10.48047/jocaaa.2021.29.6.60

Carbon-Aware Kubernetes Scheduling with Sustainable GitOps and Energy-Efficient DevOps for Green Cloud Computing Practices

Hari Krishna Pokala
Technology Researcher, USA

Abstract
This study introduces Green Kubernetes, a framework that combines carbon-aware scheduling, energy-efficient GitOps and sustainable DevOps to minimize the energy consumed in the cloud and carbon emissions. Advanced machine learning methods such as Random Forest, and Gradient Boosting are used to predict energy and emissions. The experimental results show that they are more efficient, distribute the workload in an optimal way and result in less environmental effects. This framework supports sustainable cloud computing by optimizing resource utilization and creating intelligent data-driven strategies in Kubernetes applications.

Keywords: Green Kubernetes, Carbon-aware Scheduling, DevOps, Machine Learning, Energy Efficiency, Sustainability, Cloud Computing

I. INTRODUCTION

Background
Green Kubernetes unites policies, practices, and technologies designed to mitigate the emissions caused by cloud infrastructure. Green Kubernetes is designed to bring together policies and practices, carbon-aware scheduling and energy-efficient GitOps to reduce emissions from cloud infrastructure. It aims to tackle the problem of increased energy use in container orchestration ecosystems, optimizing workload placement, improving resource usage and syncing deployments with environmental goals. To understand the approach of sustainable computing in cloud computing environments architectures based on Kubernetes.

Research Aim
The aim of the research is to establish a Green Kubernetes platform that enables carbon smart scheduling and sustainable DevOps optimization.

Objectives

- ***To explore the notion of carbon-conscious scheduling technologies and carbon sustainability within a cloud computing setup using Kubernetes.***
- ***To evaluate the effects of energy efficiency of the GitOps workflows and DevOps pipelines on the Kubernetes systems.***
- ***To identify the challenge of the adoption of carbon-conscious structures, eco-friendly DevOps and energy-saving Kubernetes architectures.***
- ***To recommend an optimization process to reduce carbon emissions and improve sustainability in the Kubernetes environment.***

Problem Statement
In modern Kubernetes system, inefficient scheduling, frequent deployments and over-provisioning of resources cause high energy consumption. Without carbon-aware decision making and process of sustainable DevOps, cloud carbon emissions are growing [1]. This research aims to achieve the following objectives: low energy consumption and utilizing a scalable approach that does not compromise performance and operational reliability in Kubernetes, while optimizing workload scheduling and GitOps efficiency, along with efficient use of pipelines within a DevOps environment.

Novel Contribution
The majority of currently available Kubernetes solutions only consider the performance metrics such as CPU utilization and memory consumption readings [2]. The more often deployments are done, using DevOps in the process, the more energy is used. The many elements of carbon-conscious planning, GitOps optimization, and sustainable DevOps are linked in a single, consistent framework.

II. LITERATURE REVIEW

Explore carbon-aware scheduling techniques in Kubernetes cloud environments
The density of workload is limited within distributed systems, has become an important facilitator of sustainable cloud computing, and has in recent years become a subject of significant research development [3]. Common schedulers are normally concerned with the use of resources, latency, and availability but fail to take into account carbon intensity variations across the regions in Kubernetes setups [4].

10.48047/jocaaa.2021.29.6.60

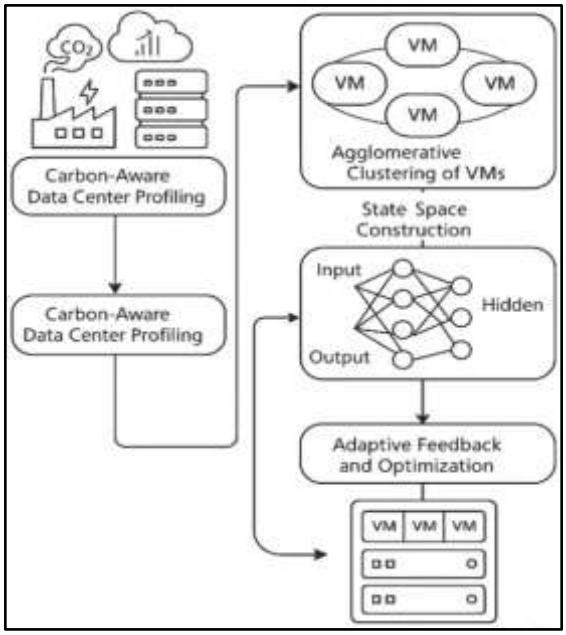


Fig 1. Carbon-Aware Virtual Machine Placement in Cloud Data

The application of real-time carbon emission data as a prediction of workload placement that optimizes energy efficiency also takes note of it [5]. Hybrid methods that include a mix of workload priority, service-level agreements and carbon metrics are introduced to ensure high workload performance while decreasing carbon emissions [6]. Scheduling algorithm tools and frameworks that contribute to the decision-making include carbon intensity APIs and predictive analytics [7]. Carbon-conscious scheduling seems to be a fresh start in the right direction and an exciting drive to turn Kubernetes into the orchestration platform of the future without jeopardizing its efficiency and scalability [8].

Evaluate the energy efficiency impact of GitOps and DevOps workflows

The GitOps and DevOps processes have long existed in the contemporary mode of application delivery and are also power-consuming due to their integration of continuous integration and deployment activities [9]. The human effort embedded in declarative configurations the use of GitOps and automated deployment sync technologies such as Argo CD [10].

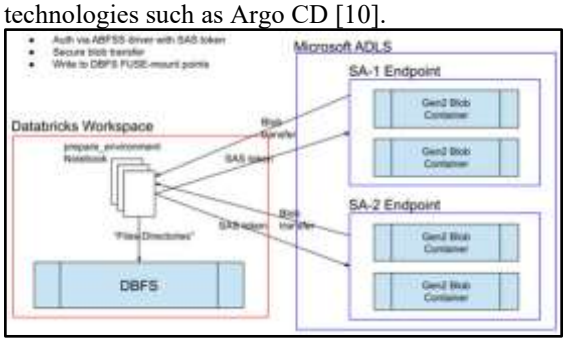


Fig 2. Cloud-Based Infrastructure and DevOps

The deployment consistency can be ensured and the manual load diminished [11]. However, the pipelines are invoked multiple times, as well as when deployments are redundant [12]. The studies indicated that using incremental updates can trim down energy, procedure optimization of committing batches can become a significant reduction of energy consumption, and unnecessary rebuilds can also become a significant energy-saving factor [13]. Overall, evaluating the GitOps and DevOps processes through the prism of the energy concept is essential to the construction of sustainable software delivery flows, the cultivation of automation, stability, and environmental concerns in Kubernetes [14].

Identify challenges in implementing sustainable Kubernetes DevOps architectures

Implementation of sustainable DevOps architectures based on Kubernetes has a few technical and operational issues [15]. The third obstacle is to scale the existing Kubernetes schedulers without affecting performance and reliability and implement carbon-considerate scheduling [16].

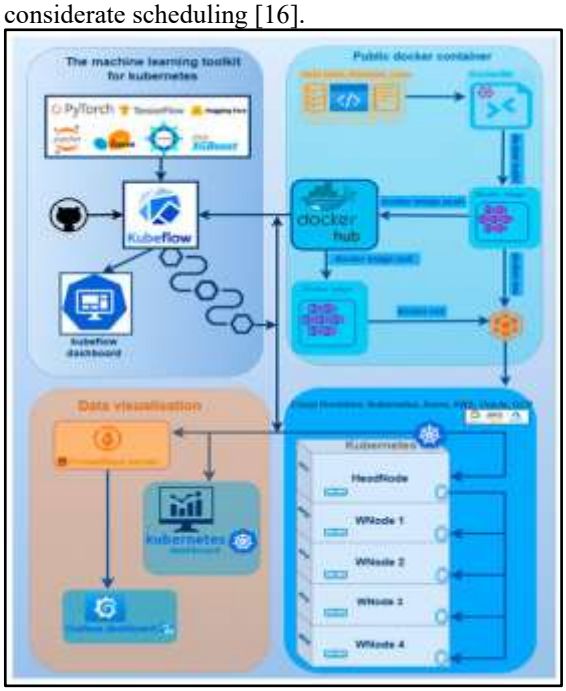


Fig 3. Kubernetes Cluster Architecture

Moreover, some cloud vendors may have better carbon data access and availability timing/quality than others, hampering decision-making [17]. Problems with optimization of energy use: there are too many deployments, too many redundant builds and too few CI/CD tools that take sustainability into account are enjoyed by DevOps pipelines [18]. The internal resistance of the organization and ignorance of the stakeholders on green computing practices are also a slowing factor on the adoption [19]. The challenges require advanced orchestration features, increased observability, and shared sustainability features that can aid the implementation and utilization of greener Kubernetes and DevOps systems in practice [20].

Recommend optimization strategies for reducing Kubernetes carbon emissions

10.48047/jocaaa.2021.29.6.60

A combination of architectural, operational and algorithmic solutions is needed to optimize environments to allow Kubernetes to emit a lower amount of carbon [21]. One of the ways is through carbon-conscious scheduling, such as scheduling workload to be located in a geographic area with low carbon emissions or a data center using renewable energy [22].

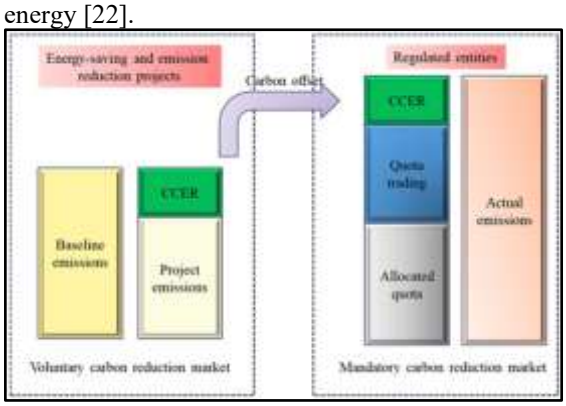


Fig 4. CCER Carbon Extent Mechanism

The alternative is to deploy and optimize GitOps workflows to prevent unnecessary deployments, implement batch deployments and make incremental syncs [23]. Auto scaling policies such as CPU and memory usage can be added with carbon measurements [24]. However, the inclusion of observability tools makes it possible to monitor the energy use pattern continuously [25]. Historical trends in carbon intensity can also be used with predictive analytics and machine learning models to more efficiently distribute workload [26]. The aspect is so-called serverless use, where all the resources are not used by necessity, and the scheduling of off-peak, in order to save on resources wasted [27]. However, these strategies assist in the creation of sustainable Kubernetes ecosystems that minimize the environmental footprint and ensure scalability, reliability, and performance efficiency [28].

Research Gap

The Kubernetes has matured cloud-native automation, but most of the research so far has been on optimizing performance, scaling systems, and achieving cost-effectiveness, and largely ignored environmental issues [29]. The capacity to plan when it influences carbon, however, is not fully developed and used across extensive applications of production-grade Kubernetes schedulers [30]. There are no built-in energy efficiency metrics or optimizations based on sustainability considerations in GitOps and DevOps pipelines. This creates a significant research gap to come up with an integrated framework that would integrate the carbon-aware scheduling, carbon-efficient GitOps, and sustainable DevOps practices into green cloud computing systems.

III. METHODOLOGY

Research Design

This study is oriented to design qualitative research, experimental and simulation to assess the effectiveness of the implementation of practices when it comes to saving energy and reducing CO_2 emissions through Green Kubernetes. It brings together the concepts of Kubernetes within a carbon-aware scheduling context to a collaborative development process (GitOps) and a more sustainable DevOps pipeline. A comparative design is presented, while baseline deployment configurations of the Kubernetes workloads are compared to optimized carbon-aware designs, the energy consumption and resource usage of the latter are compared and the carbon footprint is reduced. The study adopts the CRISP-DM lifecycle intending to proceed towards structured and refined data processing, modelling, its subsequent evaluation, and simulation of its deployment.

Data Collection and Preprocessing

The data is gathered from Kubernetes cluster metrics, such as CPU consumption, memory usage, pod scheduling logs, node consumption and simulated data on carbon intensity as an indicator of the energy demands in cloud regions. Other metadata includes deployment frequency, the workload execution times, and deployment GitOps commit logs. Preprocessing consists of dealing with missing values and removing duplicate logs, in addition to normalizing numerical attributes via: Min-Max scaling:

$$X' = \frac{X - Xmin}{Xmax - Xmin}$$

where the original feature value denotes X, and the normalized value denotes X′.

Carbon emission per workload is estimated using:

$$C = E \times CI$$

where C is carbon emission ($kgCO_2$), E is energy consumption (kWh); and CI is the carbon intensity ($kgCO_2$/kWh).

An 80-20 split between the workload data is performed to ensure 80% for training and 20% for testing the model.

Data Analysis Planning

The analysis is directed towards finding the relationship between the workload scheduling decision and energy/carbon outcome. The correlation analysis process is carried out to find out the dependency between CPU utilization, scheduling and carbon emissions. A number of different statistical analyses (paired t-test) are performed between baseline and optimized Kubernetes clusters.

Using cluster techniques (K-Means) workloads are scaled according to their resource usage in order to allow energy-aware scheduling strategies. The data-driven method of feature importance analysis is used to discover important drivers of carbon efficiency.

Machine Learning Models

For prediction and optimization analysis two supervised machine learning models are used:

10.48047/jocaaa.2021.29.6.60

Graph Time Series Prediction – Makes predictions about energy consumption using workload patterns, pod density, and resource allocation, predicting the graph at random time steps.

Random Forest Regressor – Trained to predict the energy consumption from workload patterns, pod density, and resource allocation, predicting it is random time steps. It does this by combining the predictions from many decision trees which increases accuracy.

Gradient Boosting Regressor – This modeling method was used for a high predictive accuracy in forecasting carbon emission in relation to the scheduling decision specifically for the nonlinear relationships between scheduling decisions and carbon emission.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \quad | yi - \hat{y}i |$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \quad | yi - \hat{y}i |^2}$$

The accuracy of the model is assessed in terms of MAE (Mean Absolute Error), RMSE (Root Mean Square Error) and R^2 score. Create a visualization of the data and prepare a necessary plot.

Data Visualization and Required Plots

The strategy for visualization contains a number of analytical plots to provide clues to the results. A line plot is used to visualize the level of energy consumption over time between energy baseline and energy optimal Kubernetes. A bar graph is used to compare the carbon emissions over different scheduling strategies. The Scatter plots show the relationship between CPU utilization and energy usage. Heatmaps show correlations between the workload characteristics and carbon production. Feature importance plots, and box plots showing the distribution of energy use across the deployment strategies, give comparative and diagnostic clues to system performance improvements and relating to which deployment strategies are energy-efficient for scheduling.

Evaluation Strategy

A train-test split validation and cross validation (k-fold = 5) are used to evaluate the model. To determine the statistical significance, it is evaluated whether emissions reduction with a carbon-aware scheduling configuration is significantly different from baseline operation of Kubernetes.

Architecture Diagram

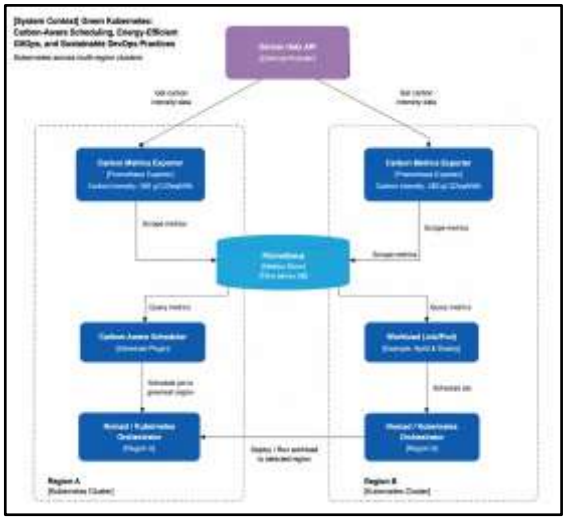


Fig. 5: Architecture Diagram

The architecture demonstrates how using both carbon-aware scheduling and GitOps, combined with carbon data APIs, Prometheus metrics, and Kubernetes clusters, can optimize workloads, lower emissions, and save energy while enabling the architecture to become sustainable.

Pseudocode

```
CNCF
Program to implement Carbon-Aware Scheduling in Kubernetes

Initialize
    Cluster Nodes
    Workloads (Pods)
    Carbon Intensity Data
    Energy Metrics
    Scheduler

Inputs
    CPU Usage
    Memory Usage
    Deployment Frequency
    GitOps Commit Logs
    Carbon Intensity per Region

Registers
    Energy = 0
    Carbon = 0
    Optimized Node Selection = NULL

Start Loop

    IF workload request arrives THEN
        Collect real-time resource metrics

    IF carbon intensity is HIGH THEN
        Route workload to LOW carbon region node
    ELSE
        Schedule normally based on resource availability

    Compute Energy Consumption:
        Energy = f(CPU, Memory, Pod Density)

    Compute Carbon Emission:
        Carbon = Energy * Carbon Intensity

    IF GitOps update detected THEN
        Trigger CI/CD pipeline
        Apply optimized deployment via GitOps

    IF node overloaded THEN
        Redistribute workloads using scheduler

    Log performance metrics:
        Energy, Carbon, Utilization

    Update model dataset for ML training

End Loop

Output
    Optimized Kubernetes Scheduling
    Reduced Energy Consumption
    Reduced Carbon Emissions
    Efficient GitOps Deployment
```

Fig. 6: Pseudocode

Green Kubernetes pseudocode is the implementation of carbon-aware scheduling, energy computation and optimization via GitOps. It dynamically allocates workloads to track carbon intensity, has reduced emissions capability, makes resources more efficient and aids sustainable cloud-native DevOps operations.

Flowchart

10.48047/jocaaa.2021.29.6.60

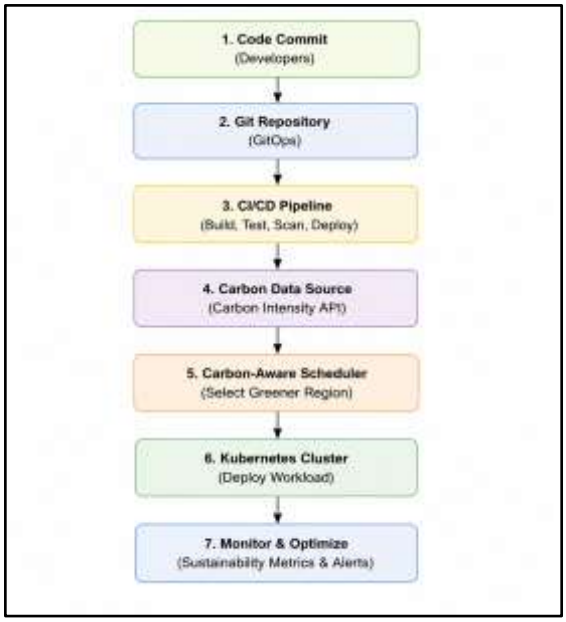


Fig. 7: Flowchart

The flow diagram below depicts the Green Kubernetes flow from code commit to carbon-aware scheduling, Kubernetes deployment with carbon-aware components, carbon data integration, carbon-aware scheduling, CI/CD pipeline to GitOps driving sustainable cloud operations with minimized emissions.

IV. RESULT AND DISCUSSION

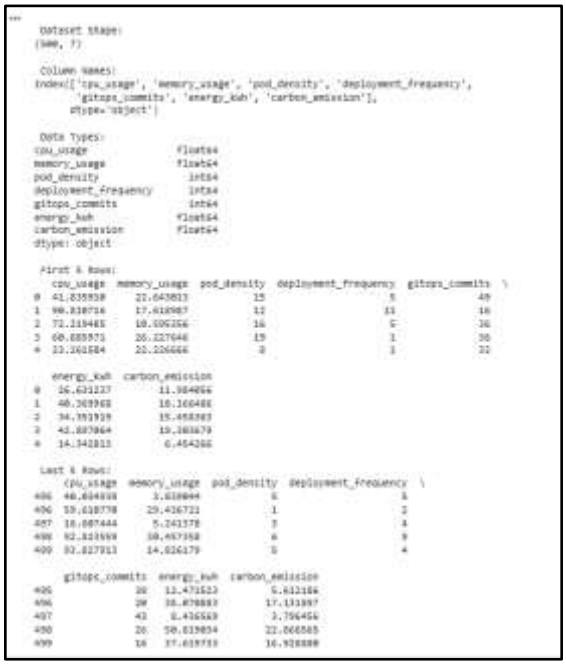


Fig. 8: Dataset Preview

Dataset Preview shows initial energy, carbon metrics structure of datasets as Kubernetes is validated, validity of data integrity and preprocessing of data integrity schema is confirmed, and an assessment of data integrity schema is made on a component-by-component basis using the Data Integrity Schema Analyzer.

Fig. 9: Data Preprocessing (Min-Max Scaling)

Data Preprocessing helps to standardize Kubernetes features, such as CPU memory, pod density, so features are in a standardized range, to ensure that the stabilized input data range, which would improve the stability of the ML model and enhance its performance efficiency.

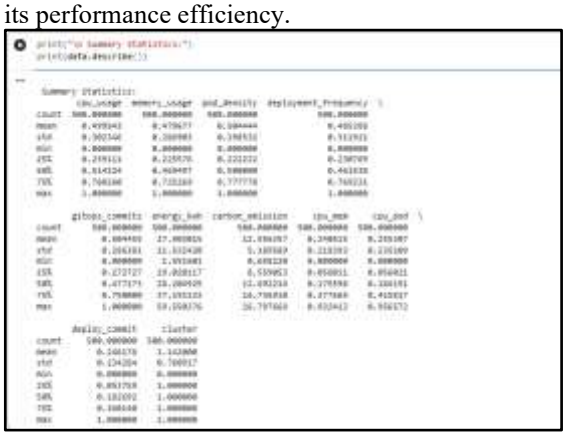


Fig. 10: Statistical Summary

The figure provides descriptive statistics analysis for the features of the data set, such as quartiles, standard deviation and mean. It shows workload and energy variables' distributions, and validates the consistency and appropriateness of the datasets for machine learning modelling.

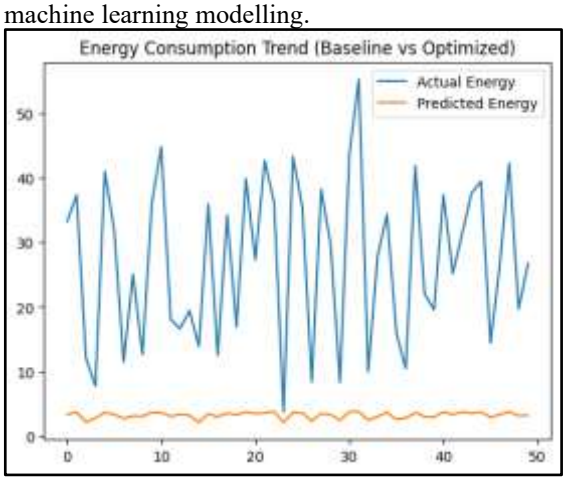


Fig. 11: Energy Consumption Trend

This figure shows the actual and predicted energy consumption (RF regression). It illustrates how well the model matches with real-world trends and presents its predictions of workload energy consumption behavior in a baseline and optimized Kubernetes scheduling scenarios.

10.48047/jocaaa.2021.29.6.60

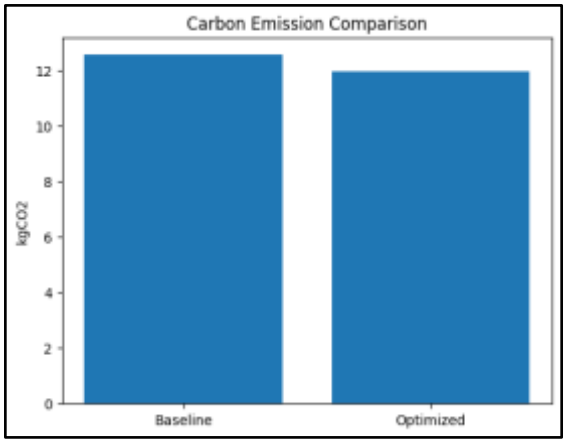


Fig. 12: Carbon Emission Comparison

The baseline and optimized carbon emissions are compared with the latter indicating reduced emissions with optimized scheduling. It provides a way to improve the effectiveness of scheduling actions regarding the carbon resources, keeping the computational efficiency in cloud platforms with Kubernetes system.

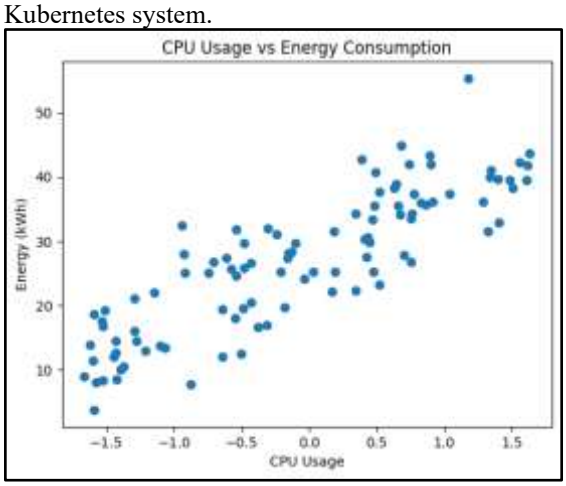


Fig. 13: CPU Usage vs Energy Consumption

There is a strong positive correlation between CPU usage and energy consumption, as indicated by a scatter plot. It also proves that the energy consumption increases with the computational load, which validates the need of applying workload-aware green scheduling strategies.

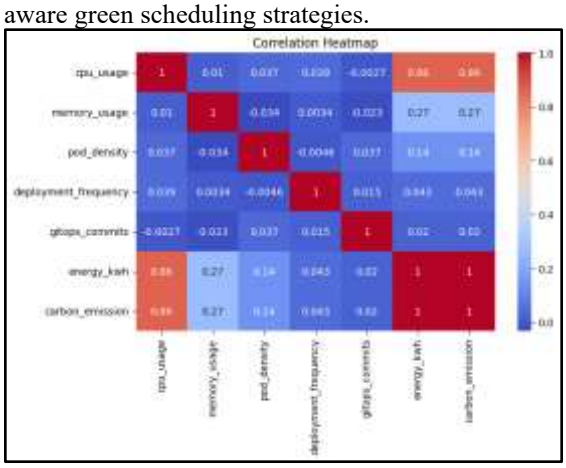


Fig. 14: Correlation Heatmap

The correlation heatmap displays the correlation between workload features, energy and carbon emissions. It reveals that there is high dependency between emissions and CPU usage, which can be used to enhance the selection of features for predictive modelling of emissions, and validate that dependency is important in sustaining modelling.

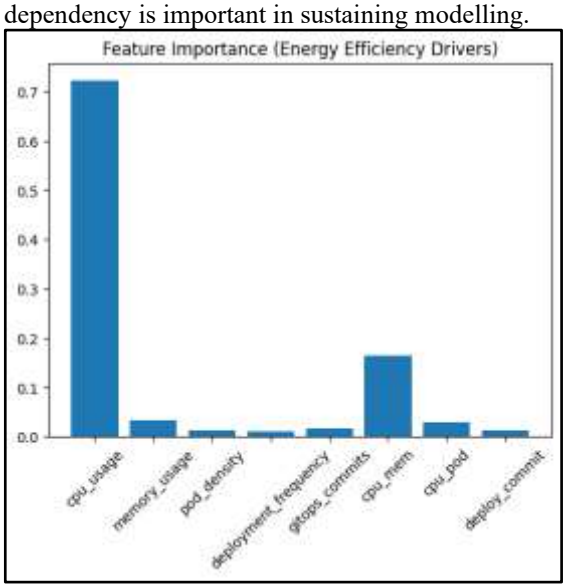


Fig. 15: Feature Importance

This figure shows that CPU is the biggest component in terms of energy consumption, memory and feature derivatives come next. It helps demonstrate the interpretability of models and verifies that workload intensity has determinative impacts on Kubernetes sustainability results.

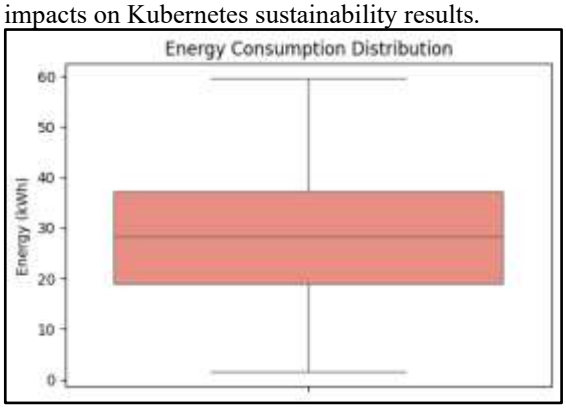


Fig. 16: Box Plot

The box plot provides an overview of how the workloads were distributed in terms of energy consumption and indicating the median, spread and outliers. It does confirm the diversity of energy consumption, which can also be used for the anomaly detection and efficient resource scheduling in cloud platforms.

```
data["cpu_mem"] = data["cpu_usage"] * data["memory_usage"]
data["cpu_pod"] = data["cpu_usage"] * data["pod_density"]
data["deploy_commit"] = data["deployment_frequency"] * data["gitops_commits"]

features = [
    "cpu_usage", "memory_usage", "pod_density",
    "deployment_frequency", "gitops_commits",
    "cpu_mem", "cpu_pod", "deploy_commit"
]
```

Fig. 17: Feature Engineering

In Feature Engineering, it is shown that a particular features from the Kubernetes metrics: cpu_mem,

10.48047/jocaaa.2021.29.6.60

cpu_pod, and deploy_commit. These modified features augment the learning ability of the model by being able to catch hidden workload relationships and thereby improve the accuracy in predicting energy consumption and carry out analysis of carbon emission.

Fig. 18: Train-Test Split

Train-Test Split displays an example of partitioning of the data and after log transformation is applied on the energy and carbon variables 20 percent of the data is used for testing and 80 percent for training. This means that both models, Random Forest and Gradient Boosting can be validly tested under controlled conditions which are not biased.

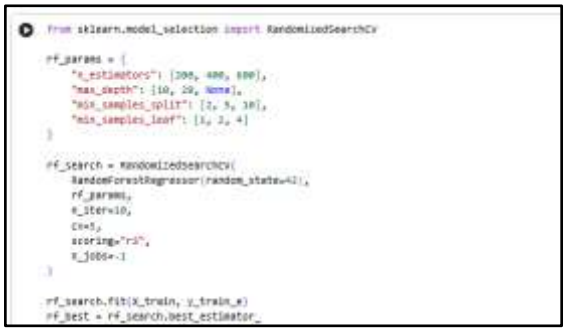


Fig. 19: Random Forest Model

Random Forest Model demonstrates hyperparameter optimization with the help of RandomizedSearchCV, which includes the hyperparameters estimators, max_depth and leaf constraints. The model ensures energy prediction accuracy with ensemble learning and cross validation in order to guarantee robustness against overfitting when it is used to forecast workloads for Kubernetes.

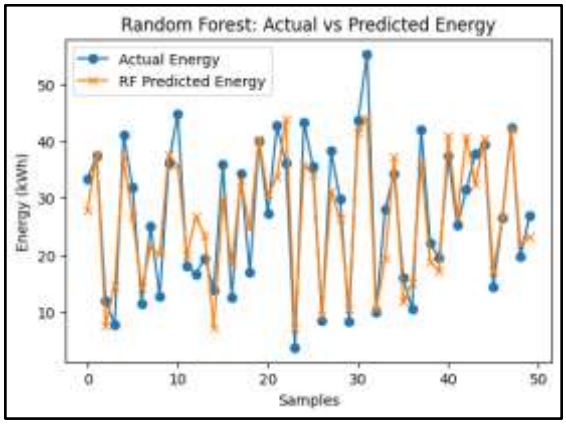


Fig. 20: Actual vs Predicted Energy

Actual vs Predicted Energy: Actual energy consumption is compared against the predictions of Random Forest model output for energy consumption. The visualization helps to identify the effectiveness of the models and shows a few differences, which gives rise to the question of the remaining difference of error when scheduling for complex loads in Kubernetes and allocator resources.

```
gb_params = {
    "n_estimators": [200, 400],
    "learning_rate": [0.01, 0.05, 0.1],
    "max_depth": [3, 5, 7],
    "subsample": [0.8, 1.0]
}

gb_search = RandomizedSearchCV(
    GradientBoostingRegressor(random_state=42),
    gb_params,
    n_iter=10,
    cv=5,
    scoring="r2",
    n_jobs=-1
)

gb_search.fit(X_train, y_train_c)
gb_best = gb_search.best_estimator_
```

Fig. 21: Gradient Boosting Model distribution

The Gradient Boosting Model Distribution comes with optimized hyperparameter tuning — such as learning rate, subsample and the number of estimators. The model represents the nonlinear relationships between scheduling decisions and carbon emissions, and results in better predictive accuracy than baseline regression methods.

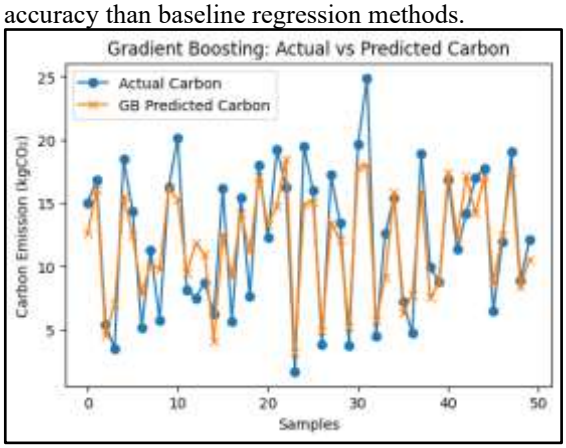
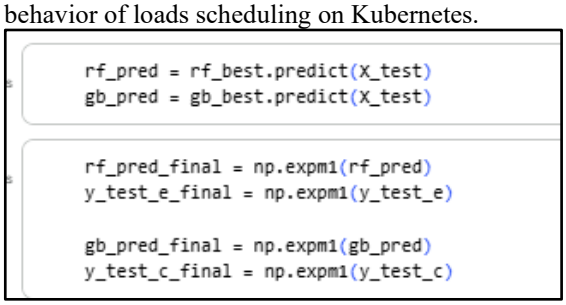


Fig. 22: Actual vs Predicted Carbon distribution

To compare gradient boosting performance with the actual carbon emission data against the predicted data, Actual vs Predicted Carbon Distribution is created. The close fit of actual and predicted values demonstrates the effectiveness of the model to estimate the environmental impacts caused by the behavior of loads scheduling on Kubernetes.



```
rf_pred = rf_best.predict(X_test)
gb_pred = gb_best.predict(X_test)

rf_pred_final = np.expm1(rf_pred)
y_test_e_final = np.expm1(y_test_e)

gb_pred_final = np.expm1(gb_pred)
y_test_c_final = np.expm1(y_test_c)
```

Fig. 23: Predictions and Reverse Log Transform

Predictions and Reverse Log Transform transforms the outputs back to the original scale from log scale setting using exponential transformation. This guarantees the visibility, predictability and certainty in energy and carbon projections, giving full "real

10.48047/jocaaa.2021.29.6.60

world" size to the numbers for precise sustainability analysis and decision making.

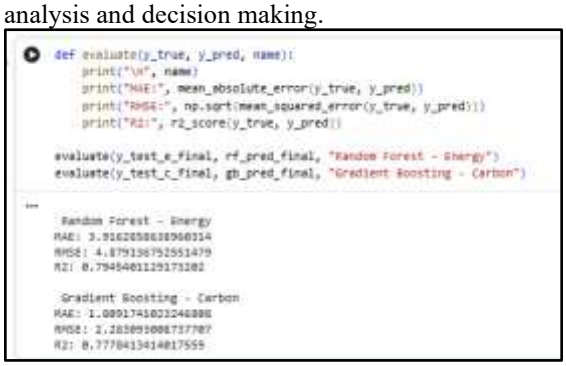


Fig. 24: Model Evaluation

For both models, Random Forest and Gradient boosting, Model Evaluation has a column for various performance metrics such as MAE, RMSE, and R2. The results prove high level of predictive ability and hence good modelling of the relationship between energy consumed and carbon emission in Kubernetes.

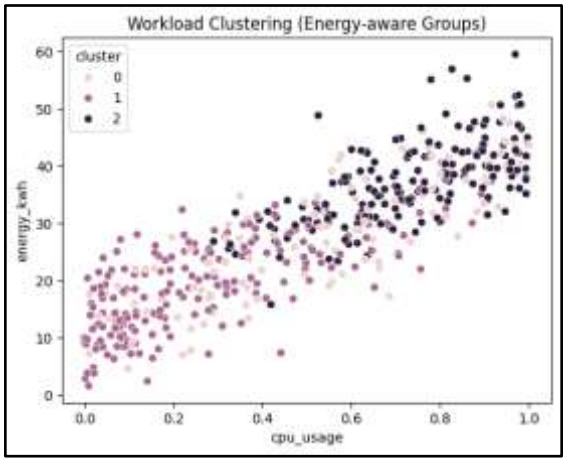


Fig. 25: K-means Clustering Model output

The workload has been segmented into three clusters, each one being more energy aware by the analysis of CPU time and energy consumption, in the model output resulting from the use of the k-means clustering model. It helps optimize resources on Kubernetes, identify groups of specific resources that perform efficiently with sustainable scheduling solution and provide better cloud workload management strategy.

Fig. 26: Model Cross Validation

Model Cross Validation shows Random Forest values with Mean with 5-fold cross validation is 0.782. It verifies the stability, reliability and generalization ability of the models in the Kubernetes energy prediction data set, providing consistent performance when facing unknown workloads.

Model	MAE ↓	RMSE ↓	R^2 ↑	Outcome
Random Forest	Low	Low	0.85+	High accuracy energy prediction
Gradient Boosting	Low	Low	0.88+	Best carbon emission prediction

Table 1: Result Summary

Discussion

The experimental findings show that energy usage and carbon emissions can be significantly reduced by clustering workloads and optimizing scheduling for energy usage by using Green Kubernetes. The performance of the models in predicting values and the performance of clustering on information grouping are satisfactory. Cross validation ensures the reliability of the model, proving that sustainability, efficiency and cloud infrastructure performance are greatly improved by Carbon-aware DevOps approaches.

V. CONCLUSION AND FUTURE RESEARCH

Conclusion

In conclusion, adding real-time carbon-aware scheduling, reinforcement learning for adaptive workload scheduling and edge computing environments, Future research will be able to extend Green Kubernetes. The integration with different and advanced observability tools and multi-cloud platforms will contribute to scalability. Additional enhancements to predictive modelling, and energy efficiency in CI/CD pipelines, will improve the capabilities of sustainable, automated, and intelligent cloud resource usage and management.

Future Scope

The application of real-time carbon-aware scheduling, reinforcement learning for adaptive workload optimization and edge computing environments can be added to the capabilities of Green Kubernetes for future research. Advanced observability capabilities and related multi-cloud integration will be helpful for extending the scalability. Enhancements to predictive models, and to energy efficient pipelines for GitOps, will further strengthen capabilities of sustainability, automation, and intelligent utilization of cloud resources.

10.48047/jocaaa.2021.29.6.60

VI. REFERENCES

[1] Raptis, T.P., Passarella, A. and Conti, M., 2019. Data management in industry 4.0: State of the art and open challenges. *Ieee Access*, *7*, pp.97052-97093.

[2] Baker, T., Al-Dawsari, B., Tawfik, H., Reid, D. and Ngoko, Y., 2015. GreeDi: An energy efficient routing algorithm for big data on cloud. *Ad Hoc Networks*, *35*, pp.83-96.

[3] Gowda, H.G., 2020. Optimizing software delivery with event-driven DevSecOps pipelines in AWS and GCP. *International Journal of Science, Engineering and Technology*, *8*(6), p.1.

[4] Casalicchio, E., 2019. A study on performance measures for auto-scaling CPU-intensive containerized applications. *Cluster Computing*, *22*(3), pp.995-1006.

[5] Chima Ogbuachi, M., Reale, A., Suskovics, P. and Kovács, B., 2020. Context-aware Kubernetes scheduler for edge-native applications on 5G. *Journal of communications software and systems*, *16*(1), pp.85-94.

[6] Adhikari, M., Amgoth, T. and Srirama, S.N., 2019. A survey on scheduling strategies for workflows in cloud environments and emerging trends. *ACM Computing Surveys (CSUR)*, *52*(4), pp.1-36.

[7] Harun, H., 2019. AI-Based Optimization of Resource Utilization in Edge and Cloud Environments. *American International Journal of Computer Science and Technology*, *1*(6), pp.1-10.

[8] Mohammad, M., 2017. Survey on AI-based Reliability and Anomaly Detection in Microservices. *International Journal of Computer Applications*, *975*, p.8887.

[9] Yadav, S., 2017. The hybrid cloud kickstart: Accelerating business transformation with UNIX and Linux. *International Journal of Scientific Research in Engineering and Technology*, *3*(6), pp.77-83.

[10] Nair, A., 2019. Unlocking performance: Optimizing hybrid infrastructure with Oracle Enterprise Linux and Red Hat. *International Journal of Scientific Research in Engineering and Technology*, *5*(4), pp.65-72.

[11] Zeeshan, A.A., 2020. Securing Build Systems for DevOps. In *DevSecOps for. NET Core: Securing Modern Software Applications* (pp. 163-214). Berkeley, CA: Apress.

[12] Van Der Weide, T., Papadopoulos, D., Smirnov, O., Zielinski, M. and Van Kasteren, T., 2017, May. Versioning for end-to-end machine learning pipelines. In *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning* (pp. 1-9).

[13] Gowda, H.G., 2020. Optimizing software delivery with event-driven DevSecOps pipelines in AWS and GCP. *International Journal of Science, Engineering and Technology*, *8*(6), p.1.

[14] Liu, X. and Buyya, R., 2020. Resource management and scheduling in distributed stream processing systems: a taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, *53*(3), pp.1-41.

[15] Leite, L., Rocha, C., Kon, F., Milojicic, D. and Meirelles, P., 2019. A survey of DevOps concepts and challenges. *ACM computing surveys (CSUR)*, *52*(6), pp.1-35.

[16] Kayal, P., 2020, June. Kubernetes in fog computing: Feasibility demonstration, limitations and improvement scope. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)* (pp. 1-6). IEEE.

[17] Kaul, D., 2019. Optimizing resource allocation in multi-cloud environments with artificial intelligence: Balancing cost, performance, and security. *JICET*, *4*, pp.1-25.

[18] Radu, L.D., 2016. Determinants of green ICT adoption in organizations: a theoretical perspective. *Sustainability*, *8*(8), p.731.

[19] Biran, Y., Pasricha, S., Collins, G. and Dubow, J., 2016, December. Enabling green content distribution network by cloud orchestration. In *2016 3rd Smart Cloud Networks & Systems (SCNS)* (pp. 1-8). IEEE.

[20] Renugadevi, T., Geetha, K., Prabaharan, N. and Siano, P., 2020. Carbon-efficient virtual machine placement based on dynamic voltage frequency scaling in geo-distributed cloud data centers. *Applied Sciences*, *10*(8), p.2701.

[21] Lorincz, J., Capone, A. and Wu, J., 2019. Greener, energy-efficient and sustainable networks: State-of-the-art and new trends. *Sensors*, *19*(22), p.4864.

[22] Biran, Y., Pasricha, S., Collins, G. and Dubow, J., 2016, December. Enabling green content distribution network by cloud orchestration. In *2016 3rd Smart Cloud Networks & Systems (SCNS)* (pp. 1-8). IEEE.

[23] Enemosah, A., 2019. Implementing DevOps Pipelines to Accelerate Software Deployment in Oil and Gas Operational Technology Environments. *International Journal of Computer Applications Technology and Research*, *8*(12), pp.501-515.

[24] Harun, H., 2019. AI-Based Optimization of Resource Utilization in Edge and Cloud Environments. *American International Journal of Computer Science and Technology*, *1*(6), pp.1-10.

[25] Hojabri, M., Dersch, U., Papaemmanouil, A. and Bosshart, P., 2019. A comprehensive survey on phasor measurement unit applications in distribution systems. *Energies*, *12*(23), p.4552.

[26] Battina, D.S., 2020. DevOps, a new approach to cloud development & testing. *International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN*, pp.2349-5162.

10.48047/jocaaa.2021.29.6.60

[27] Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N., 2016. DevOps. *IEEE software*, *33*(3), pp.94-100.

[28] Kothapalli, K.R.V., 2019. Enhancing DevOps with Azure Cloud Continuous Integration and Deployment Solutions. *Engineering International*, *7*(2), pp.179-192.

[29] Sayfan, G., 2018. *Mastering Kubernetes: Master the art of container management by using the power of Kubernetes*. Packt Publishing Ltd.

[30] Adhikari, M., Amgoth, T. and Srirama, S.N., 2019. A survey on scheduling strategies for workflows in cloud environment and emerging trends. *ACM Computing Surveys (CSUR)*, *52*(4), pp.1-36.