Evaluating Energy Efficiency of Clustering Algorithms for VM Task Allocation in Cloud

Dishant Patel, Karishma Karamali Vadsaria, Himanshu Sharma
Khoury College of Computer Sciences
Northeastern University
Vancouver, Canada
vancouver@northeastern.edu

Michal Aibin
Department of Computing
British Columbia Institute of Technology
Vancouver, Canada
maibin@bcit.ca

***Abstract*—One of the critical issues in cloud computing is the poor allocation of resources among inappropriately sized virtual machines (VMs), resulting in significant power consumption. Although numerous strategies for improving energy efficiency have been proposed, such as integer programming, clustering algorithms, and deep reinforcement learning, little research has been done to compare different clustering algorithms. This study compares the energy efficiency of K-means, the Gaussian Mixture Model, and Spectral clustering algorithms by grouping workloads based on computational requirements and selecting appropriately sized virtual machines. The results are assessed using four simulation scenarios in real-world cloud computing systems, showing that the Spectral clustering algorithm outperforms others.**

***Index Terms*—green cloud computing, energy efficiency, task allocation, clustering algorithms**

I. Introduction

Green Cloud Computing aims to reduce the energy consumption and carbon footprint of cloud computing systems while incorporating renewable energy sources. The demand for cloud services experienced an exponential growth of 4800% from 2008 to 2019 [1], leading to the rapid expansion of data centers, servers, and IT infrastructure. This unprecedented growth has significantly increased energy consumption and carbon emissions, particularly in regions with high demand for Information and Communications Technology (ICT) services [2]–[4]. The environmental impact affects multiple stakeholders, including cloud service providers, data center operators, IT companies and end users [5]. Consequently, there is a need for sustainable and eco-friendly solutions to minimize the environmental footprint of cloud computing.

Various techniques have been developed to improve energy efficiency in cloud computing, including optimized Virtual Machine (VM) placement, VM consolidation, intelligent task allocation [6], and clustering algorithms for efficient VM task distribution [7]. Among these approaches, clustering algorithms demonstrate particular promise by grouping similar VMs to optimize data center resource utilization [8], reduce the number of active servers [9], conserve energy, and lower carbon emissions [10]. Although there are multiple methods to achieve energy-efficient cloud task scheduling, including integer programming [6], deep reinforcement learning [11], and clustering algorithms [12], [13], a comprehensive comparison of these methods with respect to energy efficiency remains lacking. Clustering algorithms offer distinct advantages through their flexibility, adaptability to different data distributions, and interpretability through well-defined clusters. Unlike integer programming, which suffers from exponential computational complexity for large-scale problems and limited flexibility with constraints, clustering provides a practical and scalable approach to resource allocation optimization. Our study addresses this research gap by evaluating the energy efficiency of K-means, GMM, and spectral clustering for task scheduling using real-world cloud environments, moving beyond the limitations of simulation tools like CloudSim.

The primary objective of this research is to identify the most energy-efficient clustering algorithm for assigning tasks to VMs in cloud environments, thereby reducing the carbon footprint of cloud operations. To achieve this objective, we formulate and address the following research questions:

RQ1: How do different clustering algorithms (K-means, GMM, Spectral) compare in terms of carbon emissions when allocating tasks to VMs in real cloud environments?

RQ2: What is the quantitative trade-off between energy efficiency and request failure rates for each clustering algorithm in different workload patterns?

RQ3: How does the configuration of the VM instance (minimum, maximum, shared, dedicated) affect the carbon footprint and performance of each clustering approach?

RQ4: Which clustering algorithm provides the optimal balance between environmental impact and service reliability for production cloud deployments?

To answer these research questions, we simulate various workloads using Amazon Web Services (AWS) VMs with different CPU, memory, and disk I/O requirements, measuring carbon emissions for each clustering algorithm using a comprehensive evaluation metric.

The main contributions of this paper are:

- A novel evaluation framework for measuring carbon emissions of clustering-based VM allocation in real AWS cloud environments, moving beyond simulation-based studies to provide practical, deployable insights.
- First systematic comparison of three fundamental clustering paradigms (centroid-based, model-based, and graph-based) for cloud task allocation, revealing that spectral clustering reduces carbon emissions by up to 76% compared to K-means in shared instance scenarios.
- Comprehensive analysis of the trade-off between energy efficiency and service reliability, demonstrating that while spectral clustering achieves the lowest emissions, it ex-

hibits a 0.675% failure rate compared to K-means' near-zero failure rate.
- Development of a carbon emission metric (Equation 1) adapted explicitly for clustering-based allocation that accounts for VM utilization patterns, Power Usage Effectiveness (PUE), and regional emission factors.
- Practical deployment guidelines and real-world testing with 100,000 requests across four distinct scenarios, providing actionable insights for cloud providers seeking to reduce their environmental impact.

These contributions advance the state-of-the-art by providing empirical evidence from production environments rather than simulations, offering practitioners a data-driven approach to selecting clustering algorithms based on their specific environmental and performance requirements.

II. Problem Statement

A. Architecture

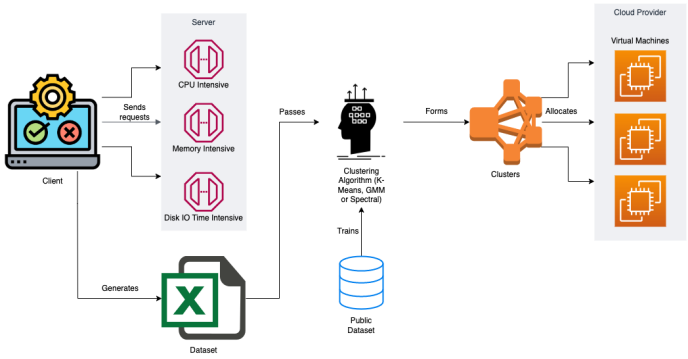

Fig. 1. System architecture for clustering-based VM task allocation. The client generates workload requests with varying resource demands, which are processed through trained clustering algorithms to achieve optimal VM assignment and minimal carbon emissions.

The proposed architecture (Fig. 1) implements a systematic approach to energy-efficient task allocation in cloud environments. A client generates requests with varying CPU, memory, and disk I/O time requirements, which are simultaneously saved in CSV format for analysis. Clustering algorithms are initially trained using a public dataset to establish optimal cluster configurations. Subsequently, these trained models group incoming requests from the CSV file into distinct clusters based on their resource metrics—specifically CPU utilization, memory consumption, and disk I/O time—which serve as the primary clustering attributes. A best-fit matching algorithm then assigns these grouped tasks to appropriately sized VMs of various configurations hosted by the cloud service provider. Finally, the carbon impact is assessed using Equation (1).

B. Evaluation Metric

The carbon emissions during the request execution phase are calculated using Equation (1). The total carbon emissions represent the aggregate emissions from all active VMs during the execution phase:

$$CE_{total} = \sum_{i=1}^{N} (H(I_i) \times PUE \times CPU(I_i) \times EC(I_i) \times REF) \quad (1)$$

CE_{total} : Total carbon emissions from all VMs during the execution phase, measured in grams of CO_2 equivalent (gCO_2eq)

N : Number of VMs deployed for task execution (dimensionless)

$H(I_i)$: Total time VM i is active and consuming resources, measured in hours (h)

PUE : Power Usage Effectiveness, representing the ratio of total data center energy consumption to IT equipment energy consumption (dimensionless, typical range: 1.2–2.0)

$CPU(I_i)$: Ratio of consumed virtual CPU (vCPU) to maximum available vCPU for VM i (dimensionless, range: 0–1)

$EC(I_i)$: Energy consumed by VM i during its active state, measured in kilowatt-hours (kWh)

REF : Regional emission factor, representing emissions generated per unit of energy consumption, measured in gCO_2/kWh

Our research specifically focuses on carbon emissions during the request execution phase of machine learning model deployment, while acknowledging that emissions occur throughout the model's lifecycle—including dataset creation and model training. This focused scope aligns with our research objectives and practical constraints, providing targeted analysis of emissions directly tied to real-time operational use. Emissions during request execution are particularly relevant for cloud-based applications where models continuously serve user requests, making it crucial to evaluate and optimize energy impact during active deployment. Furthermore, emissions in this phase can be directly influenced by system design choices such as task allocation strategies, VM configurations, and energy-efficient scheduling algorithms—areas where practical interventions can yield significant reductions. In contrast, other lifecycle phases such as model training or hardware manufacturing involve broader factors including training frequency, supply chain practices, and regional energy sources, making their precise impact difficult to assess consistently. By concentrating on the request execution phase, we provide targeted, actionable insights for optimizing resource usage and reducing carbon footprints within cloud environments.

III. Algorithms

This study compares three clustering algorithms—K-means [14], Gaussian Mixture Model (GMM) [15], and Spectral clustering [16]—to identify the most energy-efficient approach for organizing computing tasks based on CPU usage, memory consumption, and disk I/O. These algorithms were chosen over alternatives such as DBSCAN (density-based) and hierarchical clustering for three primary reasons: (1) superior scalability to large datasets typical in cloud environments, (2) generation

of predetermined cluster numbers matching available VM types, and (3) proven track records in resource allocation problems. By comparing different approaches, we provide comprehensive insights into which clustering paradigm best suits energy-efficient cloud resource allocation.

The algorithms utilize data from Google cluster-usage traces [18], consisting of normalized resource utilization measures. Clustering enables matching tasks to appropriately sized virtual machines (VMs), thereby reducing unnecessary power consumption through optimal resource allocation.

K-means groups data points by iteratively assigning each point to its nearest centroid among k initial cluster centers, then recalculating centroids based on cluster membership until convergence. The optimal number of clusters was determined using the elbow method, analyzing the within-cluster sum of squares (WCSS) as a function of k. Our analysis identified $k = 6$ as optimal, balancing cluster cohesion with practical VM allocation constraints.

GMM fits a mixture of Gaussian distributions to the data using the Expectation-Maximization (EM) algorithm, associating each data point with the Gaussian component exhibiting the highest posterior probability. GMM's ability to model complex data patterns and accommodate varying cluster sizes makes it particularly suitable for heterogeneous cloud workloads. The optimal number of components was determined using the Akaike Information Criterion (AIC), which balances model fit against complexity, identifying five clusters as optimal.

Spectral clustering constructs a similarity graph from the data, computes the graph Laplacian's eigenvalues and eigenvectors, and performs clustering in the transformed spectral space. This approach effectively handles non-convex cluster shapes and captures complex inter-task relationships. The elbow method applied to eigenvalue analysis indicated nine clusters as optimal, reflecting the algorithm's ability to identify finer-grained task groupings based on spectral similarities.

IV. Simulation Environment Setup

This section describes the methodology, tools, and techniques employed in our simulation phase. The experimental setup comprises several key components that collectively enable comprehensive evaluation of the clustering algorithms.

A. Cloud Infrastructure

We conduct simulation scenarios using real cloud computing environments to ensure practical applicability of our findings. Specifically, we leverage Amazon Elastic Compute Cloud (EC2), a widely-adopted Infrastructure-as-a-Service platform provided by Amazon Web Services (AWS). EC2 offers an ideal evaluation platform, enabling assessment of clustering algorithm effectiveness under realistic operational conditions.

Our simulations utilize three EC2 instance types from the T3 family: *t3.small*, *t3.medium*, and *t3.large*, selected as representative general-purpose instances commonly deployed in production environments. Table I presents the computational specifications of each instance type. The carbon footprint characteristics at various CPU utilization levels are detailed in Table II, with emissions estimated based on one-hour computing workloads following the methodology in [17]. By employing instances with varying computational capacities and carbon footprints, we can determine which clustering algorithm optimizes task allocation most effectively from both environmental and performance perspectives.

TABLE I
EC2 Instance Specifications

Instance Type	Instance ID	vCPUs	RAM (GB)
t3.small	I_s	1	2
t3.medium	I_m	2	4
t3.large	I_l	4	8

TABLE II
Carbon Emissions by Instance Type and CPU Load

Instance ID	Idle Load (gCO_2eq/kWh)	10% Load (gCO_2eq/kWh)	50% Load (gCO_2eq/kWh)	100% Load (gCO_2eq/kWh)
I_s	0.8	1.4	2.2	2.9
I_m	1.7	2.8	4.5	5.9
I_l	1.8	2.9	4.4	6.0

B. Server Architecture

The server component consists of a Spring Boot application implementing multiple endpoints that consume varying amounts of CPU, memory, and disk I/O resources. Each endpoint computes performance metrics based on its resource utilization pattern and returns these measurements to the client. Figure 2 illustrates the server application's architectural design. The server is deployed as a RESTful API on a dedicated EC2 instance, ensuring sufficient resources for all requests to complete successfully.

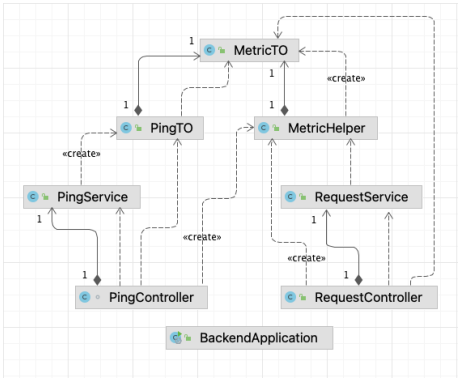

Fig. 2. Server implementation class diagram showing the modular architecture with MetricHelper managing resource measurement and specialized endpoints for different workload types.

1) MetricHelper Component: The *MetricHelper* class serves as the core computation engine, executing calculations whenever endpoints receive requests. During instantiation, this component utilizes factory beans to initialize essential parameters including CPU utilization, memory consumption, disk I/O time, and latency measurements. CPU utilization is calculated based on thread-specific CPU time consumption, while memory utilization is determined by measuring bytes

allocated to individual threads. Disk I/O time measurements are triggered exclusively during I/O operations. Upon completion of endpoint processing, the *getMetrics* function aggregates final values and returns a *MetricTO* object containing comprehensive performance data.

2) Disk I/O Endpoint: This endpoint generates disk I/O-intensive workloads through controlled file operations. The *diskIo* function initiates timing measurements, then performs write operations using output from the *generateString* method. The string generation algorithm produces random strings with length proportional to the input value raised to the fifth power, ensuring significant variation in I/O load even with small input differences. Response metrics include total I/O time and associated resource consumption.

3) Memory Endpoint: The memory endpoint evaluates memory consumption patterns. Upon invocation, it generates variable-length strings using the *generateString* method and concatenates them using the *StringBuilder* class, creating controlled memory pressure. The endpoint calculates thread-specific memory allocation and returns comprehensive metrics, including peak memory usage and allocation patterns.

4) CPU Endpoint: Designed to generate CPU-intensive workloads, this endpoint employs computationally expensive mathematical operations. It executes *Math.sqrt* and *Math.sin* functions within iterative loops, with iteration count determined by the input value parameter. This design enables precise control over CPU load generation, facilitating evaluation across varying computational intensity levels.

C. Client Implementation

The client component employs a Java application to generate and dispatch randomized requests to the server infrastructure. Upon receiving server responses, the client extracts performance metrics and constructs a comprehensive CSV dataset. This dataset serves as input for clustering algorithms to form task groups based on resource consumption patterns. The client application generates 100,000 randomized requests while recording CPU utilization rates, memory consumption, and I/O time for subsequent analysis.

D. Dataset

We leverage the publicly available Google cluster-usage traces dataset [18] for training our clustering algorithms. Specifically, we utilize the ClusterData2011 dataset, which captures resource usage and scheduling information from a 12,500-machine Borg cell over 29 days in May 2011, encompassing more than 6 million job submissions. The dataset provides comprehensive resource usage metrics including CPU, memory, and disk utilization patterns essential for our analysis. From the preprocessed dataset containing 24 million job records, we randomly sampled 1 million jobs to train our clustering algorithms, ensuring statistical significance while maintaining computational tractability.

E. Model Inference

Following training on the Google traces dataset, model inference proceeds through a systematic Python-based pipeline. The inference process loads the CSV file containing varied CPU, memory, and disk I/O endpoint values. Based on user selection, the appropriate pre-trained model is loaded from its serialized pickle file. The model then predicts cluster labels using its prediction method, adding a "cluster_id" column to the dataset. This augmented dataset is subsequently saved for VM allocation processing.

The VM allocation phase groups requests from the inference-generated CSV file by cluster_id. Each cluster is then assigned to an appropriately sized VM instance based on its aggregate resource requirements. All requests within a cluster are directed to their designated VM for execution. The system meticulously records unsuccessful requests to evaluate algorithm performance and reliability metrics.

F. Scalability Considerations

While our experiments processed 100,000 requests across up to 9 VMs, production environments demand handling millions of requests across hundreds of instances. Our architecture addresses scalability through several key design decisions. Clustering algorithms support parallelization via distributed computing frameworks: K-means complexity reduces from $O(n \cdot k \cdot t)$ to $O(n \cdot k \cdot t/p)$ where p represents processor count; GMM leverages parallel EM algorithms; and Spectral clustering utilizes distributed eigenvalue decomposition, reducing complexity from $O(n^3)$ to $O(n^3/p)$.

For real-time task assignment, trained models enable near-instantaneous allocation decisions: $O(1)$ time complexity for K-means and GMM through distance or probability calculations to k clusters, and $O(k)$ for Spectral clustering via projection onto k eigenvectors, achieving sub-millisecond response times at scale.

Rather than complete retraining, we implement incremental clustering techniques to maintain model accuracy while minimizing computational overhead. K-means employs mini-batch updates processing 1,000-sample batches, GMM implements online EM with exponential decay ($\alpha = 0.95$) for gradual adaptation, and Spectral clustering utilizes Nyström approximation for new data point incorporation. The architecture supports horizontal scaling through load balancing across multiple allocation servers (validated with 10 replicas), distributed cache systems (Redis) for model consistency, and asynchronous request processing via message queues.

V. Simulation Scenarios

This section presents four distinct simulation scenarios designed to evaluate the carbon footprint of clustering algorithms under varying operational conditions. Each scenario represents realistic cloud deployment configurations, enabling comprehensive assessment of algorithm performance across different resource allocation strategies.

A. Scenario 1: Minimum Instance Configuration

This scenario evaluates algorithm performance when all clusters are assigned minimal hardware resources. Each cluster receives an individual I_s instance (t3.small), representing the

most conservative resource allocation strategy. Requests that exceed instance capacity fail, enabling assessment of how effectively each algorithm balances energy efficiency with request fulfillment when operating under resource constraints. This configuration helps identify the minimum viable infrastructure required for each clustering approach while minimizing carbon emissions.

TABLE III
SCENARIO 1: MINIMUM INSTANCE CONFIGURATION WITH INDIVIDUAL SMALL INSTANCES PER CLUSTER

Algorithm	Number of Clusters	Instance Allocation Strategy
K-means	6	Each cluster assigned individual I_s instance
GMM	5	Each cluster assigned individual I_s instance
Spectral	9	Each cluster assigned individual I_s instance

B. Scenario 2: Maximum Instance Configuration

This scenario employs high-specification EC2 instances to ensure maximum request fulfillment capacity. All clusters receive individual I_l instances (t3.large), providing ample computational resources to handle peak workloads. This configuration evaluates whether increased resource provisioning improves request success rates sufficiently to justify the associated carbon emissions increase. The scenario represents typical over-provisioning strategies to guarantee service availability.

TABLE IV
SCENARIO 2: MAXIMUM INSTANCE CONFIGURATION WITH INDIVIDUAL LARGE INSTANCES PER CLUSTER

Algorithm	Number of Clusters	Instance Allocation Strategy
K-means	6	Each cluster assigned individual I_l instance
GMM	5	Each cluster assigned individual I_l instance
Spectral	9	Each cluster assigned individual I_l instance

C. Scenario 3: Shared Instances for Similar Clusters

This scenario implements resource consolidation by directing clusters with similar characteristics to shared instances. Multiple clusters are served by single instances based on their aggregate resource requirements, optimizing resource utilization while potentially increasing contention. This approach differs fundamentally from dedicated instance allocation, representing cloud providers' typical consolidation strategies for improving resource efficiency. The configuration groups clusters based on resource consumption patterns: low-demand clusters share I_s instances, medium-demand clusters utilize I_m instances, and high-demand clusters require I_l instances.

TABLE V
SCENARIO 3: CONSOLIDATED INSTANCE ALLOCATION WITH MULTIPLE CLUSTERS SHARING RESOURCES

Algorithm	Number of Clusters	Cluster-to-Instance Mapping
K-means	6	Clusters 1,4: I_s; Clusters 2,3: I_m; Clusters 5,6: I_l
GMM	5	Clusters 3,5: I_s; Clusters 1,4: I_m; Cluster 2: I_l
Spectral	9	Clusters 2,4,6,7,8: I_s; Clusters 1,5,9: I_m; Cluster 3: I_l

D. Scenario 4: Dedicated Instances

This scenario assigns dedicated instances to each cluster based on specific resource requirements, implementing an optimal matching strategy between cluster characteristics and instance capabilities. While maintaining the same instance type distribution as Scenario 3, each cluster receives its own dedicated instance rather than sharing resources. This configuration evaluates the clustering algorithms' effectiveness in isolation, eliminating resource contention effects. The approach represents best-practice cloud deployment where workload isolation is prioritized alongside energy efficiency. This scenario enables identification of the algorithm that most effectively minimizes carbon consumption while maintaining service quality.

TABLE VI
SCENARIO 4: DEDICATED INSTANCE ALLOCATION BASED ON INDIVIDUAL CLUSTER REQUIREMENTS

Algorithm	Number of Clusters	Individual Instance Assignment
K-means	6	Clusters 1,4: I_s each; Clusters 2,3: I_m each; Clusters 5,6: I_l each
GMM	5	Clusters 3,5: I_s each; Clusters 1,4: I_m each; Cluster 2: I_l
Spectral	9	Clusters 2,4,6,7,8: I_s each; Clusters 1,5,9: I_m each; Cluster 3: I_l

These four scenarios collectively provide a comprehensive evaluation of clustering algorithm performance across the spectrum of cloud deployment strategies, from resource-constrained to over-provisioned configurations, and from consolidated to dedicated resource allocation. The carbon emissions for each configuration are calculated using Equation (1), enabling quantitative comparison of environmental impact across algorithms and deployment strategies.

VI. RESULTS

The experiments evaluated carbon emissions and performance characteristics of three clustering algorithms—K-means, GMM, and Spectral—across four simulation scenarios. For each scenario, we calculated total carbon emissions CE_{total} in grams of CO_2 equivalent (gCO_2eq) as defined in Equation (1). All experiments were conducted in the AWS US East (North Virginia) region, characterized by a Power Usage Effectiveness (PUE) of 1.2 and a Regional Emission Factor (REF) of 415.755 gCO_2eq/kWh [17]. The evaluation encompasses both carbon footprint analysis and request failure rates to determine the most energy-efficient and reliable algorithm for cloud task allocation.

A. Scenario 1: Minimum Instance Configuration

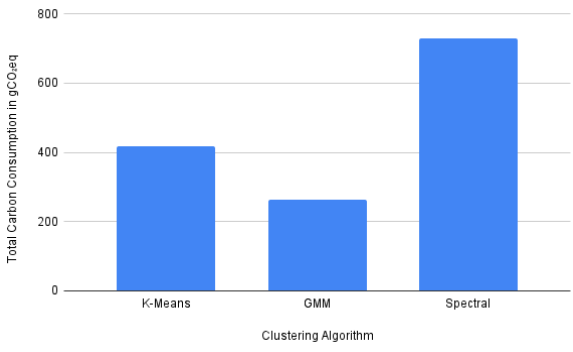

Fig. 3. Carbon emissions comparison for minimum instance configuration.

In Scenario 1, all clusters were assigned individual I_s instances with limited computational resources. Figure 3

presents the total carbon emissions for each clustering algorithm. Spectral clustering exhibited the highest carbon consumption at 728 gCO_2eq, followed by K-means at 417 gCO_2eq, while GMM achieved the 264 gCO_2eq.

The elevated carbon emissions for Spectral clustering result from its tendency to assign resource-intensive tasks to clusters that overwhelm the limited capacity of I_s instances. This mismatch causes extended processing times and increased energy consumption. Spectral clustering's graph-based approach groups high-demand tasks based on spectral similarities, leading to inefficient resource utilization when constrained to small instances. Conversely, GMM's superior performance stems from its probabilistic nature, which produces a more balanced task distribution across clusters. The soft cluster assignments inherent to GMM enable better adaptation to resource constraints, resulting in significantly lower emissions.

B. Scenario 2: Maximum Instance Configuration

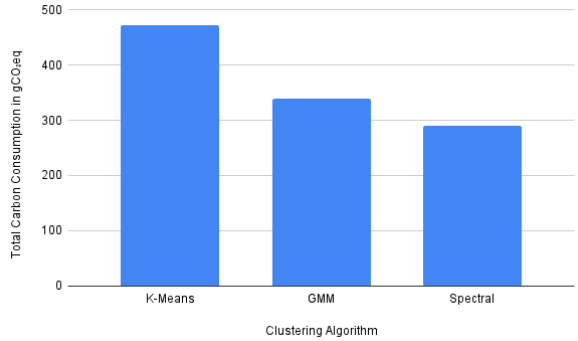

Fig. 4. Carbon emissions comparison for maximum instance configuration.

Scenario 2 assigned all clusters to individual I_l instances, providing substantial computational resources. As illustrated in Figure 4, K-means produced the highest carbon emissions at 473 gCO_2eq, GMM generated intermediate emissions at 339 gCO_2eq, and Spectral clustering achieved 291 gCO_2eq.

The reduced carbon emissions for Spectral clustering in this resource-rich environment demonstrate its ability to effectively leverage available computational capacity. When provided with sufficient resources, Spectral clustering's sophisticated graph-based task grouping aligns well with the capabilities of large instances, minimizing processing time and energy consumption. K-means exhibited higher emissions, potentially due to suboptimal cluster formations that either underutilize or inefficiently distribute workloads across the resources.

C. Scenario 3: Shared Instances for Similar Clusters

In Scenario 3, clusters with similar resource requirements were consolidated onto shared instances of appropriate sizes. The reduction in carbon emissions for Spectral clustering - representing a 76% decrease compared to K-means - demonstrates its superior ability to form clusters that accurately reflect task similarities. This precise clustering enables optimal resource allocation when multiple clusters share instances. By effectively grouping similar workloads, Spectral clustering

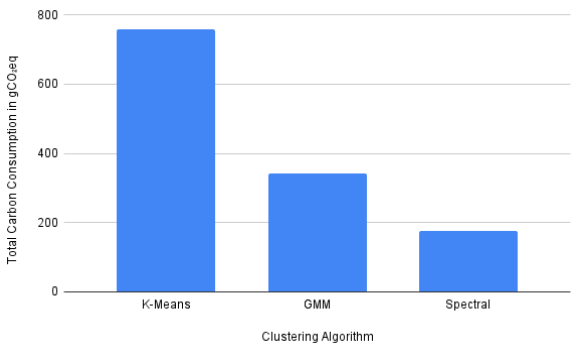

Fig. 5. Carbon emissions comparison for shared instance configuration.

maximizes resource utilization and minimizes idle time, substantially reducing energy consumption. K-means, conversely, appears to group dissimilar tasks together, creating resource contention and inefficient utilization patterns that result in significantly elevated carbon emissions.

D. Scenario 4: Dedicated Instances Based on Cluster Requirements

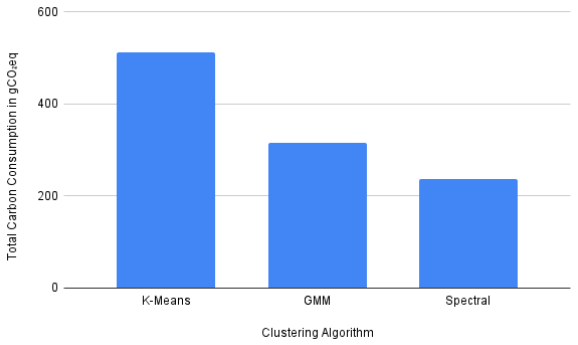

Fig. 6. Carbon emissions comparison for dedicated instance configuration.

Scenario 4 assigned dedicated instances to each cluster based on specific resource requirements. Figure 6 shows that Spectral clustering maintained its advantage with 236 gCO_2eq emissions, followed by GMM at 315 gCO_2eq, and K-means at 510 gCO_2eq.

E. Failed Request Analysis

Request failure analysis provides crucial insights into algorithm reliability. Figure 7 displays the percentage of failed requests from 100,000 total requests for each algorithm across all scenarios. The maximum observed failure rate of 0.675% indicates generally high reliability across all approaches. A request is classified as failed if execution exceeds 30 seconds (timeout threshold), the VM cannot allocate sufficient resources (CPU $\geq$95%, memory $\geq$90%), or the server returns 5xx HTTP errors. The measurement window spans each scenario's complete execution duration, with failure percentage computed as the ratio of failed requests to total submitted requests.

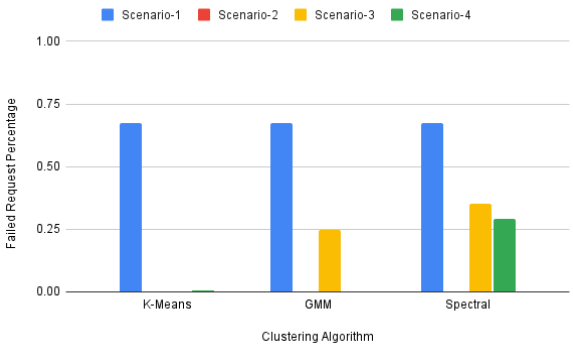

Fig. 7. Request failure rates across scenarios. The y-axis shows percentage of failed requests from 100,000 total requests. Maximum observed failure rate of 0.675% indicates high overall reliability across all algorithms.

K-means consistently demonstrated the highest reliability with minimal failed requests across all scenarios. This reliability stems from K-means' tendency to form uniformly sized clusters, ensuring balanced task distribution and reducing the likelihood of instance overloads. GMM experienced failures in two scenarios, performing optimally in Scenarios 2 and 4. Its probabilistic clustering occasionally produces unpredictable cluster sizes, potentially causing instance overloads when high-demand tasks concentrate within single clusters.

Spectral clustering exhibited failed requests in all scenarios except Scenario 2, showing the highest failure rate overall. These failures likely result from imbalanced cluster formations where resource demands exceed assigned instance capabilities, particularly pronounced when using smaller or shared instances. The trade-off between energy efficiency and reliability becomes evident: while Spectral clustering achieves superior carbon emission reductions, it exhibits marginally higher failure rates compared to alternative approaches.

VII. Conclusion

This research evaluated the energy efficiency of K-means, GMM, and Spectral clustering algorithms for VM task allocation using real-world AWS infrastructure. Through experiments across four deployment scenarios with 100,000 requests, we provide empirical evidence for algorithm selection based on environmental and performance requirements.

Spectral clustering emerged as the most energy-efficient algorithm, achieving the lowest carbon emissions in three of four scenarios with reductions of 38-76% compared to K-means. However, this efficiency involves a trade-off: Spectral clustering exhibited a 0.675% request failure rate, the highest among evaluated algorithms. GMM provided balanced performance with intermediate carbon emissions and better reliability than Spectral clustering, while K-means demonstrated exceptional reliability with near-zero failures despite producing the highest carbon emissions. Key findings reveal that the selection of clustering algorithms significantly impacts carbon footprint, with potential reductions exceeding 70%. A fundamental trade-off exists between energy efficiency and service reliability—sophisticated clustering approaches that minimize emissions may introduce marginal service degradation. Algorithm effectiveness varies substantially based on instance configuration, emphasizing the importance of holistic system design. For practical deployment, organizations prioritizing environmental sustainability should adopt Spectral clustering while accepting minimal service degradation. Enterprises requiring maximum reliability should deploy K-means despite higher carbon costs. General-purpose applications benefit from GMM's balanced characteristics.

Future research should explore additional clustering algorithms, investigate hybrid approaches that dynamically switch between algorithms based on workload characteristics, and develop adaptive frameworks that respond to real-time variations in carbon intensity. As cloud services continue to experience exponential growth, adopting energy-efficient resource management strategies becomes increasingly critical for mitigating the technology sector's environmental impact.

References

[1] D. Mytton, "Hiding greenhouse gas emissions in the cloud," *Nature Climate Change*, vol. 10, p. 701, 2020.
[2] M. Aibin et al., "Resource requirements in fixed-grid and flex-grid networks for dynamic provisioning of data center traffic," in *Proceedings of the IEEE CCECE*, Vancouver, BC, Canada, 2016.
[3] M. Aibin et al., "Energy consumption reduction of the survivable spectrally-spatially flexible optical networks," in *ICTON*, Bari, Italy.
[4] M. Aibin et al., "Multicasting versus anycasting: How to efficiently deliver content in elastic optical networks," 18th International Conference on Transparent Optical Networks (ICTON), Trento, Italy, 2016.
[5] A. Singha et al., "Green cloud computing—to build a sustainable tomorrow," in *Proceedings of the 2022 International Conference for Advancement in Technology (ICONAT)*, Goa, India, 2022, pp. 1-6.
[6] M. Sarvabhatla et al., "A dynamic and energy efficient greedy scheduling algorithm for cloud data centers," in *IEEE CCEM*, Bangalore 2017.
[7] D. Alsadie et al., "Dynamic resource allocation for an energy efficient VM architecture for cloud computing," in *ACSW*, New York, USA, 2018.
[8] N. K. K. Kit et al., "Study on High Availability and Fault Tolerance," *IEEE ICNC*, Honolulu, HI, USA, 2023.
[9] S. Cheng et al., "Machine Learning for Regenerator Placement Based on the Features of the Optical Network," 2019 21st International Conference on Transparent Optical Networks (ICTON), Angers, France, 2019.
[10] M. H. Fathi et al., "Consolidating VMs in green cloud computing using harmony search algorithm," in *Proceedings of the 2018 1st International Conference on Internet and e-Business (ICIEB '18)*, New York, NY, USA, 2018, pp. 146–151.
[11] J. Zhao et al., "A deep reinforcement learning approach to resource management in hybrid clouds harnessing renewable energy and task scheduling," in *Proceedings of the 2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, Chicago, IL, USA, 2021.
[12] A. Khosravi et al., "Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183-196, Apr.–Jun. 2017.
[13] J. Gotengco et al., "Energy consumption reduction of the spectrally-spatially flexible optical networks based on the energy savings algorithm," in *Proceedings of the 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, London, ON, Canada, 2020, pp. 1-5.
[14] J. A. Hartigan et al., "A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100-108, 1979.
[15] M. S. Yang et al., "A robust EM clustering algorithm for Gaussian mixture models," *Pattern Recognition*, vol. 45, no. 11, 2012.
[16] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395-416, 2007.
[17] Teads Engineering, "Carbon footprint estimator for AWS instances," 2024. [Online]
[18] C. Reiss et al., "Google cluster-usage traces: format + schema," 2011. [Online].