Resource Management in Serverless Computing: A Review and Perspective on Educational Integration

Olusegun Victor Adeola
Department of Information Systems and Cyber Security
University of Texas at San Antonio
San Antonio, USA
olusegun.adeola@my.utsa.edu

***Abstract*—This report reviews "A Holistic View on Resource Management in Serverless Computing Environments" by Mampage, Karunasekera, and Buyya (2021), a recent paper on resource management in serverless computing, with a focus on workload modeling, scheduling, and scaling. The paper proposes a detailed taxonomy and identifies research challenges across commercial and hybrid platforms. Key ideas are connected to core cloud computing topics, including orchestration, elasticity, and cost efficiency. The discussion also explores how serverless concepts could enhance learning in a university cloud computing course.**

Keywords—Serverless, computing, resource management, cloud computing, education

I. INTRODUCTION

Serverless computing, often referred to as Function-as-a-Service (FaaS), is a modern cloud architecture that allows developers to run code without managing servers or underlying infrastructure. Code is triggered by events, such as HTTP requests, database updates, or file uploads, and executed in short bursts on demand. This design simplifies deployment, accelerates development, and allows for highly granular billing. Instead of provisioning virtual machines or containers, developers simply focus on writing functions, leaving scalability, fault tolerance, and resource allocation to the cloud provider.

However, this convenience introduces complexity behind the scenes. Serverless platforms must dynamically manage resources, ensure fast startup times, isolate tenants, and optimize performance at scale. These challenges require sophisticated strategies for workload modeling, function scheduling, and automatic scaling.

The paper by Mampage et al. (2021) addresses these topics in depth, proposing a taxonomy that organizes existing solutions and research directions. This report provides a structured summary of the paper, explores how its ideas align with core topics in cloud computing education, and proposes ways to integrate serverless principles into coursework for practical understanding.

II. SUMMARY OF THE PAPER

The paper by Mampage, Karunasekera, and Buyya (2021) presents a comprehensive overview of resource management in serverless computing environments. It introduces a three-part taxonomy that focuses on application workload modeling, resource scheduling, and scaling, all of which are essential for efficiently managing serverless platforms, such as AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions. The authors also discuss related aspects such as isolation mechanisms, deployment environments (e.g., cloud vs. edge), and pricing models..

A. Application Workload Modeling

In serverless computing, users do not manage servers directly, so the system must autonomously learn and anticipate workload characteristics. The paper emphasizes that understanding function invocation patterns, including request frequency, concurrency bursts, and resource requirements, is crucial for minimizing cold starts and enhancing performance. Techniques such as historical log analysis, time-series forecasting, and lightweight machine learning models are suggested to help predict when and how functions will be triggered. Accurate modeling enables providers to “pre-warm” containers, ensuring faster execution.

B. Resource Scheduling

Resource scheduling is more challenging in serverless systems than in traditional cloud platforms. Since functions are typically short-lived and stateless, they must be assigned to nodes in real time, often with millisecond-level constraints. The authors categorize existing strategies into three main approaches: rule-based, load-based, and learning-based. They also consider function placement in edge-cloud hybrid environments, where latency sensitivity or data locality might influence scheduling decisions. Additionally, the paper explores container reuse and microVM usage (e.g., Firecracker by AWS) to strike a balance between performance and isolation.

C. Scaling

While serverless is known for its auto-scaling capabilities, the authors argue that current implementations are not always

efficient. Many systems rely on simple reactive scaling, which can lead to over- or under-provisioning. The paper reviews more innovative strategies, such as queue-aware scaling, proactive instance allocation, and function grouping (where similar functions share containers). These methods can reduce cold starts and increase cost-efficiency.

An extensive literature review backs the taxonomy and includes examples from leading platforms. For instance, AWS Lambda supports limited container reuse and scales per invocation, while Azure Functions uses a combination of event-driven scaling and dynamic VM pools. The authors highlight challenges like energy-aware scheduling, edge integration, and flexible pricing models that account for cold starts and service guarantees. Overall, the paper not only categorizes current solutions but also lays a foundation for future innovation by highlighting gaps in serverless design and performance optimization. These ideas echo earlier observations by Jonas et al. [4], who emphasized the potential of serverless platforms to simplify cloud programming by abstracting infrastructure concerns.

The core architectural components of a serverless platform, as described by Mampage et al., are summarized in Fig. 1. These include the API gateway, function workers, a resource scheduler, and monitoring systems that manage the execution environment.

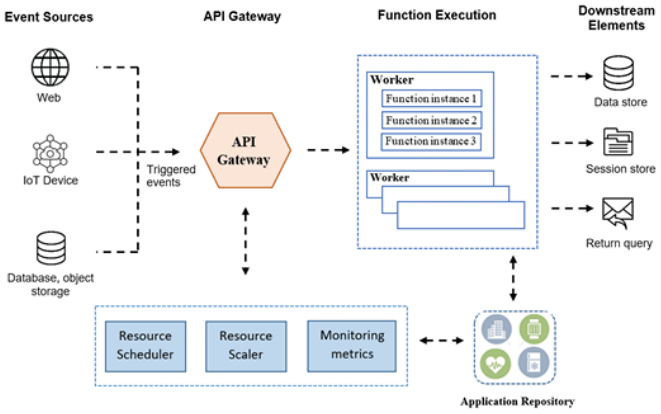

Fig. 1. Serverless architecture and control components. Adapted from Mampage et al. [1].

III. DISCUSSION

Although serverless computing was not the primary focus of our cloud computing course, many of the concepts discussed in the paper by Mampage et al. (2021) strongly connect with core topics we explored, such as containerization, virtualization, and distributed orchestration. Throughout the course, we utilized tools such as Docker, OpenStack, and Kubernetes to learn how cloud applications are deployed, scaled, and monitored. These platforms require explicit configuration of compute instances, resource quotas, and scaling policies. Serverless platforms abstract these concerns, offering automatic provisioning and on-demand scaling. However, this abstraction comes at the cost of more complex internal resource management, which is the focus of the reviewed paper.

The paper's taxonomy provides a helpful lens for revisiting the systems we worked with. For example, OpenStack's Nova scheduler is responsible for deciding where virtual machines are placed within the cluster. This decision typically considers compute capacity, availability zones, and load distribution policies. Serverless function placement, while hidden from the user, shares the same underlying objective of matching application needs with infrastructure capacity. However, serverless placement must operate under more severe latency and concurrency constraints due to the real-time nature of function triggers.

Kubernetes, which we use for container orchestration, provides autoscaling based on resource utilization metrics, such as CPU and memory. This aligns with the scaling discussions in the paper, which compare simple metric-based autoscaling with more advanced techniques, such as proactive and queue-aware scaling. The idea of container reuse also parallels Kubernetes' pod lifecycle management, though serverless containers tend to be shorter-lived and less predictable.

An area where serverless computing diverges sharply from our coursework is pricing. In most labs, we calculated costs based on instance runtime and storage duration. Serverless pricing introduces finer granularity, typically based on per-invocation duration and memory allocation. This shift encourages developers to write lightweight, efficient code that executes quickly and consumes fewer resources. In practice, this could lead to architectural decisions, such as decomposing monolithic services into smaller functions, even if it increases complexity in deployment and monitoring.

To better align serverless computing with existing curriculum components, several additions could be made to the curriculum. A short lab could guide students through deploying a basic API or file processing service using AWS Lambda or Google Cloud Functions. Students would then compare this experience with deploying the same service on a Docker container managed via Kubernetes or OpenStack. By measuring factors such as startup latency, cost per request, and scalability under load, students would gain a firsthand understanding of the strengths and limitations of each model. **These differences are summarized in Table I**, which compares key features of container-based and serverless deployment models.

By adding this lab and reflective comparison, the course can introduce students to real-world trade-offs between flexibility and automation, control and simplicity, and cost versus performance. It would also reinforce cloud-native principles that extend beyond traditional virtual machine provisioning, aligning with how many organizations now build production systems.

Integrating serverless into the curriculum would not displace existing tools and topics. Instead, it would extend them by offering a modern perspective on how applications can scale with minimal operational burden. Given the increasing adoption of serverless computing in the industry, providing students with exposure to these models is essential for preparing them to build responsive and scalable cloud solutions.

TABLE I. Comparison of Container-Based and Serverless Deployment Models

Feature	**Docker on OpenStack/K8s**	**Serverless (e.g., AWS Lambda)**
PROVISIONING	Manual or semi-automatic	Fully automated
SCALING GRANULARITY	Pod or VM level	Per function call
COLD START LATENCY	Minimal after boot	Varies, may introduce delay
BILLING STRATEGY	Per hour or second usage	Per invocation and duration
DEVELOPER CONTROL	High (OS, runtime, resources)	Limited (code and config only)
COMPLEXITY OF SETUP	Higher	Lower
ISOLATION	Substantial (VM/container)	Varies, may use microVMs or containers
USE CASE FIT	Long-running or stateful apps	Short-lived, event-driven apps

IV. FUTURE WORK

The paper by Mampage et al. (2021) outlines several promising directions for future research in serverless computing, especially in how platforms manage resources under diverse and dynamic workloads. One of the key areas highlighted is the use of machine learning for predictive scheduling. Rather than relying on simple thresholds or reactive policies, platforms could leverage models trained on historical data to forecast traffic bursts, select optimal execution nodes, and minimize cold start delays.

Another challenge lies in extending serverless platforms to edge computing environments. While the cloud offers elasticity and global reach, edge devices provide lower latency and better data locality. Coordinating function placement across cloud-edge hybrid systems, while ensuring isolation and consistency, is still an open problem.

Additionally, the authors call for energy-aware resource management, where functions are scheduled to nodes with lower carbon footprints or executed during times of renewable energy availability. As environmental concerns shape infrastructure planning, integrating sustainability into platform design becomes increasingly important.

These open challenges underscore the ongoing evolution of serverless systems, presenting numerous opportunities for innovation in both academia and industry.

V. CONCLUSION

Serverless computing has emerged as a powerful abstraction in the evolution of cloud platforms, enabling developers to build and deploy event-driven applications with minimal operational overhead. By shifting responsibility for provisioning, scaling, and fault tolerance to the cloud provider, serverless platforms allow faster development cycles and cost-efficient execution. However, this simplicity on the surface masks a complex system beneath, where real-time resource management must balance performance, scalability, and cost.

The paper by Mampage et al. (2021) provides a well-organized taxonomy that explains how current platforms handle these challenges. By analyzing workload modeling, scheduling, and scaling, the authors give a clear picture of both what serverless systems do today and where they must evolve. The discussion of deployment models, pricing structures, and platform constraints also reflects how design trade-offs shape user experience and provider efficiency.

This report has shown that many of the paper's insights align with core cloud computing topics such as orchestration, elasticity, and cost modeling. Integrating serverless computing into cloud computing education would not only modernize the curriculum but also prepare students for the fast-changing landscape of cloud-native development. As serverless continues to mature, it will play an increasingly central role in building responsive, scalable applications.

REFERENCES

[1] A. Mampage, S. Karunasekera, and R. Buyya, "A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions," *arXiv preprint arXiv:2105.11592*, 2021. [Online]. Available: https://arxiv.org/abs/2105.11592

[2] AWS Lambda Documentation, "How AWS Lambda Works," Amazon Web Services, 2023. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[3] Azure Functions Documentation, "Azure Functions Overview," Microsoft, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview

[4] J. Jonas, E. Schleier-Smith, V. Sreekanti, et al., "Cloud Programming Simplified: A Berkeley View on Serverless Computing," *UC Berkeley Technical Report*, 2019. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-3.pdf