IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

# Operating Systems for Serverless and Edge Computing: A Shift Towards Lightweight and Specialized OS Designs

**Dr. D. Thamaraiselvi[1] and Siri Chandana Kathyayani[2]**
[1] Assistant Professor, [2] Student, (Computer Science & Engineering),
Sri Chandrasekharendra Saraswathi Viswa Maha Vidyalaya, Kanchipuram

**Abstract**: *The advent of serverless computing and edge computing paradigms has caused a major shift in the challenges organizations face in operating systems (OS) design. Existing operating systems (OS), monolithic Linux and Windows, are overkill that is slow and resource-heavy in the context of distributed, cloud, and edge environments. This article analyzes lightweight OS to address the problem. We specifically examine the architecture and implementation of unikernels, library OSs, and micro-OS designs (i.e. MirageOS, IncludeOS, and OSv). We also highlight the serverless computing and edge attributes of these OSs, namely the deployment speed, low latency, and small size. Finally, we look at the future of OS design characterized by security, scalability, and levels of automation*

**Keywords**: lightweight virtualization, IoT, embedded systems, FaaS, secure boot, fast startup, latency-sensitive applications, cloud-native workloads

## I. INTRODUCTION

With serverless and edge computing's recent rise, we have seen a change in how applications are deployed and managed. Serverless computing takes away the abstraction of managing infrastructure, so developers can concentrate on operating at the level of code for functions. Edge computing aims to perform computation as close to the data source as possible, like in IoT devices, smart sensors, and autonomous vehicles. However, the devices in edge computing environments often have constraints such as limited memory, storage, and CPU, but require low latencies in computational processing and real-time processing.

Operating systems (OS) like Linux or Windows were designed for general-purpose computing environments and do not offer the performance or resource utilization that serverless and edge computing need. In addition, operating systems often take unwanted resources to start up, or it will have longer boot times. Hence, there is an increasing need for operating systems (other than general-purpose OSs) designed for serverless and edge computing applications.

In this paper, we will examine lightweight, specialized operating systems, like unikernels, library operating systems, and micro-OS architectures, examined from a design and effect perspective.

## II. CHALLENGES IN TRADITIONAL OS ARCHITECTURES

Conventional operating systems are complex, multi-purpose systems that are intended to host a multitude of applications. In serverless and edge computing, conventional operating systems are confronted with:

1. **Resource Overhead:** Classic OSs are big codebases that use considerable memory and a CPU in order to execute, and thus use plenty of resources, which is not efficient on a resource limited device.
2. **Long Boot Time:** OSs such as Linux take seconds or minutes to boot, which is infeasible for applications with an urgent need for fast start times (e.g. serverless functions).
3. **Monolithic Design:** The majority of OSs are monolithic, as they are pre-bundled with all types of features into the kernel. It may result in an unnecessarily huge attack surface and further vulnerability.
4. **Latency:** Anything that introduces overhead on making things on-time in any real-time processing situation is bad, particularly for critically timed applications like in autonomous systems or real-time analytics.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

To solve these issues, we have started to witness operating systems that are made to be light, boot fast, and optimize for a particular workload.

## III. LIGHTWEIGHT OS DESIGNS

Lightweight operating systems (OSs) have been created to address the challenges faced by modern distributed systems. Lightweight OSs use minimal constructs and only enough OS functionality to deploy applications in cloud and edge native environments.

### 3.1 Unikernels

Unikernels are single-purpose lightweight OSs that compile application code with just the functionally relevant OS constructs. The final application is a single binary file. The binary includes the application code, kernel, and libraries. By conceptually imprisoning the code within a unikernel, binaries eliminate much of the OS overhead normal to traditional OSs. In addition to reducing traditional overhead, unikernels start os processes more quickly and consume in system resources less memory. The final applications are also safer to exploit by reducing the attack surface.

Notable unikernel projects:

MirageOS is a unikernel built for the cloud. Written in OCaml, MirageOS runs directly on the hypervisor without needing a full OS layer. This makes application unique for lightweight, stateless serverless functions.

IncludeOS is a C++ unicaernel optimized to run on virtual machines. IncludeOS is relatively minimalistic, which minimizes system services to run cloud-native applications to improve system efficiency while also minimizing its attack surface.

OSv is an OS designed for cloud workloads we have similar operating system configurations to allow applications to run directly on virtual machines while minimizing overhead from the underlying OS. Its the OS has a minimal kernel with enhanced performance compared to other virtualized environments.

### 3.2 Library Operating Systems

Library OSs allow operating system services to be provided as libraries that can be linked into applications. This allows for flexibility about which parts of the OS are used and permits applications to utilize only the parts of the OS they need to run.

Example:

Graphene Library OS: A Library OS that allows Linux applications to run in isolated environments. It is intended for applications that need to run in secure / isolated / sandboxed environments but want to be compatible with existing software.

### 3.3 Micro-OS Architecture

Micro-OS designs focus on splitting the operating system into small module units that can be customized for specific situations. These designs allow developers to build very specialized operating systems which only include the parts of the OS required to perform tasks such as networking, storage or real-time processing.

Example:

RIOT OS: An open-source, micro-OS meant for IoT devices. Supports low-power devices, real-time applications and a variety of hardware.

Zephyr OS: A Real-Time Operating System which targets connected, resource constrained devices: typically deployed in edge computing. It has a variety of hardware support and scalability with consideration to security in distributed systems.

## IV. EDGE-OPTIMIZED OPERATING SYSTEMS

Edge computing refers to data processing done in close proximity to the source of the data (i.e. at the sensor level, wearables, autonomous systems, etc.). Edge devices typically have very constrained power, bandwidth, and latency.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

Hence, edge operating systems need to be lightweight and have effective processes for real-time data processing.Beyond being lightweight, edge optimized operating systems have other important features as well.

1. Low Power Consumption: Edge devices are frequently powered using batteries or low-power processors. Therefore, power consumption is always a priority.
2. Fast Wake-Up: Devices in edge networks need to wake quickly from low power dormant states to process incoming data faster than real-time.
3. Real-Time Processing: Edge applications are targeted to run inside the constraints set by real-time expectations; therefore, deterministic behavior is anticipated. Real-time operating systems (RTOS) are typically assumed in edge applications.

Some examples of edge optimized operating systems include:

- FreeRTOS. FreeRTOS is a real-time operating system for embedded systems and Internet of Things (IoT) devices. FreeRTOS is available for many different microcontroller architectures, making it particularly teacher and student friendly as a development platform for edge computing applications.
- RIOT OS. RIOT OS is an open-source operating system optimized for use cases with low-power, resource-constrained devices. RIOT OS has support for real-time operations if desired by the programmer.
- Mbed OS. Mbed OS is designed for IoT devices with an emphasis around fast execution and secure communications.

## V. SECURITY CONSIDERATIONS

While lightweight operating systems such as unikernels and micro-OSs can offer tremendous performance improvements, the risk for security issues are still a complication. The lightweight nature of these systems can make vulnerabilities less obvious, and adding security mechanisms needs to happen for every layer.

Some key security factors which developers should consider are:

- Isolation: Isolation means that applications running on the same hardware should not be able to disrupt or leak data.
- Secure Boot: The OS and applications should be verified to be authentic before the boot process.
- Data Integrity: Provide means that data being used and stored on edge devices is protected, and can't be manipulated or corrupted.

Technologies such as hardware-backed, hardware-assisted secrets (Intel SGX, ARM TrustZone) or software-based isolation (sandboxing) can be critical for ensuring the safety of applications running in edge and serverless contexts.

## VI. FUTURE TRENDS IN OS DESIGN FOR SERVERLESS AND EDGE COMPUTING

Both serverless and edge computing operating systems are poised to see ongoing innovation in the future. Some of the trends to consider are:

1. **Composable Operating Systems:** Future operating systems may enable dynamic composition of components to allow developers to only include the modules they need based on the specification of the workload.
2. **AI-Enabled Dynamism:** Operational systems may leverage AI (machine learning-based) techniques to automatically optimize resource allocation in terms of minimizing energy utilization and enhancing performance based on usage patterns.
3. **Hybrid Operating Systems:** Hybrid operating systems for serverless computing could also emerge that combine the benefits of unikernels (speed, isolation) with the advantages of containers (management ease).
4. **Decentralized Operating Systems:** Blockchain-based operating systems may provide decentralized access control features, ensuring that elements and services that are found in serverless and edge computing environments are adequately secured and treated transparently.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

## VII. CONCLUSION

The emergence of serverless and edge computing has created a need for lightweight operating systems (OSs) for specific use cases. While classic OSs generally rely on excess memory and unnecessary processes, unikernels, library operating systems (OSs), and micro-OSs offer major enhancements in performance, scalability, and security. These paradigms provide the operating systems necessary to leverage the modern programming model of resource-constrained (computationally-constrained, timing-constrained, etc.) and low-latency environments required by modern distributed systems. Much like computing will evolve, we expect the OS to evolve too, with performance, scalability, security, and automation defining the next generation of OS.

## REFERENCES

**[1].** Madhavapeddy, A., et al. (2013). *Unikernels: Library Operating Systems for the Cloud*. ACM SIGPLAN Notices, 48(4), 461–472.
**[2].** Williams, R., & Johnston, M. (2019). *IncludeOS: A Minimal, Tailored Operating System for Cloud and Edge Environments*. IEEE Internet Computing, 23(2), 58–66.
**[3].** Levi, D., & Zilberman, N. (2020). *Fast and Secure Serverless Computing Using Unikernels*. USENIX Annual Technical Conference.
**[4].** Kivity, A., et al. (2014). *OSv—Optimizing the Operating System for Virtual Machines*. USENIX Annual Technical Conference.
**[5].** Morabito, R., & Beijar, N. (2016). *Hypervisors vs. Lightweight Virtualization: A Performance Comparison*. IEEE Conference on Cloud Engineering.
**[6].** Harter, T., et al. (2014). *Slacker: Fast Distribution with Lazy Docker Containers*. USENIX Symposium on Operating Systems Design and Implementation.
**[7].** Zephyr Project. https://www.zephyrproject.org
**[8].** RIOT OS. https://www.riot-os.org
**[9].** FreeRTOS. https://www.freertos.org
**[10].** Mbed OS. https://os.mbed.com
**[11].** Graphene Project. https://grapheneproject.io
**[12].** VMware Research. (2019). *Edge Computing: Vision and Challenges*. VMware White Paper.
**[13].** Amazon Web Services. (2023). *AWS Lambda and the Evolution of Serverless*. AWS Technical Guide.
**[14].** Microsoft Research. (2021). *Drawbridge: A Library OS for Windows Applications*. Microsoft Technical Report.
**[15].** Xen Project. https://xenproject.org