WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Real-Time Carbon Footprint Estimator for Cloud Workloads A Multi-Cloud Framework for Cost, Carbon, and Performance Evaluation

[1] Bommanaboina Tirumala Ganesh, [2] Allada Sujith Kumar, [3] Devadasu Akhil, [4] Gurugubelli Chandini, [5] Yalla Siva Srikanth, [6] Suraj Soren

[1-5] UG Student, [6] Assistant Professor, Department of CSE, GMRIT Deemed to be University, Rajam, Andhra Pradesh, India

ABSTRACT :

Cloud computing provides scalable infrastructure for modern applications, but its operation requires electricity and contributes to greenhouse-gas emissions. Cloud infrastructure is commonly compared using price and computing capacity, while environmental impact is often evaluated separately. This paper presents a Real-Time Carbon Footprint Estimator for Cloud Workloads, a multi-cloud decision-support framework for comparing eligible virtual-machine configurations across Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. Users specify workload requirements, including virtual CPU capacity, memory, storage, operating duration, and region, and assign preference weights to cost, estimated operational carbon emissions, and computational performance. A prepared dataset combines instance specifications, cloud pricing, regional electricity carbon intensity, power-consumption assumptions, data-center Power Usage Effectiveness (PUE), and processor benchmark information. The estimation engine calculates monthly infrastructure cost and operational carbon emissions, derives a benchmark-based performance indicator, and combines normalized criteria through a weighted scoring model. The configuration with the highest score is recommended according to the user's priorities. The proposed implementation uses a React and TypeScript interface, AWS API Gateway, AWS Lambda, and Amazon DynamoDB for analysis-history storage. Here, real-time refers to immediate on-demand calculation using a prepared dataset snapshot; it does not mean continuous physical measurement of cloud-server electricity consumption. Verified numerical experimental results must be added from the implemented system before publication.

IndexTerms - Cloud computing, carbon footprint, carbon intensity, sustainable computing, multi-cloud optimization, workload estimation, Power Usage Effectiveness, cloud cost.

I. INTRODUCTION

Cloud computing has become a fundamental model for delivering computing, storage, and networking resources. Organizations use cloud platforms to host web applications, databases, analytics systems, and other computational workloads. Infrastructure-as-a-Service offerings allow users to select virtual-machine configurations according to processing, memory, storage, and geographical requirements. Major providers such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure offer a wide range of instance families and pricing models.

The flexibility of cloud computing does not eliminate the environmental cost of computation. Physical servers, storage devices, networking equipment, and cooling systems consume electricity in data centers. The greenhouse-gas emissions associated with this electricity depend on energy demand, facility efficiency, and the carbon intensity of the electricity supply. Consequently, two configurations that satisfy similar workload requirements may have different estimated cost and carbon outcomes.

Computational carbon-estimation research has demonstrated that energy consumption and regional electricity carbon intensity can be combined to estimate emissions [1]. Carbon-aware computing research has also shown that the timing and location of workloads can influence electricity-related emissions [2]. The Software Carbon Intensity (SCI) specification provides a framework for expressing software-related emissions in relation to a functional unit and considers operational and embodied emissions [3]. These approaches motivate the inclusion of environmental indicators in software and infrastructure decisions.

A cloud customer choosing an infrastructure configuration may need to balance several competing objectives. A low-cost instance may not provide the highest benchmark performance, while a configuration in a region with lower

carbon intensity may have different pricing or resource availability. A single metric therefore cannot represent every user's priorities. A comparative decision-support tool should expose the assumptions behind its estimates and allow users to specify the relative importance of cost, carbon, and performance.

A. Problem Statement

Cloud infrastructure selection is frequently performed by comparing provider-specific prices and resource specifications. Carbon-emissions information and processor-performance data may be maintained separately, making it difficult to evaluate candidate configurations through one consistent process. This work addresses the design of a multi-cloud estimation framework that compares eligible configurations using estimated monthly cost, estimated operational carbon emissions, and benchmark-based performance indicators, while allowing users to control the trade-off among these criteria.

B. Objectives

- Develop a common comparison workflow for eligible AWS, GCP, and Azure virtual-machine configurations.
- Estimate monthly compute, storage, network-transfer, and applicable request costs from documented pricing inputs.
- Estimate operational electricity-related emissions using modeled IT energy, PUE, and regional carbon intensity.
- Use benchmark information as a comparative performance indicator while documenting its limitations.
- Normalize the three evaluation criteria and combine them using user-defined weights.
- Present recommendations together with calculation assumptions, data provenance, and analysis history.

C. Contributions

The proposed contribution is an integrated, user-configurable decision workflow. It combines multi-cloud candidate filtering, cost estimation, operational carbon modeling, benchmark-based performance comparison, and preference-weighted ranking. It also emphasizes data provenance and disclosure of assumptions so that users can distinguish provider-published prices, benchmark observations, and modeled carbon estimates.

II. RESEARCH METHODOLOGY

The proposed methodology consists of workload requirement collection, dataset preparation, candidate configuration selection, metric estimation, normalization, and weighted ranking. The system is designed to provide an immediate response after a user submits a workload request.

A. System Architecture

The frontend is designed using React, TypeScript, and Vite, with Tailwind CSS for styling and Recharts for visual comparison. The frontend sends a request to AWS API Gateway. API Gateway invokes an AWS Lambda function that validates the input, reads the prepared dataset, filters candidate configurations, executes the estimation formulas, and calculates the weighted ranking. Amazon DynamoDB stores analysis records for later retrieval. A separate offline data-preparation pipeline cleans and combines provider and reference datasets into a structured JSON snapshot.

The system workflow is: User input → React frontend → API Gateway → Lambda validation and calculation engine → candidate filtering → cost, carbon, and performance estimation → normalization and weighted ranking → recommendation displayed in the frontend. Analysis history is stored in DynamoDB.

B. Workload Requirements and Input Validation

The user provides the workload category, required virtual CPUs, RAM, storage capacity, geographical region, expected operating duration, and preference weights for cost, carbon, and performance. Inputs are validated before processing. Candidate instances are eligible only when they meet the resource and region constraints represented in the prepared dataset.

Let w_c, w_e, and w_p denote the weights assigned to cost, emissions, and performance, respectively. The weights are non-negative and satisfy:

$$w_c + w_e + w_p = 1 \quad (1)$$

C. Dataset Preparation

The offline data pipeline combines cloud instance specifications, compute and storage pricing, network-transfer pricing, regional electricity carbon-intensity values, PUE assumptions, server power assumptions, and CPU benchmark information. Raw data are validated, cleaned, and combined into a structured dataset in JSON format. The dataset acts as a prepared snapshot consumed by the backend calculation engine.

Each dataset record should retain its source, retrieval date, region, currency, billing unit, benchmark metric, and any conversion or estimation assumptions. Provider-published values must be distinguished from modeled or third-party values. Where server power or PUE is not available at the required granularity, any proxy must be identified as an assumption.

D. Cloud Infrastructure Cost Estimation

The estimated monthly infrastructure cost is composed of compute, storage, network-transfer, and applicable API request charges:

$$C_{total} = C_{compute} + C_{storage} + C_{network} + C_{API} \quad (2)$$

For an hourly instance price P_h and billable operating duration H, compute cost is:

$$C_{compute} = P_h \times H \quad (3)$$

For storage capacity S and monthly unit price P_s:

$$C_{storage} = S \times P_s \quad (4)$$

For billable data transfer D and unit transfer price P_d:

$$C_{network} = D \times P_d \quad (5)$$

The request component is included only where applicable. The estimate must disclose whether taxes, discounts, free tiers, licenses, support charges, committed-use pricing, and provider-specific billing conditions are included or excluded. A 720-hour month represents a continuous 30-day operating assumption.

E. Carbon Footprint Estimation

The carbon model estimates operational electricity-related emissions. It does not directly measure the power draw of the user's running workload. Where idle and maximum server power assumptions and average utilization U are available, average IT power is approximated as:

$$P_{avg} = P_{idle} + (P_{max} - P_{idle}) \times U \quad (6)$$

For an operating duration H hours, estimated IT energy consumption is:

$$E_{IT} = (P_{avg} \times H) / 1000 \quad (7)$$

where E_IT is in kWh and power is in watts. Facility energy is estimated by applying Power Usage Effectiveness:

$$E_{facility} = E_{IT} \times PUE \quad (8)$$

If regional carbon intensity CI is expressed in gCO_2e/kWh, estimated operational emissions in kilograms are:

$$CO_2e_{kg} = (E_{facility} \times CI) / 1000 \quad (9)$$

The result depends on the power assumptions, utilization, PUE, and regional carbon-intensity dataset. If the energy input already represents facility-level energy, PUE must not be applied again. If carbon intensity is time-varying, a time-aligned factor is preferable; otherwise, the result represents the selected dataset snapshot or regional average.

F. Performance Evaluation

The system uses available processor benchmark information, particularly SPEC CPU2017 rate benchmark data where comparable records are available [4]. Benchmark values are used as an indicative CPU-performance measure. A benchmark metric must be consistent across candidates; SPECrate and SPECspeed metrics must not be treated as interchangeable. CPU benchmark scores do not fully represent application performance because memory, storage, networking, and workload characteristics also influence execution.

For a candidate benchmark value B_i, with minimum B_{min} and maximum B_{max} among the eligible candidates, the normalized performance score is:

$$P_{score,i} = 100 \times (B_i - B_{min}) / (B_{max} - B_{min}) \quad (10)$$

A higher score indicates a higher value for the selected benchmark metric. If all candidates have identical values, equal scores are assigned to avoid division by zero. Missing benchmark information must be disclosed rather than replaced by an invented value.

G. Normalization and Weighted Scoring

Cost and carbon are criteria for which lower values are preferred. Their normalized scores are calculated as:

$$C_{score,i} = 100 \times (C_{max} - C_i) / (C_{max} - C_{min}) \quad (11)$$

$$E_{score,i} = 100 \times (E_{max} - E_i) / (E_{max} - E_{min}) \quad (12)$$

The final weighted score for candidate i is:

$$S_i = w_c C_{score,i} + w_e E_{score,i} + w_p P_{score,i} \quad (13)$$

The recommended configuration is the eligible candidate with the maximum score:

$$i^* = \arg\max_i S_i \quad (14)$$

The recommendation is preference-dependent. Changing the user weights can change the ranking. The interface should display the individual metric values and normalized scores alongside the aggregate score.

H. Implementation Technologies

Table 1. implementation technologies

Component	Technology	Purpose
Frontend	React, TypeScript, Vite	User interface
Styling and charts	Tailwind CSS, Recharts	Presentation and comparison charts
API layer	AWS API Gateway	Request handling
Backend processing	AWS Lambda	Validation and calculation
Database	Amazon DynamoDB	Analysis history
Validation	Zod	Input validation
Data preparation	Offline pipeline	Dataset cleaning and integration
Data format	JSON	Prepared cloud dataset

I. Analysis History

Amazon DynamoDB is used to store previous analysis records, including workload requirements, candidate metrics, estimated cost, estimated carbon emissions, performance scores, and the final recommendation. A Time-to-Live (TTL) policy can expire records after a configured retention period. A temporary undo-delete interaction is provided as a user-interface feature.

III. RESULTS AND DISCUSSION

For demonstrating the calculation workflow, Tables 2–5 contain illustrative sample values for a common 2-vCPU, 8-GB RAM, 100-GB storage workload over 730 hours. These values are synthetic assumptions and are not live provider quotations, measured server power, verified regional carbon-intensity observations, or benchmark test results. They demonstrate how the proposed estimator calculates and ranks candidates; they must be replaced with dated provider pricing records, documented energy and carbon datasets, and actual benchmark records before the manuscript is submitted as an empirical evaluation.

A. Experimental Configuration

The evaluation should apply the same workload requirements to suitable candidates from AWS, GCP, and Azure. The paper should record the exact instance identifiers, region, pricing model, currency, data retrieval date, carbon-intensity source and period, PUE assumption, power model, utilization,

benchmark metric, and dataset version.

Table 2. Experimental configuration (illustrative assumptions)

Parameter	Illustrative value / assumption
Workload type	General-purpose web application
vCPU / RAM / storage	2 vCPU / 8 GB RAM / 100 GB SSD
Operating duration	730 hours (one 30.4-day month), 100% provisioned time
Regions	AWS ap-south-1 (Mumbai); GCP asia-south1 (Mumbai); Azure Central India
Pricing source and date	Illustrative hourly rates in USD; assumed for demonstration, not live quotations
Carbon-intensity source	Illustrative regional grid factors: 650 gCO2e/kWh (Mumbai); 680 gCO2e/kWh (Central India)
PUE and power assumptions	AWS: 1.25 / 22 W; GCP: 1.20 / 20 W; Azure: 1.30 / 24 W (modeled assumptions)
Benchmark metric	Illustrative relative CPU index (82, 85, 80); not measured SPEC CPU2017 results
Dataset version	Demonstration dataset v1.0; synthetic values prepared for sample calculation

B. Monthly Cost Comparison

Table 3. Monthly cloud infrastructure cost (illustrative USD)

Provider / instance	Compute	Storage	Network / API	Total monthly cost
AWS t3.large	68.91	8.00	3.00	79.91
GCP e2-standard-2	48.91	10.00	3.00	61.91
Azure D2as_v5	70.08	10.00	3.00	83.08

In this illustrative scenario, GCP has the lowest estimated monthly cost (USD 61.91), followed by AWS (USD 79.91) and Azure (USD 83.08). The comparison uses assumed component charges and excludes taxes, discounts, support plans, licensing, and unmodeled network charges.

C. Operational Carbon Comparison

Table 4. Estimated operational carbon emissions (illustrative)

Provider / instance	IT energy (kWh)	PUE	Carbon intensity (gCO2e/kWh)	CO2e (kg)
AWS t3.large	16.06	1.25	650	13.05
GCP e2-standard-2	14.60	1.20	650	11.39
Azure D2as_v5	17.52	1.30	680	15.48

Under the illustrative power, PUE, and grid-intensity assumptions, GCP has the lowest estimated operational emissions (11.39 kg CO2e), followed by AWS (13.05 kg CO2e) and Azure (15.48 kg CO2e). These values are scenario outputs, not provider-reported emissions.

D. Performance and Weighted Ranking

Table 5. Normalized metrics and weighted ranking (illustrative)

Provider	Benchmark value	Cost score	Carbon score	Performance score	Final score
AWS t3.large	82	15.0	59.4	40.0	37.8
GCP e2-standard-2	85	100.0	100.0	100.0	100.0
Azure D2as_v5	80	0.0	0.0	0.0	0.0

The illustrative ranking uses weights of 0.40 for cost, 0.40 for carbon, and 0.20 for performance. The weights sum to 1.00. Under these assumptions, GCP receives the highest weighted score. The benchmark index is synthetic and is included only to demonstrate normalization; actual SPEC results must be inserted for a defensible performance comparison.

E. Validation and Limitations

Cost calculations should be checked against provider pricing records or calculators using the same service, region, billing unit, and assumptions. Carbon calculations should be independently recomputed from the recorded energy, PUE, and carbon-intensity inputs. Benchmark records should be checked against their original source and exact metric. The ranking implementation should be tested for invalid weights, missing values, identical candidate values, and zero denominators.

The main limitations are that server power may be modeled rather than measured; provider-specific PUE and hardware allocation may not be available at the required granularity; regional carbon intensity can vary over time; CPU benchmark scores may not predict application-level performance; pricing estimates may exclude taxes, discounts, support, licenses, and provider-specific billing conditions; and embodied hardware emissions are outside the

current operational model.

IV. CONCLUSION

This paper presented a Real-Time Carbon Footprint Estimator for Cloud Workloads, a multi-cloud decision-support framework for comparing eligible infrastructure configurations using estimated monthly cost, operational carbon emissions, and benchmark-based computational performance. The proposed system accepts workload requirements and user-defined preference weights, processes a prepared dataset containing cloud specifications and reference information, calculates the three evaluation criteria, normalizes the results, and ranks candidates through a weighted scoring model.

The framework brings financial, environmental, and performance considerations into one comparison workflow. Its real-time characteristic refers to on-demand execution of the estimation process; the freshness and accuracy of the output remain dependent on the prepared dataset and modeling assumptions. The current method estimates operational electricity-related emissions and should not be interpreted as direct measurement of a user's cloud workload or a complete life-cycle carbon assessment.

Future work includes integrating supported live pricing and carbon-intensity sources, updating regional electricity factors at finer temporal resolution, calibrating power models against measured workloads, incorporating workload-specific benchmarks, extending the model to GPU and storage-intensive workloads, estimating embodied hardware emissions, and validating recommendations against deployed cloud workloads. Sensitivity analysis and uncertainty intervals can further clarify the effect of data quality and modeling assumptions.

REFERENCES

[1] L. Lannelongue, J. Grealey, and M. Inouye, “Green Algorithms: Quantifying the Carbon Footprint of Computation,” Advanced Science, vol. 8, no. 12, Art. no. 2100707, 2021, doi: 10.1002/advs.202100707.

[2] Google, “Carbon-Intelligent Computing,” Google Research. [Online]. Available: https://research.google/pubs/carbon-intelligent-computing/. Verify the exact publication metadata and access date before submission.

[3] Green Software Foundation, “Software Carbon Intensity (SCI) Specification.” [Online]. Available: https://sci.greensoftware.foundation/. Verify the version cited and access date before submission.

[4] Standard Performance Evaluation Corporation (SPEC), “SPEC CPU2017 Documentation.” [Online]. Available: https://www.spec.org/cpu2017/Docs/. Use the exact benchmark metric and result record in the final manuscript.