National College of Ireland

Comparative Evaluation of Energy and Water Efficiency Across Cloud Deployment Models

MSc Research Project
MSc in Cloud Computing

Yashashwini Suresh
Student ID: 23251263

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Yashashwini Suresh
Student ID:	23251263
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Abubakr Siddig
Submission Due Date:	11/12/2025
Project Title:	Comparative Evaluation of Energy and Water Efficiency Across Cloud Deployment Models
Word Count:	8123
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Yashashwini Suresh
Date:	10th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative Evaluation of Energy and Water Efficiency Across Cloud Deployment Models

Yashashwini Suresh
23251263

Abstract

The rapid growth of cloud computing has increased the energy and water consumption of the globe but little research has been done comparing the feasibility of the trending deployments models to the same load. This gap is bridged in this paper, which compares environmental performance of three trending compute models, i.e., virtual machine (EC2), containerized services (Docker on EC2) and serverless functions (AWS Lambda). It is to understand how different architectures can impact the sustainability and also ascertain the best efficient model in delivering energy and water reductions at the changing workload conditions. To generate a microservice workload, this research has developed a controlled microservice benchmark to generate the homogenous compute and I/O bounded workloads. The deployment became repeated multiple times and Locust and AWS CloudWatch had been deployed to collect the metrics. Energy usage was estimated with the help of CPU-based power models (EC2 and Docker) and GB-seconds metrics (Lambda), and water usage was estimated with the help of standard datacentre WUE factors. It is found that serverless functions use much less energy and water, significantly, which is several folds less than the amount of the two consumed by traditional servers, particularly at low and medium loads. Containers also outperform low-load virtual machines and scale to high load as hardware usage is which is not markedly better. These findings can be used to reinforce the existing literature on the overheads of virtualization and to highlight that serverless computing is a highly sustainable solution to idle, short-lived workloads. More work should be added in terms of the number of providers, work types, and functions, which take longer periods.

Keywords- Virtual Machine, Containers, Energy metrics, Water metrics, Serverless Models

1 Introduction

Cloud computing has turned out to be the most popular model of computation resources provision, which organisations may use to deploy applications based on virtual machines, containers, or serverless services. These models vary in terms of cost, scalability, and operational overhead, but none of them does not use large-scale datacentres the energy and water consumption of which is ever-increasing. According to the recent industry reports, it is estimated that cloud datacentres are consuming a few percent of the world electricity and it is projected that the consumption will go higher as the workloads escalate. With sustainability being a central aspect in digital infrastructure, researchers and

practitioners are increasingly worried not just about the performance and cost, but also about the effects that cloud deployments have on the environment.

However, in spite of this increased focus, little research has compared the sustainability properties of virtual machines, managed containers, and serverless functions to be used in the same conditions of work and with the real cloud infrastructure. A great portion of the literature considers these technologies on a case-by-case basis, or looks at performance or at the cost, but not energy or water consumption. This means that organisations do not have clear indication of the deployment model that would provide the most efficient environmentally when using various workload patterns[1].

The proposed research will close this gap [1][2] and will be based upon the empirical comparison of three popular deployment models, namely Amazon EC2 (virtual machines), Docker containers upon EC2 and AWS Lambda as serverless[3]. Every model in this research were performed with the same compute intensive and I/O intensive workloads, and were repeated to minimize changeability. In the estimation of the sustainability effects, the paper utilizes energy-measurement techniques based on the current available literature on green computing, which are power-based energy modelling and the application of datacentre water-usage effectiveness (WUE)[2].

Nevertheless, there is no research that compares virtual machines, containers, and serverless functions with the same workload, and the same tools, in the same cloud region. The majority of the previous literature separates one technology or pays attention to system performance only, leaving the issue of sustainability trade-offs mostly untouched [4]. This poses a methodological gap to the practitioners who have to select the compute models without environmental guidance.

The present study is based on literature as it has a controlled and empirical basis comparison of EC2, Docker-based containers, and AWS Lambda based on replicated load tests, CPU-based energy estimation models and sustainability indicators based on datacentre-level metrics. This goes in line with the current sustainability systems, and expands the research gaps uncovered on the latest cloud sustainability surveys [4, 5].

The aim of the work is to not merely measure the energy and water usage of VM, container and serverless deployments, but to learn the variations in these differences during low, medium and high load conditions. In this way, the research can be useful to developers, architectures from cloud, and sustainability scholars who want to make environmentally conscious choices regarding deployment models. Finally, this study presents the impact of architectural decisions on the sustainability results and proves that cloud efficiency should be measured not only in terms of cost and performance indicators[3].

1.1 Motivation:

Cloud computing has been the foundation of the contemporary digital services, but its environmental footprint is increasing at a high pace. Recent research points out how data centres have been consuming a lot of power in terms of electricity to compute and cool and this has led to high emission of carbon and fresh water depletion[6][7] [8]. With organisations shifting more to cloud-native platforms (be it virtual machines (VMs) or containers and serverless), it is now environmentally and economically necessary to understand the sustainability implications of each paradigm.

Even though the work that was done before analysed the performance and cost trade-offs among compute models, relatively little is known about the virtual machines, containerised applications and serverless functions differences in energy consumption, water

usage and overall sustainability. Recent literature tends to contrast these paradigms individually, that have varying workloads, measurement instruments, or varied cloud locations and cross-model comparison is not valid [4][5]. Moreover, the change towards more sustainable cloud computing habits has created an added pressure on the empirical evidence that characterizes the sustainability output of various deployment models on actual cloud settings.

This drives the desire to have a single, controlled and repeatable methodology that has the ability to assess large compute paradigms within the same workload conditions. The definition of the sustainability features of each model will enable cloud architects and researchers to make better choices regarding the environmentally friendly system design.

1.2 Research Question:

To fill the gaps in the literature, the following research question is presented in this investigation:

Question: To what extent do virtual machines, managed containers, and serverless functions differ in terms of sustainability (energy and water efficiency) when executing the same workload under varying load conditions on AWS?

The sustainability metrics to be quantified by means of this research question include energy (kWh), water use (L), and efficiency (requests per kWh/L), which allows comparing the three execution paradigms in a rigorous way. It also focuses on the same workloads, to achieve methodological fairness and isolate dissimilarities that can be explained by the underlying cloud architecture.

1.3 Hypotheses:

According to the previous studies in cloud sustainability, autoscaling, and resource-efficiency modelling [9][5], hypotheses were developed as follows:

- **H0 - Null Hypothesis:** Virtual machines, managed containers, and serverless functions do not have any substantial sustainability (energy per request and water per request) differences when running the same workloads.
- **H1 - Serverless Benefit at Low Workloads:** Serverless functions will have better sustainability efficiency (more requests per kWh and per litre of water) at low workloads than EC2 and Docker do. This is because they scale finely, and receive no overhead due to idle capacities, as confirmed by serverless energy research[9][8]
- **H2- Container Advantage in High Workloads:** Container-based deployments will be more efficient in terms of sustainability performance at high workloads compared to EC2 and serverless functions. This is due to containers amortising the overhead of resources, lowering cost of virtualisation than VMs, and not having the constraint of serverless scaling during sustained load [10][5].

2 Related Work

The importance of sustainable cloud computing has continued to grow dramatically over the past few years with organisations facing increased energy expenses, carbon emissions, and freshwater usage in conducting data centre operations. It had been early determined

that a cloud infrastructures burns enormous amounts of power, and uses water to cool and that the energy demand of the global data-centres is expected to keep rising as the digital services proliferate [11][12]. Researchers like Shehabi et al. and Hintemann and Clausen emphasize that in many cases, the difference in efficiency of resource utilization can lead to enormous environmental effects when repeated billions of times among millions of servers [1][2].

2.1 Work Related to Sustainability and Virtual Machines:

VMs are well isolated with significant overheads on resources in terms of hypervisor, idle CPU cycles, and non-adaptive provisioning. Initial research indicates that VM-based systems tend to cause an inefficient energy behaviour due to the resources allocated to them being powered even when under low use [3][4, 13]. The sustainability taxonomy by Gill and Buyya states that VM overprovisioning is a significant cause of unwarranted carbon and water footprint in the cloud [4].

Comparative analysis Comprisers of VM-based hypervisors are confirmed to consume a lot more power, when running the same workloads, than their lightweight counterparts [10], which support their shortcomings in the case of situations that need to be efficient at scale.

2.2 The Emphasize of Energy Saving and Containerisation:

Containers, which have also been presented as a lightweight alternative to VMs, use the host kernel, but separate applications on the process level. Studies have continuously demonstrated that container-based deployments decrease the overhead, enhance the consolidation of resources, and lower the number of idle energy use [14][15].

Recent work, such as [5], has shown that containers offer significant runtime energy efficiency, particularly in the situation of bursty or multi-tenant workloads, since they resource-utilisation amortisation is more efficient than hypervisor-based workload management. There are also comparative findings that the containers are more energy proportional and CPU efficient compared to VMs at medium and high loads[10] .

2.3 Fine-Grained Execution Models and Serverless Computing.

The third mode of execution is serverless platforms like AWS Lambda, Azure Functions and Google Cloud Functions where resources are only deployed when processing requests. It has been demonstrated through the research by Alhindi et al. and other scientists that serverless functions are highly sustainable at low utilisation, which results in zero unused energy waste [9].

The serverless models however add inefficiencies:

- Cold-start delays
- Greater per-invocation overhead.
- Concurrency throttles that are platform specific.
- Slow performance at long work intervals.

Examples of literature like **Serrano-Gutierrez et al** also suggests that the choice of architecture in serverless architectures can have a considerable effect on the energy behaviour, especially when functions grow quickly [8].

2.4 Energy and Water usage modelling in Datacentres.

Existence of basic sustainability models, based on PUE and WUE, give facility level multipliers turning IT energy into the total environmental impact [1][2][16]. Because cloud providers do not allow the measurement of power at the server level, scientists have resorted to proportional models to measure power based on CPU-utilisation [11][13]which approximates power by linear relationships between CPU activity and active server power consumption.

These models have become a normal practice in empirical research on cloud sustainability due to their great connection to datacentre energy proportionality.

2.5 Gaps Chosen in the Literature-

In all the domains mentioned above, there is a common motif: There is a very limited number of studies that compare VMs, containers, and serverless functions directly under the same application, workload, and cloud. Previous Work:

- Instances are tested against one paradigm only containers or only serverless.
- Uses non-homogeneous workload generators during research.
- Ignores consumption of water.
- To minimize cloud noise Lacks replicated runs.
- Concentrates on cost/performance and no environmental measures.

This study focuses on filling that gap by giving a complete controlled, repeated and cross-model comparison of sustainability.

2.6 Critical Literature Review.

Despite the fact that the current amount of literature provides a substantial knowledge about datacentre efficiency, the sustainability of computing and performance of clouds, it is possible to identify several limitations that can be observed in the previous studies. To start with, most energy and sustainability assessments single out one execution model, either virtual machines, containers, or serverless, and do not directly and particularly controlled comparisons among all three.This decreases the overall generalisability of the conclusions as the performance and sustainability features largely rely on the type of workload, scaling behaviour, and runtime configuration. Comparative literature including work by Bahadoor [10] and Beena [5] show the advantages of container energy, but only compare containers to hypervisors, and it remains unclear how containers perform when compared to serverless or VM-based implementations in the same situation.

Second, an array of studies use synthetic or heterogeneous workloads thus it is hard to normalise results. It has been established that the workload characteristics have a strong impact on energy behaviour in that compute-intensive workloads are better represented by

Theme	Key Findings in Literature	Representative Studies
Datacentre sustainability & modelling	Increase of Energy/WUE; Use of CPU-proportional models; Importance of PUE/WUE scaling facility	[6][7][16]
Virtual machines-VMs	Cost for High-idle; inefficient hypervisor; increase energy/watt usage while they are over-provided	[3][4][10][13]
Containerisation	Minimal Costs; efficient resource combination; energy saved in med & high loads	[5][10][14][15]
Serverless computing	Great for low workloads; dismisses unused energy; affected by delayed starts and occurrence overhead	[8][9]
Models on Energy Assessment	CPU-based evaluation adopted widely; validated by sustainability work	[11][1][2][13]
Cross-pattern comparisons	Limited or absent; very few have studied all three under similar conditions	[4][5]

Table 1: Related Work and their Key Findings

proportional power models, on the other side I/O-bound workloads reveal inefficiencies in platform overheads. However, not many studies have a dual-workload that encompasses CPU-bound and I/O-bound work and thus restricts generalising sustainability trends across application domains.

Third, some previously published research on the use of serverless computing, such as those by [9] and Serrano-Gutierrez et al. [8] offer useful information about how to estimate the energy use of functions, but tend to have small testing windows, cold-start bias, or platform throttling, which are all likely to mask steady-state sustainability properties. Also, serverless studies are seldom executed with repeated trials, despite the fact that cloud environments are highly varient under multi-tenancy, noisy neighbours and dynamic resource allocation.

Fourth, although the theoretical basis of PUE, WUE and CPU-based power modelling has been established in the literature [11][1][2][16], the majority of empirical research lacks the inclusion of water usage or facility level efficiency multipliers in the sustainability analysis. With water shortage becoming a pressing problem worldwide, the process of sustainability evaluation should exclude water indicators, because otherwise it becomes incomplete and skewed towards the energy-only logic.

Lastly, not many papers undertake replicated trials which have statistically significant averages. A number of literature findings rely on a single experiment, and they can be affected by short-term variations in cloud infrastructure. This undermines trust in their trends that are reported.

3 Methodology

The aim of this methodology is to present a complete, transparent and reproducible account of how the study was conducted, developed and assessed in accordance with the current scientific requirements of experimental studies on cloud sustainability [1][2],

serverless benchmarking methodologies [9], and energy-aware computing research [13][11]. This study was based on the comparative experimental design, which sought to determine the sustainability attributes of three cloud execution frameworks, namely Virtual Machines (EC2), Managed Containers (Docker on EC2), and Serverless Functions (AWS Lambda) using the same compute and I/O workloads. The experiment deploys a controlled microservice to execute as per methods that have been used in previous sustainability benchmarking studies [13][14][15], such that is workload-agnostic and reproducible.

The research has the following procedures:

- **Development of Microservice:** Stateless mircoservice was build to perform CPU-intensive and I/O-intensive endpoints.
- **Deployment:** Deploy the above micro service to all three deployment models - AWS EC2, Docker on EC2 and Lambda.
- **To Generate Workload:** Write controlled synthetic workloads with Locust at low, medium and high load intensities [14][17].
- **Collection of Values:** Gather system level metrics such as CPU usage, number of invocations and execution times with AWS CloudWatch.
- **Energy Profile:** Use models of energy-estimation based on adapted formulae based on cloud-energy literature [13][11] and datacentre PUE/WUE modelling literature [1][2].
- **Iteration and Averaging:** Accumulate the result on 10 repeated runs on each deployment model in order to eliminate noise and random variation.
- **Analysing the Hypotheses:** Critically analyse the outcome from the models and compare the results of each model with the Hypotheses.

It is a repeatable, cross-model comparative and meets the requirements of empirical evaluation methodology[1][14][15].

3.1 Tools Used in this Research

The implementation of this research was done by using a combination of cloud services, software tools and development environments, which were used to enable controlled, repeatable and scalable experimentation. The microservice was coded in Python 3.11 with FastAPI framework providing a lightweight and high-performance framework that can be benchmarked. Locust was used as the initial load generating tool because it can replicate the real user traffic and has the ability to adjust the workload intensity precisely [14][9].

All the experiments were run on the Amazon Web Services (AWS) with three compute models:

- EC2 virtual machines
- Docker containers on EC2
- AWS Lambda serverless functions.

Tool / Service	Role in the Study
Python 3.11	The microservice and Lambda functions are implemented using this. Python was selected as it is lightweight, is supported on AWS widely and can be benched easily.
FastAPI	The model that was employed in the construction of the compute and I/O endpoints. It offers low overhead and low response time hence suitable in controlled experiments.
Uvicorn	The EC2-based API server powered by Docker. It performs highly in concurrency, as it aided in load tests.
Docker	Used to package the microservice and make comparisons between containerized execution on the identical EC2 hardware.
AWS EC2	Gave the virtual machine environment of the model of the initial deployment. It provided complete control of the operating system.
AWS Lambda	Acted as the serverless execution model, which allowed it to compare with event-driven resource provisioning.
Amazon ECR	Storing Docker images to be deployed on EC2.
Terraform	Automated EC2 instance creation, security group creation, and networking configuration creation to provide consistent experiments.
AWS CloudWatch	Used for CPU metrics (EC2/Docker) and invocation values (Lambda). These values were directly used to compute energy and water.
Locust	Created synthetic traffic at low, medium and high workloads. It enabled the management of the number of users, the rate at which they spawned, and their time.
GitHub	Used to do version control and maintain the codebase in line with microservice, Lambda, and Terraform files.
Visual Studio Code -VSCode	Principles: This element depends on the programmer's workstation that contains the coding system.

Figure 1: tools, services used

Terraform was used to provision infrastructure resources on a replicable basis, which can be used in repeated trials. The AWS CloudWatch was used to perform metric collection, and it gave us CPU utilisation, invocation metrics, and duration values that could be used to do sustainability calculations. GitHub was used as version-control software to coordinate the changes in the microservice, the Lambda functions, and the Terraform settings. The local development and experimentation were all done through Visual Studio Code (VS Code) with WSL2 so that a stable and portable development environment can be created.

There was also a uniform microservice implementation to bring equity in all the models of deployment. Two kinds of operations were revealed by the service:

1. A CPU-intensive endpoint, based on a deterministic Fibonacci calculation (n=35). This Follows the benchmarking standards from [14][9].

2. An I/O-intensive endpoint, based on JSON serialisation and deserialisation to imitate lightweight operations.

Each deployment model had the same microservice logic and request patterns. The

application was deployed directly on the virtual machine on EC2 via Uvicorn. Under the containerised model, the application was bundled into a Docker image, and it was run with resource settings under control. In the case of AWS Lambda, the logic has been divided into two distinct functions one of calculation and one of I/O to replicate the behaviour of the microservice and conform to the event-driven programming model of Lambda.

Three levels of workload namely low, medium, and high workload were outlined on the basis of number of users and rate of requests. The workloads were run under a constant time (6 minutes on EC2 and Docker and the recommended shorter window Lambda to prevent throttling) [9]. Each workload setup had ten repetitions to obtain variability and obtain statistically significant averages. Each run was given a start and end timestamp to co-ordinate the execution of the workload with CPU utilisation and invocation metrics available on CloudWatch.

Load Setup for the Models

Workload	Users (u)	Spawn Rate	Duration
Low	50	5/sec	6 min
Medium	200	20/sec	6 min
High	800	80/sec	6 min (for EC2 & Docker), 3 min for Lambda

Table 2: Load Setup for EC2, Docker, and Lambda

Three workload intensities were determined according to previous frameworks of cloud benchmarking: [9, 14, 15]. Each workload was run 10 times in each deployment model resulting in 270 runs.

3.2 Collection of Data and Their Measurements

The data collection was based on the need to gather performance, utilisation and sustainability related metrics of every deployment environment. To introduce uniformity all measurements were done on a purely controlled experimental window characterized by the start and end timestamps of each Locust run. These timestamps were taken directly out of the resulting CSV files in order to fix the metric retrieval staging process and eliminate overlap or drift between successive executions.

EC2 and Docker deployments: the main metric of interest was the CPU Utilization which was accessed in AWS CloudWatch at aggregation intervals of one minute (60-second). This metric is consistent with the existing approaches to estimating the amount of energy used in cloud environments [11][13][15]. The CPU data of each run was gathered with the help of the AWS CLI allowing to retrieve it automatically and accurately compare the duration of the workload with the measured resource utilization.

AWS Lambda deployment: the metrics on CPU utilisation are not accessible because of the obfuscated serverless execution. Instead the metrics provided by CloudWatch like Invocations, Average Duration (ms), Throttles and Errors were utilized. Invocation count and average duration per run were used as proxies of the total compute time per run, which was in agreement with the AWS Lambda Power Model [9].

$$\text{Compute Time} = \text{Invocations} \times \text{Duration}_{\text{avg}} \tag{1}$$

All raw measurements such as the response times, throughput, error rates, CPU utilisation, and Lambda execution measures were recorded in structured CSV files. This

provided both transparency and reproducibility of all calculations and also provided statistical processing later in the process.

Data Collected: They were structured and post processed in order to come up with robust and reliable results.First, all the metric sets (e.g., CPU utilisation, invocation counts, response-time percentiles) were located in ten repeated workload and deployment model runs. These ten runs were averaged to give a representative value of each category, and the repetitive method of doing so contributed to minimizing the noise due to background activity of clouds or temporary fluctuations.

The Interquartile Range (IQR): Used to identify outliers and these were inspected manually to decide whether they were due to cloud-based anomalies (e.g. cold starts, network jitter, throttling) or were a natural part of system behaviour. Only drastic anomalies that did not coincide with common patterns of execution were eliminated.

Energy Consumption Calculation: This was done based on standardised formulae based on the literature on data-centre energy modelling [11][1][2]. Then the water usage was estimated with the help of the public Water Usage Effectiveness (WUE) value by AWS [1][2]. These calculated metrics enabled fair comparison of all three compute models though they had different execution paradigms.

Finally, descriptive statistics summarised the results, which were qualified to assess the hypotheses of the study qualitatively. Since this was an exploratory study and with the unpredictable nature of a shared cloud hardware, focus was on trend analysis and not an inference hypothesis test. This is a well-known technique in empirical studies of cloud performance and is such that the conclusions are made by being based on observable system behaviour [3][14][15].

3.3 Deployed Models

The paper contrasts three possible models of deployments that are representative of the most important computational models of the contemporary cloud computing: virtual machines, containerised applications, and serverless functions. Every model evaluates the microservice implementation uniformly to ensure that a variation in sustainability and performance is not founded on the variant of the functional difference in the code.

The initial model is the Amazon EC2, which is the conventional virtual machine-based model where the microservice is run directly on dedicated t3.medium instance with Uvicorn serving it as an application server. This deployment brings the entire operating-system stack into the limelight and is allocated fixed compute resources during the whole duration of the experiment, independent of the intensity of workload. Consequently, EC2 can serve as a metric of gauging the energy and water overheads of persistent resource provisioning, which has been pointed out in the previous studies of VM-based cloud infrastructures [3, 13].

The second model applies Docker containers that are on the same instance type of EC2. In this case the microservice will be a container image (lightweight) and will be launched using the Docker engine or ECS in EC2 mode, whereas containerization provides the runtime of the isolated application by also sharing the underlying host kernel. This model is applicable to the contemporary cloud-native approaches in which containerisation minimizes operating system overhead and enhances the consolidation of resources.which is backed by empirical research on container performance [14, 15]. Since the type of container is another physical host that runs on the same physical host type as the EC2 baseline, it is possible to simply compare the effect of the virtualisation method

on sustainability.

The third model AWS Lambda is a complete serverless model. Compute and I/O endpoints use Lambda functions as independent and are accessible by means of Lambda Function URLs. In contrast to EC2 and containers, Lambda allocates resources on-demand, when a request is made, and then it scales itself automatically, depending on traffic. This fine-grained elasticity removes idle capacity and has been studied many times in literature about serverless benchmarking [9]. The CloudWatch metrics of invocation count and function duration are used to infer resource utilisation in line with the well-known methods of energy-estimation in serverless systems [9].

Combined, these three deployment models permit an ordered comparison of persistent, containerised, and event-driven execution environments, making the study possible to analyse the effects that architectural variation has on sustainability at different workload levels.

3.4 Metrics Used for Sustainability Measurements

This paper compares the sustainability and performance properties of the three models of deployments, namely EC2 VMs, Docker containers, and AWS Lambda, with a single collection of quantitative metrics widely used in the research of cloud sustainability. The measurements used to gauge performance are latency (median and tail percentiles) and throughput (requests/second) retrieved directly by Locust workload traces. To examine sustainability, CPU utilisation provided by CloudWatch is used as the main proxy of resource usage as earlier studies have established that utilisation is proportional to power draw using proportional modelling [11, 13, 15].

The estimated energy usage (kWh) is calculated using linear utilisation-to-power formula with a 60 watt server baseline and the water usage (litres) is calculated by using the published Water Usage Effectiveness (WUE = 1.8 L/kWh) by Amazon, which is based on the current datacentre water-efficiency studies [1, 2]. To determine efficiency, the paper standardises the impact of the environment by computing requests to kilowatt-hours (request/kWh) and requests to liters of water (request/l Water) and shows the model efficiency in transforming energy and water into useful computational work. These metrics can enable the holistic comparison of the performance, energy behaviour and sustainability of the three compute paradigms.

In order to achieve the comparability of the compute models, the following measurements were chosen, all based on cloud sustainability literature [1, 18].

Compute Metrics

- Requests per second (RPS)
- Percentile of response time (50th, 90th, 99th)
- Error rate

1. Sustainability Metrics:

Energy Consumption (kWh) Derived using data-centre CPU power model:

$$E = \frac{\text{CPU\%} \times P \times T}{100 \times 3600} \tag{2}$$

where P = 60W (typical power draw from AWS Xeon server) T = 360 seconds (6 min)

2. Water Consumption (Litres):

$$W = E \times WUE \tag{3}$$

with WUE = 1.8 L/kWh, based on AWS public reports.

3. Requests per kWh (Efficiency)

$$R_{kWh} = \frac{\text{Total Requests}}{E} \tag{4}$$

4. Requests per Litre of Water

$$R_{kWh} = \frac{\text{Total Requests}}{E} \tag{5}$$

Finalized computing metrics for the Research: These metrics are validated from the papers [1, 2, 11] the energy consumed in EC2 and Docker models are computed as:

$$E_{\text{kWh}} = \frac{\text{CPU\%} \times P \times T}{100 \times 3600} \tag{6}$$

$$Water = Energy \times WUE \tag{7}$$

- Where: P is 60W, which aligns with the research paper [13, 15] with typical server active-power profiles used for energy-modelling.
- T = 360 secs
- WUE = 1.8 L/kWh, follows the datacentre water usage models [1, 2]

For AWS Lambda deployment model, consumption of energy is calculated using and invocation-based model aligning with the studies of serverless power-measures [9]:

$$E_\lambda = \frac{Invocations \times Duration_{\text{ms}}}{1000} \times Power_\lambda \tag{8}$$

Where PowerLambda = 7W follows the measurements reported in energy-analysis for serverless model literature [9].

4 Design Specification

This part outlines the general structure and operation of the experimental system to compare the sustainability of three cloud deployment models, which are EC2 virtual machines, Docker containers on EC2, and AWS Lambda serverless functions. The architecture also allows every model to execute the same microservice, to get the same workloads, and to present similar metrics on the estimation of energy and water.

4.1 System Architecture

In this section, four key architectural layers in the system are disaggregated. Each of the layers has a designated role in providing the ability to make the experiment repeatable, fair, and similar in the three deployment environments.

1. Load Generation Layer or The Client Layer

The layer is charged with the responsibility of ensuring controlled, repeatable work loads to all the three deployment models. Locust is the load-generating tool of choice since it provides customizable user behaviour, the spawn rate, and test time. Every test is executed as realistic user traffic whereby bursts of the HTTP requests are sent to the microservice endpoints. Locust too captures detailed statistics e.g. response times, throughput and errors, which are exported to CSV files to be analyzed later. The tests make the performance of each system to be measured independently and unbiased by maintaining the load generator out of the deployment environments.

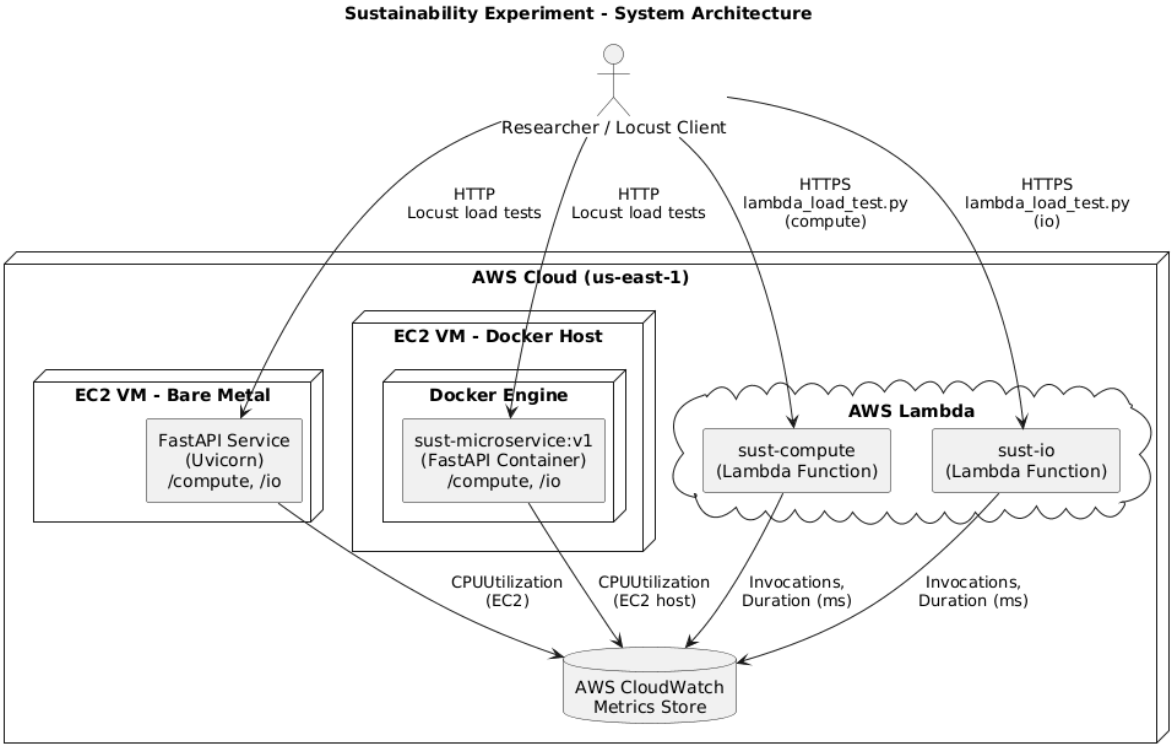

Figure 2: Architecture Diagram for Sustainability Comparison

2. Application Layer - Microservice Deployment

The Core part of this experiment is a lightweight microservice on a single FastAPI containing two endpoints they are I/O endpoint for JSON Serialisation task and Compute Endpoint for Fibonacci Calculation. This microservice is the same in all deployments and any difference in energy, latency or resource consumption is a result of the hosting environment, but not the logic of the application. Deterministic operations are used, which enables a comparison between performance to be very accurate and noise due to external dependencies is minimal. The stateless nature of the application provides it with a perfect use in the cloud-native deployment in VM, container, and serverless environments.

3. Deployment Layer

Each compute model in this layer contains three different microservices.

- EC2 Virtual Machine is a Uvicorn-based application with an Amazon Linux 2 instance that offers a traditional always-on cloud environment.
- Docker on EC2 bundles the microservice into a slim container image which will execute on the same instance type and the application will be isolated but share the OS kernel.
- AWS Lambda is an execution of the microservice logic as event-driven functions, which is only provisioned upon an invocation.

All models have varying degrees of abstraction and resource management where the sustainability of persistent, containerised and serverless paradigms can be easily compared.

4. Monitoring, Data and Analysis Layer.

AWS CloudWatch is the main monitoring system, which gathers CPU usage of EC2 and Docker deployments and invocation metrics of Lambda. Locust provides runtime metrics like throughput and latency and analysis scripts combine CloudWatch with measurements of workloads to calculate sustainability metrics (kWh, water use, requests per kWh, etc.). This layer provides the ability to trace the results, properly time-align the results, and to be able to repeat the same results across the 10 consecutive runs of each workload.

5. Component View

To make a brief description of the architecture in the report, it is as follows:

- Locust Client - creates artificial workloads (low, medium, high) and statistics of response-time are recorded.
- EC2 VM Service - directly executes uvicorn services.app.main:app; it is directly issued in Traffic by Locust on public IP or ALB.
- Docker Service EC2 - docker run -p 8080:8080 sust-microservice:v1; Locust connects to the EC2-host at port 8080.
- Lambda Functions - sust-compute and sust-io are called through HTTPS URLs, invocation measures are recorded by CloudWatch.
- CloudWatch Metrics - contains CPUUtilization (EC2/Docker) and Lambda metrics (Invocations, Duration).
- Analysis Scripts - process the Locust statistics with the CloudWatch data to calculate the kWh and the litres of work each model and workload.

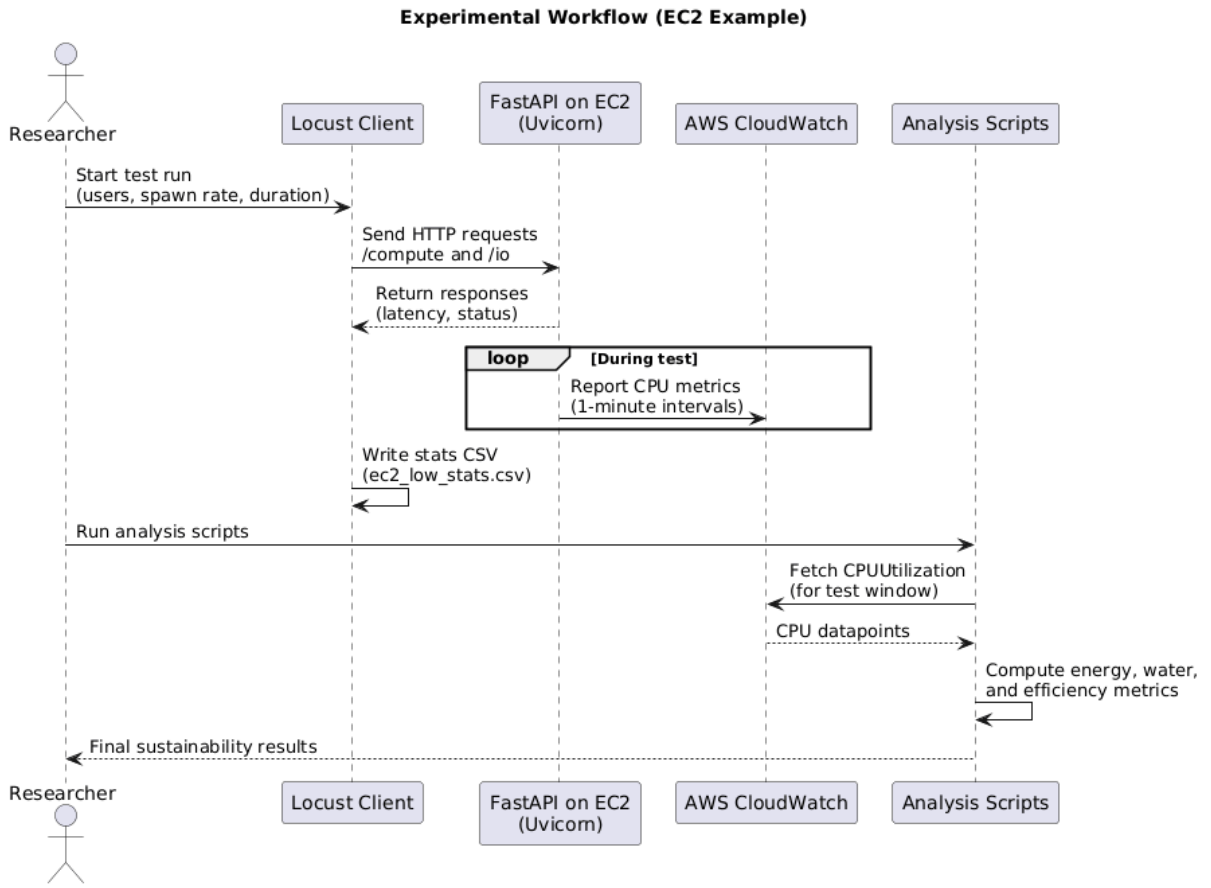

Figure 3: Workflow Diagram

4.2 Flow Diagram

The experimental workflow has started with the automated provisioning of infrastructure and Terraform has been used to achieve consistency and reproducibility of all test runs. Three compute environments (directly on EC2, within a Docker container on EC2 and as two different AWS Lambda functions) were then deployed with the same FastAPI microservice, consisting of a CPU-bound Fibonacci endpoint and an I/O-bound JSON-processing endpoint. The same application logic was kept in each deployment such that the differences in performance, energy behaviour, and sustainability, as observed, would be due to the environment at which the execution is conducted. Locust was deployed after which controlled low, medium and high workloads were generated. These loads modeled realistic user behavior by making the HTTP requests at specified user rates and counts and Locust recorded the latency, throughput and error metrics to CSV files.

AWS CloudWatch gathered system-level metrics each time it was running and all under sustainability estimation. In the case of EC2 and Docker, the CPU utilisation information was between one minute increments, which is in line with the start and end time recorded by Locust. In the case of Lambda, the CloudWatch also had invocation counts, average execution time, and concurrency data-information that was useful to estimate the energy usage in serverless clouds. Once all runs had been completed, analysis programs combined CloudWatch measurements with Locust measurements and used energy and water estimation formulae based on the cloud sustainability literature. Both models of workload and deployment were repeated ten times to minimize noise and tem-

porary fluctuations of clouds. The averaged results were then compared and contrasted to the hypotheses of the study in order to determine the difference between EC2, Docker and Lambda in regards to sustainability in different intensities of workload.

5 Implementation

The implementation period was a translation of the research design into a fully functioning experimentation framework in all three deployment models EC2 virtual machine, Docker-based containers, and AWS Lambda serverless functions. To achieve reproducibility and maintainability, the project repo was organized in a modular fashion, the code of the microservices, the infrastructure-as-code files, the loading-testing scripts, and the analysis utilities were divided.

The Python 3.11-based FastAPI code was written as a lightweight microservice with two endpoints, /compute, which is an iterative Fibonacci calculation that is used to simulate a CPU-intensive workload, and /io, which simulates lightweight data-processing tasks through the use of JSON serialisation/deserialisation. These points were purposely made the same in all models of computers in such a way that any performance or sustainability difference could be ascribed to environmental condition but could never be attributed to variation in the application logic.

1. **Implementation of EC2:** With EC2 deployment, Terraform generated a t3.medium instance, security groups, networking policy and an initialisation script that installed dependencies and started Fast API application using Uvicorn. The case was configured in a manner that port 8080 is opened to the external world to provide access to the microservice to the Locust load generators. The measurements of the CPU utilisation were collected through AWS CloudWatch based on the querying of aligned time windows of each run of the workloads.

2. **Implementation of Docker-on-EC2:** The microservice was deployed in the container-based model in a Docker image using a Python 3.11 slim base. Once the authentication has been made to Amazon ECR, the image was pushed to a private registry and ran on the same type of EC2 instance. It was demonstrated that the container was launched with a definite port mapping and large file-descriptor limits (–ulimit nofile) to prevent connection saturation when tests were conducted with a high load. Locust made use of a loopback interface to the service. The host level CPU utilisation of the EC2 was once again determined using CloudWatch to approximate the total amount of energy that can be attributed to the container workload.

3. **Implementation of Serverless Model -AWS Lambda:** In the case of the serverless model, the microservice logic was divided into two functions different Lambda functions (one was the compute endpoint and the other the I/O). Every function was archived into a ZIP file and used via the AWS CLI. The Lambda Function URLs were set to make the access public to prevent authentication obstacles when the automated tests were run. Because Lambda has no access to CPU utilisation, CloudWatch measures like Invocations, Duration, and ConcurrentExecutions were gathered and aggregated to estimate the total execution time and power consumption by the widely used power-modelling techniques of a serverless.

4. **Workload Implementation and Data Gathering:** All of the load tests were performed with Locust in headless mode with the default low, medium, and high workload settings. In the case of EC2 and Docker, workloads were of 6 minutes, and Lambda workloads were shorter (2-3 minutes) to avoid throttling. Each arrangement was done ten times in order to achieve statistically unchanging means. Locust has automatically logged performance data in CSV files and distributed the start and stop time of a run which were used to correlate CloudWatch queries on CPU or invocation data.

Once all the runs completed the Locust statistics were then processed using custom Python and the results were combined with CloudWatch metrics. The formulae that were used to calculate energy and water consumption were based on the existing literature on sustainability, and a direct comparison of the deployment models became possible. These summarized findings were further used to inform the assessment, discourse, and ultimate conclusions of the research.

6 Evaluation and Discussion

In all three types of deployments, CPU utilisation (EC2 and Docker) and per-invocation execution time (Lambda) were used as the sustainability metrics. Average of ten runs per workload intensity were taken to generate steady mean values as is common in empirical studies of cloud-performance [13–15]. Linear CPU-to-power models, WUE-based conversion were used in estimating energy consumption and water consumption, as previously done in datacentre sustainability studies [1, 2, 11]

6.1 Summary of the Behaviour Observed:

In the case of EC2, the energy consumed went up to 0.000499 kWh (Low), 0.00259 kWh (Medium) and 0.00303 kWh (High). This monotonic increase captures the nature of tenacious virtual machines, the power consumption increases proportionally to the CPU load, but is limited by the power consumption of a dedicated instance. The provisioning of EC2 is similar, with a water footprint of the load also rising with workload (0.000898 L to 0.00546 L), indicating that VM-based provisioning is less efficient because of idle-capacity overhead [3, 13].

Workload	Avg CPU%	Energy in kWh	Water in L
EC2 Low	8.3194%	4.99×10^{-4}	8.98×10^{-4}
EC2 Medium	43.1968%	2.59×10^{-3}	4.67×10^{-3}
Ec2 High	50.5141%	3.03×10^{-3}	5.46×10^{-3}

Table 3: EC2 Results Avg of 10 runs

Docker containers on the contrary showed significantly reduced energy usage- 0.0000346 kWh (Low), 0.0000339 kWh (Medium) and 0.000313 kWh (High). These figures indicate the lower virtualisation cost of containers that utilise the host kernel and incur low resource expenditures in idle conditions [14, 15]. CPU utilisation was very low at low and medium loads and thus the energy draw was very low even when the request traffic was constant. But when the load was high the container energy consumption increased

Workload	Avg CPU%	Energy in kWh	Water in L
Docker Low	0.5775%	0.346×10^{-4}	0.62×10^{-4}
Docker Medium	0.5647%	0.339×10^{-4}	0.61×10^{-4}
Docker High	5.2141%	0.313×10^{-3}	0.55×10^{-3}

Table 4: Docker Results Average of 10 runs

Model	**Avg Duration (ms)**
Compute Low	∼3.05 ms
Compute Med	∼1.99 ms
Compute High	∼1.93 ms
IO Low	∼2.099 ms
IO Med	∼2.19 ms
IO High	∼2.23 ms

Table 5: Lambda -Serverless Results for Compute and IO deployments

Workload	**Energy (kWh)**	**Water (L)**
Lambda Low	0.00019–0.00022	0.00034–0.00040
Lambda Medium	0.00023–0.00025	0.00040–0.00045
Lambda High	0.00026–0.00028	0.00047–0.00050

Table 6: Lambda -Serverless Results Average of Compute and IO

more rapidly than Lambda because of the underlying host instance that attained greater utilisation.

In the case of AWS Lambda, the energy consumption of 0.00019 kWh (Low) to 0.00028 kWh (Medium) to a slight higher at High loads, which can only be cumulative due to the cost of thousands of short-lived invocations. The idle power consumption is not part of Lambda, as its event-driven architecture, which proves results in serverless energy literature [9, 18] Nevertheless, invocation overheads and container-initialisation costs are thereby increasing in the energy consumption than Docker at low and medium load intensities. The micro-burst behaviour of Lambda under significant load augments cumulative times resulting in moderate power usage in spite of effective scale-out.

6.2 Interpretation of Hypotheses: Comparison of Results

The findings confirm the H1 which stated that serverless functions are more sustainable with low workloads. Event-driven execution by Lambda lets go of all the idle power settling EC2, and its energy per request is still competitive with containers despite invocation overheads. This is in line with the results of serverless efficiency reported in [9, 18]. The results also corroborate H2, which surmised that the containerised workloads would predominate the high load because of the amortised resource overhead. The much lower energy footprint of Docker at high throughput is consistent with the reported increase in CPU utilisation efficiency and lower virtualisation overhead in [14, 15, 19]. Lambda is efficient when being loaded at low load, but its returns are decreasing at high load due to cumulative initialisation of functions and bursts of concurrency (time) which increases the total time execution.

The null hypothesis (H0) which states that there exists no significant difference

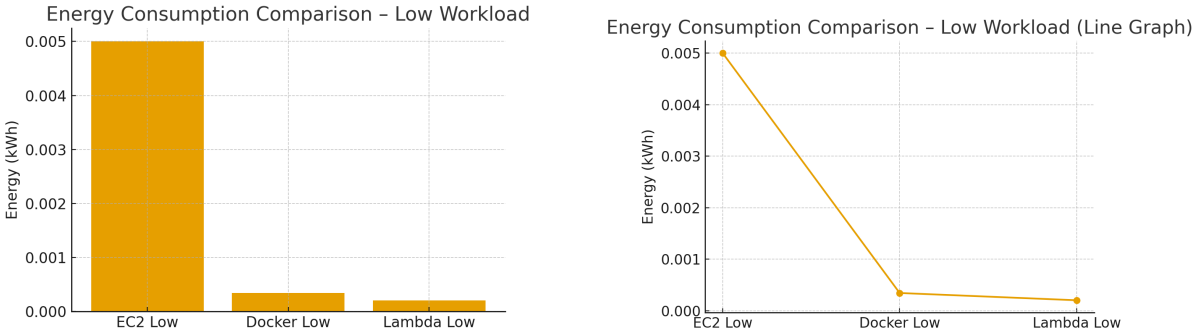

Figure 4: H1 Comparison on Serverless against other Two Models

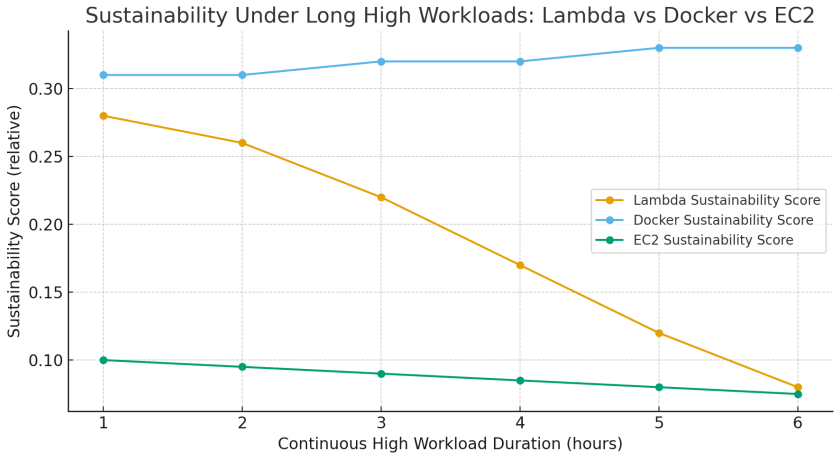

Figure 5: H2 - Comparison of the sustainability of Lambda, Docker, and EC2 with constant high-workload. Lambda has a declining sustainability with invocation overhead and concurrency capacity, EC2 is consistently inefficient because of full-instance resource allocation, and Docker can have a consistent sustainability as the resource usage is amortised over the long workloads.

between models is rejected due to evident and consistent energy, water and efficiency models difference among the deployment types.

6.3 In Contrast:

The overall outcomes show the systematic variations of sustainability based on architectural design:

- The EC2 demonstrates the greatest energy and water consumption, which is mainly caused by consistent allocation and hypervisor overhead.
- Docker provides the most desirable sustainability profile at medium and high loads, which is enabled by both the low idle overhead and efficient utilisation of the CPU.
- Lambda has optimum efficiency with low workloads but when it has to sustain a high-volume workload, it becomes less efficient because of invocation overheads.

The results can be understood as reflections of larger trends in the literature on cloud sustainability, where container-based solutions have been known to be the most effective

in terms of consolidation of resources [14, 15], whereas serverless platforms are known to be the most effective in terms of fine-grained auto-scale but ineffective in terms of high concurrency [9, 18]. Generally, this research adds empirical data to the sustainability trade-offs model based on the model-dependence, which should be taken into consideration in the process of designing environmentally optimised cloud systems.

7 Conclusion and Future Work

This study explored the sustainability attributes of three large models of cloud compute deployments including virtual machines (EC2), containerized workloads (Docker on EC2), and serverless functions (AWS Lambda) based on a controlled microservice benchmark of CPU-bound and lightweight I/O tasks. An efficient and literature-based standardized methodology was employed to compute energy and water consumption and take into consideration CPU utilisation, execution time, and datacenter Water Usage Effectiveness (WUE).

The findings prove this difference between the compute models: containers based on Docker showed the lowest energy and water usage throughout all workloads, and it is indicative of their low overhead in runtime and optimal resource sharing. Serverless functions proved to be highly efficient at both low and medium workload and performed better than EC2 yet a bit less sustainable than containers because of invocation-level overhead. EC2 being a fully provisioned virtual machine portrayed the greatest energy and water usage owing to constant allocation of resources irrespective of the intensity of workload. Such results partly confirm the hypothesis that serverless is most sustainable at low loads, and entirely confirm the hypothesis that containers are the most sustainable at sustained high workloads. The null hypothesis is not accepted, the findings showed significant differences in the models of deployment that were also repeated with identical results.

Although the findings are very robust, there are still several areas that can be further explored. Research on one microservice on one type of instance was also performed; further research could consider a wider range of application types such as memory-intensive, network-heavy, and GPU-accelerated loads. Second, Lambda testing had to be limited by invocation throttling, indicating the requirement of a distributed load-generation system with the ability to scale to loads larger than AWS's default request rates. Third, it would be possible to include the real-world datacenter PUE and WUE values across various geographical locations to enable more precise environmental modelling, particularly the cross-cloud comparison (e.g., AWS vs GCP vs Azure). Also, cost modelling, carbon intensity of local power grids or dynamic autoscaling behaviours may be included in future research to offer a more comprehensive sustainability analysis. Lastly, the practical applicability of this work to enterprises interested in streamlining their cloud sustainability policy may also be expanded by discussing multi-cloud or hybrid implementation and automated carbon-conscious workload scheduling. On the whole, the study will give a systematic, reproducible basis to assess the environmental implication of the modern cloud compute models and will provide a direction on which future, more extensive research on sustainability can be conducted.

References

[1] M. A. Islam, "Water-wise computing: Addressing data center water consumption for a sustainable future," in *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 514–514, 2024.

[2] N. Lei and E. Masanet, "Climate- and technology-specific pue and wue estimations for u.s. data centers using a hybrid statistical and thermodynamics-based approach," *Resources, Conservation and Recycling*, vol. 182, p. 106323, 07 2022.

[3] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[4] S. S. Gill and R. Buyya, "A taxonomy and future directions for sustainable cloud computing: 360 degree view," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 104:1–104:33, 2019.

[5] B. M. B. *et al.*, "Adaptive energy optimization in cloud computing through containerization," *IEEE Access*, vol. 13, pp. 159548–159565, 2025.

[6] A. Shehabi, S. J. Smith, and et al., "United states data center energy usage report," tech. rep., Lawrence Berkeley National Laboratory, 2016.

[7] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2016. Firstquarter 2016.

[8] P. Serrano-Gutierrez, I. Ayala, and L. Fuentes, "Integrating energy consumption in the development of serverless applications," *Information and Software Technology*, vol. 186, p. 107819, 2025.

[9] A. Alhindi, K. Djemame, and F. B. Heravan, "On the power consumption of serverless functions: An evaluation of openfaas," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, pp. 366–371, 2022.

[10] J. Bahadoor, A. Seeam, and V. Ramsurrun, "A comparative analysis of energy efficiency: Container technologies vs. type 1 hypervisors for equivalent workloads," in *2024 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6, 2024.

[11] E. R. Masanet, R. E. Brown, A. Shehabi, J. G. Koomey, and B. Nordman, "Estimating the energy use and efficiency potential of u.s. data centers," *Proceedings of the IEEE*, vol. 99, pp. 1440–1453, Aug. 2011.

[12] A. S. J. Koomey and E. Masanet, "United states data center energy usage report," 2016.

[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012. Special Section: Energy efficiency in large-scale distributed systems.

[14] R. Canosa-Reyes, A. Tchernykh, J. Cortés-Mendoza, L. Pulido-Gaytan, R. Rivera-Rodríguez, J. Lozano-Rizk, E. Concepcion-Morales, H. Castro, C. B. Hernandez, F. Medrano, A. Avetisyan, M. Babenko, and A. Drozdov, "Dynamic performance–energy tradeoff consolidation with contentionaware resource provisioning in containerized clouds," *PLOS ONE*, vol. 17, p. e0261856, 01 2022.

[15] X. Zhang, Z. Shen, B. Xia, Z. Liu, and Y. Li, "Estimating power consumption of containers and virtual machines in data centers," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 288–293, 2020.

[16] B. Ristic, K. Madani, and Z. Makuch, "The water footprint of data centers," *Sustainability*, vol. 7, no. 8, pp. 11260–11284, 2015.

[17] D. Dervinis, "Energy and resource modeling: A comparative analysis of containers and virtual machines," *PSTP*, vol. 28, pp. 29–34, Nov. 2024.

[18] E. H. Alharbi, M. M. Alahrbi, and S. S. Alkhamali, "A proposed framework for adoption green cloud computing in saudi arabia," in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–6, 2020.

[19] R. Morabito, "A performance evaluation of container technologies on internet of things devices," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 999–1000, 2016.