloud–edge continuum

The research presented in [10] centers on balancing local versus remote state allocation for microservices within the cloud-edge continuum, specifically focusing on the implementation of Platform-as-a-Service (PaaS) and Function-as-a-Service (FaaS) in edge computing environments. Given the limited computational resources at the edge, proper management is essential to avoid congestion. This study examines the distinct characteristics of processes in both PaaS and FaaS models, highlighting their behaviors in edge computing scenarios. Additionally, the paper explores the flexibility of dynamically switching applications between platforms during runtime. The findings conclude by identifying two key parameters that optimize the provisioning of PaaS and FaaS applications, with careful consideration of cost efficiency.

2.4 P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in Fog and Edge computing

This research [11] introduces a software solution aimed at enabling fully decentralized scheduling and load balancing in Fog and Edge environments. This innovative framework reduces latency for end users by allowing edge networks to function more independently, minimizing the reliance on centralized control. As a result, FaaS services can be delivered more efficiently, especially in latency-sensitive applications. The insights from this research are highly relevant to our project, as it provides a practical approach to preserving the cloud-edge relationship while incorporating redundancy mechanisms to enhance system resilience. By adopting peer-to-peer strategies inspired by P2PFaaS, edge networks can not only improve their ability to balance workloads autonomously but also boost their capacity for self-support. This decentralized model is a significant step forward in enhancing the efficiency, scalability, and fault tolerance of edge computing systems, which is critical for managing the growing demands on these infrastructures.

2.5 Distributed Deep Learning Model with Edge Computing

This paper [12] investigates the use of Edge Artificial Intelligence (EAI) in video surveillance (VS) systems to tackle issues such as excessive network communication overhead. The authors introduce a Distributed Intelligent Video Surveillance (DIVS) system that utilizes a multi-layer edge computing architecture along with a distributed deep learning model. Similar to ComFaaS Distributed, both systems harness the power of edge computing and parallel processing. However, their focus areas differ: while ComFaaS is primarily concerned with enhancing efficiency in FaaS applications, the DIVS system prioritizes reducing communication overhead, enabling low-latency analysis, and addressing challenges related to uneven connections and varying computational capacities. Key

contributions of the DIVS system include developing an edge computing architecture, implementing parallel processing at both the task and model levels, introducing a synchronization method for updating model parameters, and proposing a dynamic data migration strategy to improve workload balancing across the network.

2.6 EDGELESS Project: On the Road to Serverless Edge AI

The EDGELESS project [13, 14] and ComFaaS both focus on serverless computing at the edge, but they take different approaches in their goals and implementations. EDGELESS prioritizes optimizing resource efficiency across a variety of computing environments, from resource-limited edge devices to cloud platforms. By leveraging AI and machine learning, the project automates deployment and configuration, aiming to build a horizontally scalable solution that fully utilizes heterogeneous edge resources while seamlessly integrating with the cloud. With an emphasis on sustainability, EDGELESS dynamically adjusts configurations to allocate resources for performance-tolerant applications, ensuring efficient use of available capacity. This collaborative effort, led by Worldline (Spain) and involving 12 partners from six European countries, spans 36 months. Notably, EDGELESS focuses on applying the serverless paradigm across the entire edge-cloud continuum, enabling effective horizontal pooling of resources across both edge nodes and cloud infrastructures.

2.7 mpiPython

mpiPython [15–17] is a Python module designed to enable parallel computing by integrating the Message Passing Interface (MPI) standard into Python programs. Officially available as a pip package, it acts as a lightweight wrapper around MPI functions written in C, built on the MPICH implementation of MPI. The architecture of mpiPython is structured in multiple layers, ensuring smooth interoperability between Python and MPI. Its API is designed with simplicity in mind, striking an ideal balance between functionality and ease of use. Unlike more complex libraries such as mpi4py, which can involve a steep learning curve and extensive boilerplate code, mpiPython streamlines the process by minimizing the need for explicit datatype declarations. This allows developers to concentrate on building efficient, maintainable parallel code without becoming entangled in MPI complexities. Additionally, mpiPython's installation process is extremely straightforward; with a single pip command, users can install the library effortlessly, bypassing the challenges of complex builds or dependencies. This simplicity makes high-performance parallel computing accessible to developers of all experience levels, aligning well with ComFaaS's goals of usability and efficiency.

3 THE COMFAAS

3.1 Motivation

In recent years, there has been a growing interest in exploring the potential of edge computing, particularly in the context of Function-as-a-Service (FaaS) applications. The research community has recognized that edge computing offers a powerful paradigm for optimizing flexible and scalable FaaS implementations that can significantly reduce latency and improve performance in time-sensitive applications [18].

The primary challenge in traditional cloud-based FaaS deployments stems from the inherent distance between data sources and centralized cloud data centers. In regions where cloud infrastructure is limited or geographically distant—such as the Midwest of the United States—latency and data transfer uncertainty become pressing concerns. For example, AWS's cloud availability zones are concentrated on the coasts, requiring data from users in places like Minnesota and North Dakota to travel substantial distances to reach data centers in Oregon or Northern Virginia. This leads to potential delays in processing and introduces uncertainty in time-sensitive data transfers, which can negatively impact the performance of FaaS applications that rely on quick, efficient computations.

Edge computing addresses this issue by bringing computation closer to the data source, minimizing the need for long-distance data transfers. This shift enables faster data processing and decision-making, especially for applications that require fast responses. In this context, **ComFaaS** was developed to support both cloud-based and edge-based FaaS deployments, offering a detailed comparison of their efficiencies in various scenarios.

The key distinction between ComFaaS and traditional cloud systems, as shown in Figure 1, lies in the proximity and distribution of data processing. While traditional cloud computing offers significant computational power, scalability, and centralized management, it often requires data to be transmitted over networks to remote data centers, leading to higher latency. In contrast, ComFaaS leverages edge computing to process data locally, near the source, which significantly reduces network latency and optimizes bandwidth usage. This localized processing allows for streamlined decision-making and rapid responses to local events, which are crucial in applications such as healthcare monitoring, autonomous vehicles, and IoT-based systems.

By evaluating and comparing both cloud and edge computing for FaaS implementations, this research seeks to provide insights into the respective strengths and limitations of each paradigm. The ultimate goal is to highlight the potential of edge computing to complement traditional cloud services, particularly in regions where latency and data transfer challenges are more pronounced. ComFaaS aims to offer a comprehensive analysis of how these two computing models perform in real-world applications, helping inform future decisions about the optimal deployment strategies for FaaS applications across various domains.

3.2 The ComFaaS Architecture

We present a comparative analysis of edge computing integrated with Function-as-a-Service (FaaS) through a system called **ComFaaS**. In the ComFaaS architecture, three key components work together: the Internet of Things (IoT) component, the edge component, and the cloud component. These elements collaborate to address the unique challenges of processing large volumes of data in event-driven cloud environments, utilizing FaaS for efficient task execution.

The IoT component consists of a network of interconnected physical devices and sensors positioned at the outermost layer of the network. These devices collect data from their surroundings

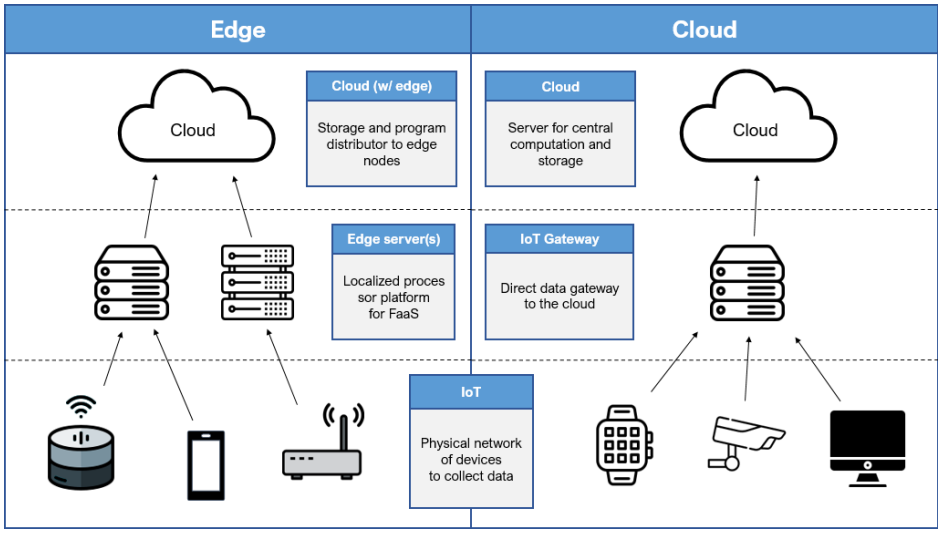

Figure 1: ComFaaS vs Traditional Cloud

in real-time, generating substantial volumes of information. IoT devices, including sensors, actuators, wearables, and other smart technologies, serve as the entry point into the edge computing ecosystem, transmitting data to the edge component for further processing.

The edge component acts as a bridge between IoT devices and cloud infrastructures. Typically consisting of edge servers or gateways, it is deployed close to the IoT devices, often at the network edge or on-site. The edge component is responsible for essential tasks such as data filtering, preprocessing, and local analysis. By processing data closer to its source, the edge component minimizes latency and optimizes bandwidth usage. It also enables real-time decision-making, allowing immediate responses to triggers without depending on cloud processing. This proximity and responsiveness make the edge component a natural fit for running FaaS services, providing users with faster results and reducing network congestion.

The cloud component represents centralized cloud infrastructure housed in remote data centers. This component is tasked with complex data processing, long-term storage, backups, advanced analytics, and other data-heavy operations. It provides scalability, global access, and extensive computing resources, supporting applications that require significant computational power. The cloud component also enables centralized management, data sharing, and collaboration across geographically distributed edge devices [19].

Together, the IoT, edge, and cloud components form a distributed architecture that facilitates efficient and scalable data processing in IoT environments. The IoT component gathers data from the physical world, the edge component processes and analyzes data locally for quick decision-making, and the cloud component handles more complex, resource-intensive tasks. This layered approach reduces network latency and enhances the overall performance and responsiveness of IoT systems.

ComFaaS is designed to automate the scheduling of event-driven FaaS processes and streamline data transfers between edge devices and the cloud, eliminating the need for manual data management in the cloud. The system empowers the edge component with greater control and flexibility, enabling it to execute FaaS services and respond to real-time edge triggers effectively.

As a comprehensive framework that includes both cloud and edge components, ComFaaS ensures seamless interaction between the two. The cloud manages system tests, ensuring compatibility across multiple operating systems, while efficient data management ensures smooth backups of edge device data to the cloud without compromising FaaS performance. To reduce latency, ComFaaS employs a dedicated Application Programming Interface (API) that streamlines communication between the cloud and edge components, optimizing data transfer and coordination. Through this integration, ComFaaS significantly enhances the performance of FaaS in cloud-edge environments, making a valuable contribution to the advancement of serverless and edge computing systems. The architecture of the ComFaaS system is illustrated in Figure 1.

3.3 The Implementation

While designed to leverage the potential of edge computing for Function-as-a-Service (FaaS) applications across multiple programming languages, ComFaaS is itself fully implemented in Java. In previous versions, the Java classes for the cloud and edge components followed a server-client model, primarily because much of the testing was initiated by the edge classes. The core operations of ComFaaS housed the communication protocols and APIs required to run Java, Python, C, and various MPI-based programs (Java + MPI, Python + MPI, and C + MPI), either sequentially or with MPICH and OpenMPI support. However, this setup was rudimentary, heavily dependent on interactions with the bash shell, and lacked flexibility.

In the latest version of ComFaaS, the implementation has been restructured to improve flexibility, scalability, and support for future enhancements. The system is now divided into a greater number of

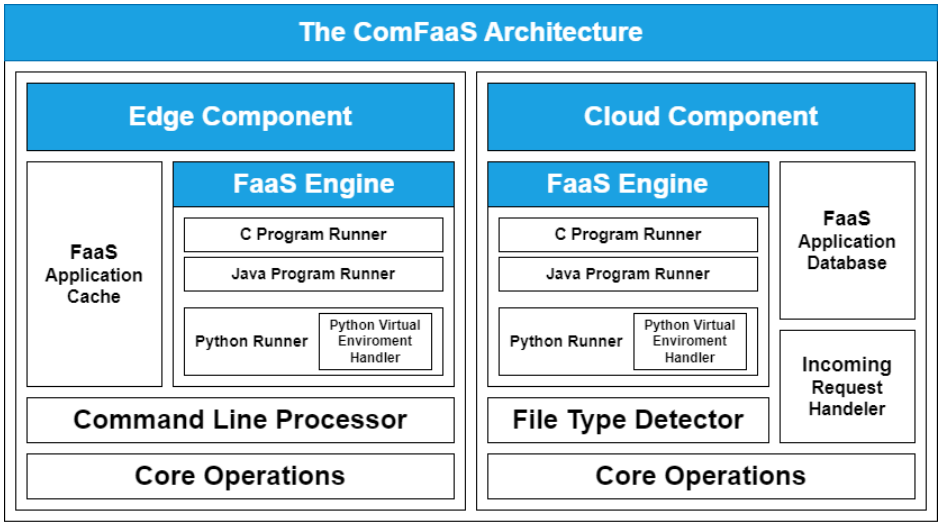

Figure 2: The ComFaaS System Architecture

Java classes, each with a specific and focused responsibility. Dedicated classes handle the execution of different types of programs, paving the way for easier expansion as new languages and frameworks are added. Additionally, terminal interactions have been streamlined to align with standard bash practices, and shell scripts have been introduced to simplify the initial setup of ComFaaS. These changes make ComFaaS more modular and maintainable, ensuring it can continue to evolve and meet the demands of a growing ecosystem of FaaS applications.

3.4 New Features

As part of the official release of ComFaaS, several critical improvements were made to enhance its flexibility, scalability, security, and ease of use, particularly in its support for Python virtual environments. A Python virtual environment is an isolated instance of Python with its own interpreter and package dependencies. This isolation is achieved by creating a separate Python instance with a dedicated filesystem, ensuring that each environment stores its own copies of the necessary dependencies for a given application. This is especially crucial for Python-based FaaS applications on ComFaaS, as it prevents conflicts between dependencies and enhances security by keeping environments separate from each other.

For example, if Program A requires mpiPython 1.0.14 and Program B requires mpiPython 2.0.1, without virtual environments, only one program could run at a time, with the other needing to wait until mpiPython is reconfigured to the appropriate version. Virtual environments solve this problem by allowing both programs to run concurrently without interference. Moreover, this isolation also addresses security concerns.

Modern Linux distributions restrict global **pip** installations to avoid system conflicts and enhance security. The recommended solution is to use Python virtual environments (**venv**), which allow local package management in isolated environments without needing admin privileges. Unlike global installations, which risk version conflicts and security vulnerabilities, virtual environments provide developers with full control over dependencies, ensuring consistent, conflict-free project setups.

This update has also streamlined the testing process. Previously, FaaS applications needed to be hardcoded into ComFaaS, but now they can be added and executed dynamically from the Edge without modifying the core system. This new capability allows for greater flexibility and scalability, as FaaS applications can be installed and run on demand during runtime. It simplifies the integration of new features as well as updates without requiring a complete system recompilation, enabling users to maintain, customize, and adapt ComFaaS more efficiently to various tasks and/or environments. These enhancements make ComFaaS a more versatile, scalable, and secure platform for edge computing and FaaS deployments.

4 Benchmark

4.1 Hardware Setup

The edge component used in this research is an Acer Aspire A315-41, featuring 8GB of RAM and powered by an AMD Ryzen 5 2500U quad-core processor with integrated Vega 8 graphics. It operates with network speeds below 100 Mbps for both download and upload, making it an ideal candidate for evaluating edge-based computations under typical commercial internet conditions. Running Ubuntu 22.04, this device is equipped with OpenJDK 21, Python 3.10, and GCC 11, providing a flexible environment for developing and testing Java, Python, C/C++, and mpiPython workloads. This configuration allows us to assess edge computing performance in real-world scenarios where limited resources and bandwidth are key factors.

The cloud component used in this research is a Dell PowerEdge 710, outfitted with dual Intel Xeon X5570 quad-core processors and 32GB of RAM. Its network connection supports download speeds below 800 Mbps and upload speeds under 300 Mbps, representing a higher-end cloud environment. The cloud runs a Debian 12 system containerized via LXC, providing an isolated, controlled testing environment. With OpenJDK 21, Python 3.11, and GCC 12 installed,

Table 1: mpiPython Pi Reduce algorithm with 1MB data transfer

Process	2	4	8	16
Chicago, IL	37.448	**26.257**	20.769	**20.149**
L.A., CA	27.469	16.278	10.79	10.17
New York, NY	36.926	25.735	20.247	19.627
Miami, FL	33.782	22.591	17.103	16.483
Edge	24.454	**14.836**	N/A	N/A

Table 2: mpiPython Pi Reduce algorithm with 500MB data transfer

Process	2	4	8	16
Chicago, IL	65.094	53.903	48.415	47.795
L.A., CA	89.043	**77.852**	72.364	**71.744**
New York, NY	79.717	68.526	63.038	62.418
Miami, FL	70.852	59.661	54.173	53.553
Edge	47.792	**38.174**	N/A	N/A

the cloud offers a consistent platform for software development and testing of Java, Python, C/C++, and mpiPython workloads.

4.2 The Benchmark Test

To thoroughly test ComFaaS, we need to evaluate a scenario where IoT devices communicate with both an edge component and cloud components across four different locations. The IoT devices will be in Moorhead, MN, and the edge component in Fargo, ND. The cloud components are in Chicago, IL (600+ miles from Fargo); New York, NY (1,400+ miles from Fargo); Los Angeles, CA (1,700+ miles from Fargo); and Miami, FL (2,000+ miles from Fargo). The test utilizes a Python collective communication library called mpiPython to enable parallel execution on both the Edge and the Cloud. It is important to note that the Edge is limited to a maximum of 4 processes per job, while the Cloud can scale up to 16 processes for a single job, providing greater capacity for resource-intensive tasks.

The test used in the four different scenarios involves a Montecarlo Pi-based computational algorithm that processes data to simulate a typical FaaS application performing operations on a dataset. The benchmarks focus on evaluating both the stability of the connection across varying distances and the efficiency of running a FaaS application on cloud infrastructure compared to running it on the edge.

4.3 Benchmark Analysis

Table 1 and 2 present the performance of the Edge and Cloud computing across different cloud locations. Overall, the edge component outperforms the cloud component, even when the cloud component utilizes more processes for computation. In parallel computing, more processes generally result in faster task completion, and this trend is reflected in both tables. However, the performance of the cloud component is hindered due to the need to transfer data to the cloud for computation, which offsets the advantage of having more processes. A unique exception to this is seen in the first table, where the cloud in Los Angeles outperformed the edge. This anomaly could be attributed to factors like better network infrastructure, reduced congestion, or more efficient resource allocation at the Los Angeles cloud data center, highlighting the variability that can occur in cloud performance based on location-specific conditions.

In Table 1, the Monte Carlo Pi-based algorithm involves a 1 MB data transfer. In this benchmark, the 4-node edge outperformed the 4-node cloud component in Chicago, IL by 43%. Additionally, when comparing the 4-node edge component to the 16-node cloud, the edge shows a 26% improvement. Table 2 represents the same test program as Table 1, with a higher data transfer load of 500 MB. The 4-node edge showed a 51% faster time compared to the 4-node cloud component in Los Angeles, CA. Moreover, the 4-node edge marked an improvement of 47% when compared to the 16-node cloud.

The benchmark analysis presented in Tables 1 and 2 demonstrates the significant advantages of edge computing over cloud-based solutions, especially in the context of ComFaaS. Edge computing consistently outperforms cloud alternatives, even when the cloud uses more computational processes, as shown in both tables. However, the exception in Los Angeles, CA (Table 1) illustrates that while cloud computing can sometimes surpass edge performance, the latency involved in transferring data to the cloud often diminishes the benefits of its additional processing power.

Despite edge computing's strengths, there are scenarios where cloud computing is more effective, particularly when large-scale computational power and extensive resources are needed. ComFaaS addresses this by incorporating dynamic load balancing, intelligently distributing workloads between edge and cloud environments. This hybrid approach ensures that the most efficient solution is chosen for each task, allowing edge and cloud computing to complement each other. By leveraging the strengths of both systems, ComFaaS provides a flexible, reliable, and scalable platform for FaaS applications, delivering optimal performance based on the specific needs of each application.

5 Conclusion

The fully functional release of ComFaaS is now available, offering substantial improvements in flexibility, scalability, security, and ease of use, with a strong focus on supporting Python virtual environments. These environments allow isolated instances of Python, preventing dependency conflicts and improving security for Python-based FaaS applications. By enabling concurrent execution of programs with different dependencies, virtual environments solve issues that previously required manual reconfiguration. Additionally, the update introduces dynamic deployment of FaaS applications from the Edge without altering the core system, streamlining updates and expanding scalability. These improvements make ComFaaS a more adaptable, efficient, and secure platform for edge computing and FaaS solutions.

The benchmark analysis emphasizes the clear advantages of edge computing over cloud-based solutions, particularly in the context of ComFaaS. Edge computing consistently delivers superior performance, even when the cloud utilizes more computational resources. However, there are situations where cloud computing may offer better results, especially when large-scale processing power or significant resources are needed. ComFaaS addresses this by implementing dynamic load balancing, intelligently distributing workloads between edge and cloud environments. This hybrid approach ensures that each task benefits from the most efficient computing option available, with edge and cloud systems complementing one another. As a result, ComFaaS provides a scalable and reliable platform for FaaS applications, maximizing performance based on the specific demands of each application.

This research sheds light on the strengths and limitations of both cloud and edge computing in FaaS applications in the context of ComFaaS. Edge computing proves highly effective in scenarios where low latency, real-time processing, and efficient bandwidth usage are essential. In contrast, cloud computing excels in cases that require vast scalability, global access, and flexible resource allocation, making it the preferred choice for tasks that demand large-scale infrastructure.

References

[1] FaaS: https://www.ibm.com/topics/faas

[2] Amazon, “AWS Lambda – Serverless Compute - Amazon Web Services,” Amazon Web Services, Inc., 2019. https://aws.amazon.com/lambda/ (accessed Jul. 15, 2023).

[3] Edge Computing: https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/

[4] Jaden Jinu Lee, Judah J. Nava, and Hanku Lee. 2023. ComFaaS: Comparative Analysis of Edge Computing with Function-as-a-Service. In Proceedings of the 2023 8th International Conference on Cloud Computing and Internet of Things (CCIOT ’23). Association for Computing Machinery, New York, NY, USA, 84–90. https://doi.org/10.1145/3627345.3627358

[5] J. J. Lee, J. Nava and H. Lee, ”ComFaaS Distributed: Edge Computing with Function-as-a-Service in Parallel Cloud Environments,” 2024 7th International Conference on Information and Computer Technologies (ICICT), Honolulu, HI, USA, 2024, pp. 133-138, doi: 10.1109/ICICT62343.2024.00027.

[6] The Parallel Solvit website: https://parallelsolvit.com/research-and-publications/

[7] The GitHub site for ComFaaS: https://github.com/judahn02/ComFaaS

[8] G. C. Fox, Vatche Ishakian, V. Muthusamy, and A. Slominski, “Status of Serverless Computing and Function-as-a-Service(FaaS) in Industry and Research,” Aug. 2017, doi: https://doi.org/10.13140/rg.2.2.15007.87206.

[9] C. Cicconetti, M. Conti, and A. Passarella, “FaaS execution models for edge applications,” Pervasive and Mobile Computing, vol. 86, p. 101689, Oct. 2022, doi: https://doi.org/10.1016/j.pmcj.2022.101689.

[10] C. Puliafito, C. Cicconetti, M. Conti, Enzo Mingozzi, and A. Passarella, “Balancing local vs. remote state allocation for micro-services in the cloud–edge continuum,” Pervasive and Mobile Computing, vol. 93, p. 101808, 2023, doi: https://doi.org/10.1016/j.pmcj.2023.101808.

[11] Gabriele Proietti Mattia and R. Beraldi, “P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in Fog and Edge computing,” SoftwareX, vol. 21, p. 101290, 2023, doi: https://doi.org/10.1016/j.softx.2022.101290

[12] J. Chen, K. Li, Q. Deng, K. Li and P. S. Yu, ”Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing,” in IEEE Transactions on Industrial Informatics, doi: 10.1109/TII.2019.2909473.

[13] Claudio Cicconetti, Emanuele Carlini, and Antonio Paradell. 2023. EDGELESS Project: On the Road to Serverless Edge AI. In Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge (FRAME ’23). Association for Computing Machinery, New York, NY, USA, 41–43. https://doi.org/10.1145/3589010.3594890

[14] https://edgeless-project.eu/

[15] H. Park, J. DeNio, J. Choi and H. Lee, ”mpiPython: A Robust Python MPI Binding,” 2020 3rd International Conference on Information and Computer Technologies (ICICT), San Jose, CA, USA, 2020, pp. 96-101, doi: 10.1109/ICICT50521.2020.00023.

[16] J. Nava and H. Lee, ”mpiPython: Prospects for Node Performance,” 2023 6th International Conference on Information and Computer Technologies (ICICT), Raleigh, NC, USA, 2023, pp. 182-187, doi: 10.1109/ICICT58900.2023.00038.

[17] J. Nava, J. J. Lee and H. Lee, ”mpiPython: Extensions of Collective Operations,” 2024 7th International Conference on Information and Computer Technologies (ICICT), Honolulu, HI, USA, 2024, pp. 468-473, doi: 10.1109/ICICT62343.2024.00082.

[18] C. Cicconetti, M. Conti, and A. Passarella, “FaaS execution models for edge applications,” Pervasive and Mobile Computing, vol. 86, p. 101689, Oct. 2022, doi: https://doi.org/10.1016/j.pmcj.2022.101689.

[19] Y. Khalidi, “Microsoft partners with the industry to unlock new 5G scenarios with Azure Edge Zones | Azure Blog | Microsoft Azure,” Azure Blog, Mar. 31, 2020. https://azure.microsoft.com/en-us/blog/microsoft-partners-with-the-industry-to-unlock-new-5g-scenarios-with-azure-edge-zones/ (accessed Jul. 15, 2023).