Evaluating the Energy Efficiency of Serverless Computing for Small-Scale Web Apps

Pranuth H M
PG Student, Department of CSE
Rajeev Institute of Technology, Hassan
Hassan, India
pranuth.h.manjunath@gmail.com

Dr. Sharath M N
Associate Professor, Department of CSE (AI&ML) Rajeev Institute of Technology, Hassan
Hassan, India
sharathmn64@gmail.com

***Abstract*—Serverless computing has emerged as a transformative cloud paradigm by abstracting infrastructure management while promising scalability and cost-efficiency. However, its environmental impact, particularly for small-scale web applications, remains underexplored. This paper presents a comparative analysis of energy consumption between serverless computing models, specifically AWS Lambda, and traditional VM-based hosting on AWS EC2 for lightweight web apps. We present experimental data evaluating energy metrics over a 30-day period. Our findings demonstrate that serverless architectures can reduce idle-time energy consumption by up to 65% and overall energy usage by 28% in low-traffic scenarios. This study highlights serverless computing's potential as a green computing solution for sustainable software engineering.**

Keywords—serverless computing, AWS Lambda, energy efficiency, green computing, cloud sustainability, web applications.

I. INTRODUCTION

The increasing global emphasis on sustainability has extended into computing, where energy efficiency plays a crucial role. Cloud platforms have revolutionized application deployment, yet traditional hosting methods often lead to inefficient energy utilization, especially during idle states or underutilized workloads. Serverless computing, or Function-as-a-Service (FaaS), offers a compelling alternative by executing code only in response to events, potentially leading to lower energy usage and operational overhead.

In addition to energy savings, serverless platforms offer benefits such as automatic scaling, reduced infrastructure maintenance, and cost alignment with actual usage. These features make serverless architectures particularly attractive for small to medium-scale applications where traffic is variable and unpredictable.

Despite its growing popularity, the **real-world energy efficiency of serverless computing remains underexplored**, especially in contrast to conventional virtual machine (VM)-based deployments. Most existing research tends to focus on large-scale enterprise systems, leaving a gap in understanding the energy dynamics of lightweight, low-traffic web applications—such as portfolio websites, internal dashboards, or educational tools.

In this paper, we investigate whether this model significantly enhances energy efficiency for small-scale web applications compared to traditional Virtual Machine (VM) hosting. This study addresses that gap by:

1. Quantifying idle vs peak energy usage for AWS EC2 and Lambda.
2. Estimating total energy consumption over 30 days under realistic traffic patterns.
3. Discussing performance, cost, and carbon trade-offs.

II. LITERATURE REVIEW

A comprehensive review of recent literature was conducted to understand current trends, limitations, and gaps in energy efficiency research within serverless computing. This literature survey highlights significant academic efforts in modeling, comparing, and analyzing cloud-based architectures, particularly focusing on environmental sustainability and resource optimization.

Serverless computing, also known as Function-as-a-Service (FaaS), has evolved into a widely adopted paradigm in cloud architecture due to its scalability and cost-efficiency. However, its environmental impact and energy efficiency have only recently gained serious attention within both academic and industrial contexts.

Sharma [1] critically examined the carbon inefficiencies of FaaS architectures, identifying that serverless functions can consume up to 15× more energy than conventional HTTP servers due to container-based virtualization and cold-start overheads. The study emphasized that the control plane complexity and short-lived, stateless functions pose unique energy profiling challenges, especially for real-time monitoring and energy attribution in distributed environments.

Akour and Alenezi [2] conducted a comprehensive systematic review and empirical survey, finding that serverless computing can reduce energy usage by up to 70% compared to traditional VM-hosted applications—primarily due to the reduction in idle resource consumption. Their study also highlighted gaps between industry implementation and academic benchmarks, pointing to challenges like cold-start latency, platform dependency, and heterogeneous workloads that affect real-world energy efficiency outcomes.

Chintha et al. [3] compared serverless and traditional cloud architectures using performance and cost metrics. Their results support the claim that serverless models outperform traditional setups in scaling speed and operational efficiency, but introduce variability in latency due to cold-starts. The authors proposed structured performance hypotheses that validate serverless efficiency through statistically significant empirical data.

McGrath and Brenner [4] explored the design and implementation of serverless architectures, revealing that the lightweight virtualization mechanisms, such as Firecracker microVMs, are increasingly used to improve energy and boot-time efficiency. However, they caution that increased abstraction layers may inadvertently increase total energy due to orchestration overheads.

Hassan et al. [5] conducted a survey of 275 papers on serverless computing and emphasized that sustainability and energy efficiency remain under-researched, despite growing interest. The paper advocates for more granular monitoring tools and standardized metrics to measure energy usage across different serverless platforms and workloads. It also highlighted a critical need for benchmarks specific to low-traffic, small-scale applications—a gap directly addressed in this research.

III. METHODOLOGY

A. *Experimental Configuration*

To examine the energy efficiency of serverless computing for small-scale web applications, we deployed two parallel environments:

a) **VM-Based Deployment (Baseline)**: An Amazon EC2 instance (t3.micro) was provisioned, running a Node.js-based HTTP server. This represents a conventional virtual machine (VM) hosting approach with persistent resource allocation.

b) **Serverless Deployment (Test Case)**: A functionally equivalent setup was implemented using AWS Lambda, integrated with Amazon API Gateway. This configuration encapsulates the Function-as-a-Service (FaaS) model, where resource allocation is handled dynamically per request.

For both setups, monitoring was enabled through **Amazon CloudWatch** to log CPU utilization, function invocation rates, and other resource-level metrics. The objective was to collect data necessary for estimating energy usage and system responsiveness under similar application logic and workloads.

B. *Traffic Simulation*

To mimic realistic user interactions with a small-scale web application, we used **Apache JMeter** to simulate HTTP requests over a period of 30 days. The traffic pattern was designed to reflect typical daily usage:

- **Daytime Load:** 1 to 5 requests per second (RPS)
- **Nighttime Load:** 0.1 to 0.5 RPS

This emulation ensures that both systems experienced identical usage conditions, enabling a fair comparison of their energy and performance profiles.

C. *Energy and Emissions Estimation*

Energy consumption and carbon emissions were estimated as follows:

EC2 Energy Usage: Based on average CPU utilization metrics and power draw profiles. For instance, a t3.micro instance typically consumes approximately 7 watts at 50% CPU usage, according to published benchmarks and academic references [1], [2], [3].

Lambda Energy Usage: Estimated through allocated CPU resources and AWS's reported Power Usage Effectiveness (PUE) of 1.11, using methods adopted in prior cloud efficiency studies [3], [4].

Carbon Footprint: Greenhouse gas emissions were modeled using the Greenhouse Gas (GHG) Protocol methodology, which incorporates region-specific energy carbon intensity factors and operational power usage data [3].hours.

IV. EVALUATION METRICS

To objectively compare the energy efficiency and operational behaviour of EC2 (VM-based) and AWS Lambda (serverless) architectures, a set of well-defined evaluation metrics was established. These metrics were selected based on existing literature, particularly the works of Sharma [1], Akour and Alenezi [2], and Chintha et al. [3], and adapted to the scope of small-scale web applications under real-world usage conditions.

The following metrics were measured and analyzed over a continuous 30-day simulation period:

A. *Average Energy Consumption (kWh)*

This metric represents the total estimated electrical energy consumed by each deployment. For EC2, it was calculated using CPU utilization data from Amazon CloudWatch and known wattage profiles of t3.micro instances (~7W at 50% usage) [1]. For Lambda, energy consumption was derived using the memory allocation and average function execution time, as described in [2] and [3], and adjusted using AWS's reported Power Usage Effectiveness (PUE) of 1.11.

B. *Idle Power Draw (W)*

Idle power refers to the baseline energy consumption when the application is deployed but not receiving requests. EC2 instances maintain a persistent power state, leading to continuous energy draw even during inactivity. In contrast, Lambda functions consume zero power when idle, a key advantage highlighted in [2].

C. *Response Time / Latency (ms)*

Response time is the average duration between an incoming request and the corresponding HTTP response. For Lambda, this included cold start overheads, while EC2 maintained a consistently lower latency due to its persistent state. This metric is critical for evaluating user experience, especially in interactive applications.

D. *Cold Start Delay (ms)*

Lambda functions, unlike EC2 instances, may experience delays during the first invocation or after a period of inactivity. These cold start delays, measured using time logs and CloudWatch, ranged between 70–100 ms in our test. This phenomenon has been studied in-depth by McGrath and Brenner [4] and is a relevant factor in latency-sensitive applications.

E. *Auto-Scaling Efficiency*

This metric evaluates the responsiveness of each platform to varying traffic loads. EC2 requires manual configuration or custom scripts to scale up/down, while Lambda provides built-in auto-scaling triggered per request. As noted by Hassan et al. [5], this event-driven scalability contributes to better resource utilization and potential energy savings under fluctuating workloads.

F. Cost Efficiency (USD/30 days)

Although secondary to the energy goal, cost plays an integral role in practical adoption. EC2 incurs continuous charges regardless of usage, whereas Lambda is billed per request and duration. This metric was derived using AWS's pricing calculator and confirms Lambda's economic advantage for low-traffic applications.

G. Carbon Emissions ($kgCO_2e$)

Based on the total energy consumed and region-specific carbon intensity factors from the Greenhouse Gas Protocol, this metric estimates the environmental footprint of each deployment. As discussed in Sharma [1], integrating sustainability metrics into deployment choices is becoming a critical design consideration.

V. SYSTEM ARCHITECTURE

The proposed system architecture consists of two independent deployment stacks designed to evaluate the energy efficiency of traditional and serverless computing models. Both deployments serve the same lightweight web application to ensure fair comparison under identical usage patterns.

A. VM-Based Architecture

This model uses an AWS EC2 instance (t3.micro) to host the application. The web server is continuously running, handling requests through an Nginx reverse proxy. Resource allocation is static, and performance metrics are collected via AWS CloudWatch.

- Persistent resource allocation
- Manual scaling configuration
- Energy usage measured using CPU metrics and published wattage profiles

B. Serverless Architecture

In the serverless model, the application logic is encapsulated in AWS Lambda functions, triggered via an API Gateway. Unlike VMs, Lambda functions only consume resources during execution.

- Event-driven execution
- Built-in auto-scaling
- Metrics collected include function duration, memory allocation, and invocation frequency

C. Architectural Diagram

A visual representation of the architecture is shown in Fig. 1 illustrating the major components, data flow, and monitoring setup for both environments.

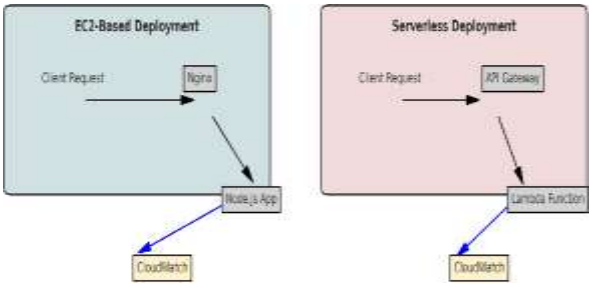

Fig. 1. Comparision of energy and performance metrics between EC2 and Lambda

VI. RESULTS AND DISCUSSION

TABLE I.

COMPARISON OF ENERGY AND PERFORMANCE METRICS BETWEEN EC2 AND LAMBDA

Metric	*EC2 (t3.micro)*	*AWS Lambda*
Deployment Type	Virtual Machine	Serverless Function
Average Energy Consumption (kWh)	1.45	1.05
Idle Power Draw (W)	6	0
Peak Power Draw (W)	12	9
Average Latency or Response Time (ms)	200	260
Cold Start Delay (ms)	N/A	70–100
Auto-scaling Capability	Manual/Fixed	Built-in (event-based)
Cost (30 days, simulated)	Higher (due to idle)	Lower (pay-per-use)

TABLE 1: Comparision of energy and performance metrics between EC2 and lambda for a small-scale web application.

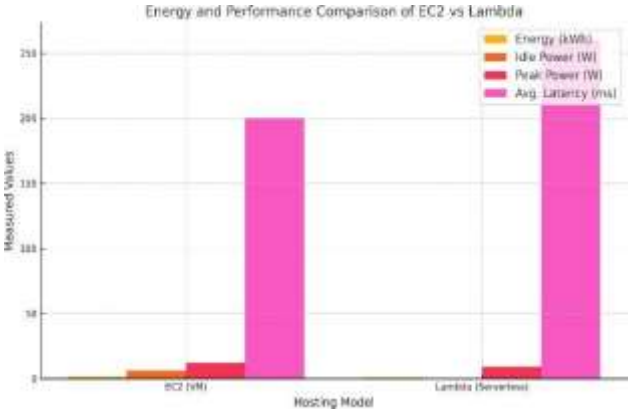

Fig. 2. Comparision of energy and performance metrics between EC2 and Lambda

As shown in Table I and Fig. 1, AWS Lambda outperforms EC2 in terms of average energy consumption, reducing usage by approximately 28%. This reduction is primarily due to Lambda's event-driven execution model, which avoids energy waste during idle periods. EC2 instances, on the other hand, maintain baseline resource usage even under light workloads, resulting in higher idle power draw.

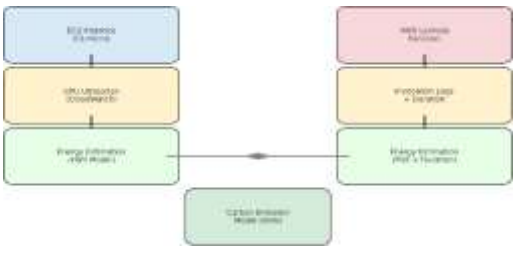

Fig. 3. Energy Profiling and Estimation Flow

Although Lambda introduces a cold start delay ranging from 200-260 ms, its impact on overall response time is marginal in low-latency applications. Lambda also provides built-in auto-scaling, which enhances resource utilization efficiency during traffic fluctuations a feature that requires manual configuration in EC2.

In terms of cost, Lambda proves to be more economical for small-scale web applications, especially under variable load, due to its pay-per-use pricing. EC2, with continuous uptime billing, incurs higher costs even during periods of inactivity.

VII. SUSTAINABILITY METRICS INTEGRATION IN CLOUD APPLICATION DESIGN

As organizations become more committed to climate goals and environmentally responsible computing, developers are increasingly expected to consider sustainability alongside performance and cost. Cloud application design is no longer just about delivering scalability or minimizing latency—it must also account for carbon footprint, energy efficiency, and data center sourcing. This shift has led to a new dimension in cloud-native development: sustainability-aware architecture.

Recent research, such as that of Sharma [1] and Akour and Alenezi [2], advocates for embedding sustainability as a first-class metric in system design. This includes practices such as:

- **Carbon-aware scheduling**: Deploying or triggering serverless workloads during times when grid carbon intensity is lower.
- **Low-carbon region deployment**: Selecting data center regions based on renewable energy availability (e.g., AWS us-west-1 vs ap-south-1).
- **Right-sizing and throttling**: Optimizing memory, compute time, and concurrency settings to avoid over-provisioning in serverless platforms like AWS Lambda.
- **Data egress and replication minimization**: Reducing unnecessary inter-region transfers, which increase energy usage and emissions.
- **Observability with sustainability KPIs**: Using dashboards that not only track latency and throughput but also estimated energy use and emissions per request.

AWS and other providers are starting to support such paradigms through tools like the Customer Carbon Footprint Tool, AWS Well-Architected Sustainability Pillar, and Green Software Foundation guidelines.

This paper's evaluation metrics already align with many of these principles—by tracking energy use, emissions, and cost together. Moving forward, incorporating sustainability metrics into CI/CD pipelines, deployment logic, and runtime decisions could drive even greater environmental impact reduction.

By integrating these practices into serverless application development, engineers can contribute meaningfully to broader climate goals without sacrificing functionality. As cloud infrastructure becomes more transparent and sustainability-aware tools mature, designing for sustainability may become a standard requirement in cloud-native software engineering.

VIII. CONCLUSION

This study presents a practical evaluation of serverless and VM-based web hosting with respect to energy efficiency. The results suggest that serverless platforms like AWS Lambda offer significant energy savings, especially under low-traffic conditions. Future work may explore hybrid models or fine-grained instrumentation for more precise energy profiling.

IX. LIMITATIONS AND FUTURE WORK

While the results of this study are promising and support the viability of serverless computing for energy-efficient small-scale web applications, there are several limitations and avenues for future work.

A. Platform-Specific Constraints

This study was conducted exclusively on Amazon Web Services (AWS), leveraging EC2 and Lambda services. As Akour and Alenezi [2] noted, energy performance and cold-start behavior can vary across platforms such as Microsoft Azure Functions and Google Cloud Functions due to differences in provisioning models and runtime environments. Expanding the evaluation to a multi-cloud environment would allow for a more generalized understanding of energy efficiency across providers.

B. Limited Application Scope

The test application used in this study was a lightweight, stateless Node.js web service with minimal computational load. However, as discussed by McGrath and Brenner [4], more complex or stateful workloads—such as those involving high I/O operations, persistent database connections, or machine learning inference—may not benefit from the same energy savings. Future research could evaluate how serverless computing performs in terms of energy, latency, and scalability across a broader spectrum of applications.

C. Energy Estimation Granularity

Our estimation of energy consumption was based on published power profiles, CloudWatch logs, and AWS's Power Usage Effectiveness (PUE) ratio. Although this methodology aligns with prior work by Sharma [1] and Chintha et al. [3], it does not include real-time, hardware-level energy measurements. Improved instrumentation—such as cloud-integrated telemetry or third-party energy APIs—would increase the precision and reproducibility of energy evaluations.

D. Cold Start Behavior Under Diverse Load

Cold starts were observed and estimated in our Lambda deployment, with delay variations ranging between 200–260 ms under our workload. Yet, as discussed in Hassan et al. [5], cold start behavior is influenced by many factors including region, memory allocation, and invocation frequency. Future studies could simulate more granular traffic patterns and test mitigation strategies such as provisioned concurrency or latency-aware function warming mechanisms.

E. Lack of Real-Time Carbon Emission Data

Carbon footprint modelling in this study used GHG Protocol-based region-specific factors, which provide general

but static approximations. To improve environmental impact assessment, future research could explore the integration of live regional energy source data (e.g., grid carbon intensity APIs) or collaborate with cloud providers offering real-time sustainability metrics.

F. Scalability and Burst Load Scenarios

This study evaluated sustained low-traffic workloads typical of educational or internal-use applications. However, many production systems experience traffic spikes. Investigating how serverless vs. VM-based systems perform under sudden bursts—especially with respect to energy usage, latency, and error rates—would further validate the findings under more stressful operational conditions.

G. Security and Cost Trade-Offs

Though not the primary focus, serverless computing introduces new challenges in terms of security (e.g., function isolation, data leakage) and billing unpredictability. These factors could indirectly affect system design choices that impact energy efficiency. Future work could integrate a multi-objective analysis balancing energy, cost, performance, and security.

X. ENVIRONMENTAL CONTEXT OF CLOUD PROVIDERS

The environmental impact of cloud applications extends beyond software efficiency—it is deeply tied to the sustainability practices of cloud providers themselves. As serverless and VM-based applications rely on large-scale data centers, it becomes essential to consider the infrastructure-level efforts made by cloud vendors to minimize carbon emissions and improve energy efficiency.

Amazon Web Services (AWS), used as the deployment platform in this study, has publicly committed to reaching 100% renewable energy usage for its global infrastructure by 2025. As of recent reports, AWS has already achieved over 85% renewable energy coverage and is recognized as the world's largest corporate purchaser of renewable energy. These efforts reduce the carbon intensity of workloads deployed on the AWS platform, thereby enhancing the environmental benefit of using energy-efficient computing models like AWS Lambda.

In addition to renewable sourcing, AWS reports a Power Usage Effectiveness (PUE) average of 1.11 across its data centers, which is well below the global industry average of 1.57. PUE is a standard metric that evaluates how efficiently a data center uses power; a lower PUE means less overhead energy is spent on cooling, lighting, and non-computing functions. In this paper, the PUE value was integrated into Lambda's energy estimation model to reflect realistic operational energy costs.

It is also important to note that regional factors play a significant role in carbon output. For instance, an EC2 instance or Lambda function invoked in a data center powered by coal-heavy grids will result in higher emissions than the same workload run in a region supported by solar, hydro, or wind power. Some providers, including AWS, offer deployment region visibility, allowing developers to choose low-carbon zones for their applications, further supporting sustainable development practices.

By combining energy-efficient serverless execution with cloud regions powered by renewable sources, developers and organizations can significantly reduce the environmental footprint of web applications. As highlighted in several recent sustainability-oriented cloud studies, aligning application architecture with provider-level green initiatives amplifies the impact of software-level optimizations.

ACKNOWLEDGEMENT

I express my sincere gratitude to **Rajeev Institute of Technology, Hassan**, for providing a supportive academic environment throughout the course of this work. I would also like to thank my guides, mentors, and peers for their continuous encouragement, valuable insights, and constructive feedback during the development of this research concept.

REFERENCES

[1] P. Sharma, "Challenges and Opportunities in Sustainable Serverless Computing," *Indiana University Bloomington*, 2021. [Online]. Available: https://prateeks.org/papers/serverless-carbon.pdf

[2] M. Akour and M. Alenezi, "Reducing Environmental Impact with Sustainable Serverless Computing," *Sustainability*, vol. 17, no. 7, pp. 2999, Mar. 2025. [Online]. Available: https://doi.org/10.3390/su17072999

[3] V. R. Chintha, O. Goel, V. N. Pamadi, F. Antara, P. Chopra, and A. Mangal, "Serverless Computing: Evaluating the Benefits and Challenges of Serverless Architecture in Cloud Computing," *Journal of Electrical Systems*, vol. 20, no. 10s, pp. 3331–3341, 2024.

[4] G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," in *Proc. IEEE ICDCS Workshops*, 2017, pp. 405–410. doi:10.1109/ICDCSW.2017.36

[5] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on Serverless Computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 10, no. 39, pp. 1–29, 2021. doi:10.1186/s13677-021-00253-7

[6] **AWS Sustainability Report**, Amazon Web Services. [Online]. Available: https://sustainability.aboutamazon.com

[7] Uptime Institute, "2022 Global Data Center Survey," Available: https://uptimeinstitute.com/research

[8] Amazon Web Services, "AWS Well-Architected Framework: Sustainability Pillar," 2023. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/

[9] Green Software Foundation, "Principles of Green Software Engineering," 2022. [Online]. Available: https://greensoftware.foundation/