Integrating Energy APIs in Cloud Computing to Incentivize Sustainable Computing

Kavita Cardozo
University of Illinois Urbana-Champaign
Champaign, IL
cardozo2@illinois.edu

Raoul Larios
University of Illinois Urbana-Champaign
Champaign, IL
rlarios2@illinois.edu

Swarup Ghosh
University of Illinois Urbana-Champaign
Champaign, IL
swarupg2@illinois.edu

Kevin Le
University of Illinois Urbana-Champaign
Champaign, IL
kevinle2@illinois.edu

***Abstract*—Efficiency in cloud computing has traditionally focused on reducing cloud operational costs by scaling, scheduling, and deferring workloads to align with demand patterns while meeting service-level objectives (SLOs). These methods also contribute to energy conservation, yet they often overlook the potential for dynamic power source optimization. This paper introduces a novel approach that integrates real-time data on brown (non-renewable) and green (renewable) electricity from APIs to enable data centers to adjust workload distribution both within and across data centers based on power grid information. Leveraging the CloudSim[18] framework, developed to simulate data centers and underlying infrastrucure, this study explores a methodology to shift loads preferentially toward green power sources while maintaining essential SLOs. Initial experiments focus on aligning power grid data with simulated data center operations, resulting in a cost and scoring mechanism to quantify potential cost savings and environmental impact. The paper further proposes a formula to calculate an energy score based on the optimized energy use. Findings and conclusions derived from these simulations will provide insights into the feasibility and impact of this approach, with the hope to further sustainable cloud computing practices.**

I. Introduction

Leading consulting firms and research analysts predict[1] that U.S. data center power demand will double by 2030, with a similar global trend. A key factor driving this surge is the growing use of AI, which requires significant computational resources. Therefore, improving energy efficiency in machine learning, model training, and cloud hosting is crucial to optimizing computing and reducing carbon footprints.

IT hardware, including CPUs and storage, accounts for 45% of a data center's power consumption, with servers and storage comprising 75% of that share (Ahmed et al. [1]). While current strategies like virtualization, containerization, autoscaling, and load balancing aim to improve resource efficiency, few incorporate real-time energy data or optimize workloads based on the grid's power state. Additionally, the lack of commercial incentives tied to "Green metrics" hinders more sustainable practices.

Promising recent work addresses these gaps. Thiede et al. [11] introduced Carbon Containers to manage carbon emissions, inspiring our approach to integrating real-time energy data for smarter workload management. Similarly, Chen et al. [10] demonstrated how optimizing GPU usage through adaptive co-location can significantly enhance energy efficiency, especially for GPU-heavy machine learning tasks. Studies by Vasconcelos et al. [16] explored cloud federations that place workloads in regions with lower carbon footprints, factoring in geographic and temporal carbon intensity variations. Jay et al. [17] analyzed software-based power metering techniques, emphasizing precise CPU and GPU power monitoring to optimize energy use and reduce emissions.

In real-time resource management, Zhang et al. [12] tackled dynamic resource orchestration in containerized cloud environments, aligning with our concept of allocating workloads across energy-efficient tiers. Zhao et al. [13] provided insights into GPU power capping for AI to control energy use in high-performance environments.

Inspired by Tyagi and Sharma's Scavenger [14], which balances cost and performance in ML training, we aim to integrate Green metrics alongside traditional pricing to encourage energy-conscious choices. Our project proposes a multi-tiered orchestration system that allocates resources based on Service-Level Objectives (SLOs), real-time energy availability, and sustainability goals—optimizing energy use, reducing emissions, and maintaining performance.

II. Related Work and Preceding Efforts

The goal of integrating energy awareness into cloud computing has led to significant research focused on optimizing workload distribution, resource allocation, and scaling mechanisms in cloud environments to reduce operational costs and environmental impact. Previous studies have explored several approaches to improve energy efficiency, optimize carbon footprints, and adapt workload management in real-time. Here, we highlight some notable research efforts that inform the foundations of our multi-tiered cloud orchestration system.

[1]https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand

1) **Energy-Efficient Resource Allocation and Scaling**
Several researchers have proposed mechanisms to adapt resource allocation dynamically based on workload demand and available energy. Thiede et al. introduced "Carbon Containers" [11], a method for vertically scaling containerized applications, migrating workloads, and utilizing suspend/resume features to enforce carbon emission targets. This work demonstrates that targeted container management can balance performance with environmental goals, setting a precedent for incorporating real-time carbon metrics into workload orchestration.

2) **Dynamic Resource Scaling for Optimized Utilization**
Tyagi and Sharma in "Scavenger" [14] emphasize both horizontal and vertical scaling techniques to match resource availability to real-time demand, optimizing energy efficiency. This research highlights the potential of adaptive scaling techniques to enhance energy savings without sacrificing system performance, forming the basis for dynamic resource management across tiers in our proposed framework.

3) **Power Capping for High-Performance Workloads**
In high-performance computing environments, Zhao et al. demonstrated how power capping mechanisms can effectively reduce energy consumption, particularly for GPU workloads in their work "Sustainable Supercomputing for AI" [13]. By limiting energy use during periods of high carbon intensity, Zhao et al.'s work illustrates how power capping can reduce carbon emissions while preserving performance—a strategy our framework adopts to ensure sustainable operation even under power constraints.
Zhao et al[13]. in Sustainable Supercomputing for AI applies GPU power capping at the HPC scale to reduce power draw and lower GPU temperature during large-scale model training.

4) **Real-Time Orchestration with Energy-Aware Distribution**
Real-time orchestration systems that adapt workload distribution according to energy metrics are exemplified in Zhang et al.'s study, "Lifting the Fog of Uncertainties" [12]. This work leverages real-time monitoring of renewable energy availability, carbon intensity, and workload urgency to optimize workload placement, enabling energy efficiency even under fluctuating conditions. Our system builds on this approach by prioritizing the most energy-efficient tier, ensuring that workloads are assigned based on both energy efficiency and performance requirements.

5) **Visibility and Control Over Energy Sources for Geographic and Temporal Load Shifting**
In *Enabling Sustainable Clouds: The Case for Virtualizing the Energy System*, Bashir et al. [19] propose enhancing cloud sustainability by providing applications with visibility and control over energy sources, enabling real-time decisions based on carbon intensity and renewable availability. They suggest dynamically shifting workloads both geographically and temporally to regions or times with lower carbon intensity, optimizing energy use by aligning workload demands with cleaner power sources.

6) **Green Metrics and Monitoring Tools**
Integrating green metrics for carbon emissions, renewable energy use, and energy efficiency scoring has also been a focus in cloud computing research. For instance, as proposed by Zhang et al. [20] GreenDRL's system monitors a range of metrics in real time, such as energy consumption, cooling system power usage, workload distribution, and renewable energy generation. Monitoring tools, as outlined by various researchers, provide critical insights into the environmental impact of workload distribution and inform the effectiveness of energy optimization strategies. These tools guide our orchestration system in real-time adjustments that enhance both operational efficiency and environmental sustainability.

The collective insights from these studies lay the groundwork for our approach, which advances multi-tiered cloud orchestration by integrating real-time energy data from external sources, including carbon intensity and renewable energy availability, to dynamically allocate resources. By incorporating green metrics alongside traditional cost-based models, our system further aims to provide actionable insights into energy savings and carbon reduction potential, offering a comprehensive energy-optimized solution for modern cloud infrastructures.

III. Proposed System Design

We propose developing a multi-region or multi-availability zone, cloud orchestration system that dynamically allocates resources and schedules workloads based on Service-Level Objectives (SLOs) and real-time energy consumption. The system can further implement tiering within a data center, where workloads can be distributed across energy-efficient, renewable-powered, and high-performance tiers based on urgency, performance needs, and sustainability goals.
The system will evaluate real-time energy data[2] like those provided by the US energy administration[3] and load balancing across tiers to optimize resource usage while minimizing carbon emissions. Additionally, it will provide green metrics, such as carbon savings and energy efficiency, that can be used to incentivize customers alongside traditional cost-based pricing models.

[2]https://www.iea.org/data-and-statistics/data-tools/real-time-electricity-tracker

[3]https://www.eia.gov/electricity/gridmonitor/dashboard/electric_overview/US48/US48

A. Tiered Configurations for Energy-Efficient Workloads

To maximize energy efficiency, we propose a tiered system for workload distribution, defined by specific Service Level Objectives (SLOs) that consider both performance and energy use, inspired by configurations like those proposed by Hu et al. [8].

- **Tier 1:** Renewable Energy Tier – Servers primarily powered by renewable energy sources (e.g., solar, wind), which offer cost-effective, green power but with variable availability.
- **Tier 2:** Energy-Efficient Tier – Designed with energy-optimized hardware for stable, low-power processing, suitable for consistent, lower-intensity workloads.
- **Tier 3:** High-Performance Tier – High-performance, energy-intensive servers for latency-critical tasks, where immediate processing is prioritized over energy savings.

B. Real-Time Energy-Aware Orchestrator

Our proposed energy-aware orchestrator will integrate data from APIs that track real-time energy metrics, such as renewable availability, grid load, and regional energy costs. By analyzing this data, the orchestrator distributes workloads across data centers located in regions with higher power availability or more efficient power use.
Within each data center, workload assignments are adjusted based on SLOs and tier suitability, ensuring that the most energy-efficient resources handle appropriate tasks while high-priority jobs are directed to performance-optimized systems.

C. Load Balancing and Scheduling Algorithms

The orchestrator uses advanced load balancing and scheduling algorithms, building on energy-efficient strategies like those discussed by Hanafy et al. [9]. These algorithms prioritize energy savings by dynamically adjusting workloads between tiers in response to current energy conditions and workload urgency. For instance, non-urgent tasks may be scheduled during peak renewable availability or moved to more energy-efficient tiers to minimize grid dependence.

D. Green Scoring Framework

We propose a Green SLA/Scoring Framework that combines carbon metrics with performance and latency requirements to ensure balanced optimization. This framework sets clear sustainability targets, such as processing a defined percentage of workloads with renewable energy or maintaining carbon emissions below specified levels. Using these green scores, data centers that meet or exceed efficiency targets receive cash incentives and a sustainability score, which drives continuous improvement.
We investigated several techniques to hone in on our proposal like predictive models for ML workloads through smart scheduling as researched by Tapan et al [2] and ML-based waiting time predictions as researched by Pradeep et al [5], where they leverage real-time data about the system to predict the waiting times for the job improving the scheduling speed. We also explored the use of Reinforcement learning models to account for real-time variations in workload changes as researched by Liu et al [3] and Horan et al [4]. We hope to assess the possibility of integrating energy consumption metrics, balancing performance improvements with energy savings. This is the novel aspect through which we aspire to provide creative approaches to managing workloads for energy efficiency.

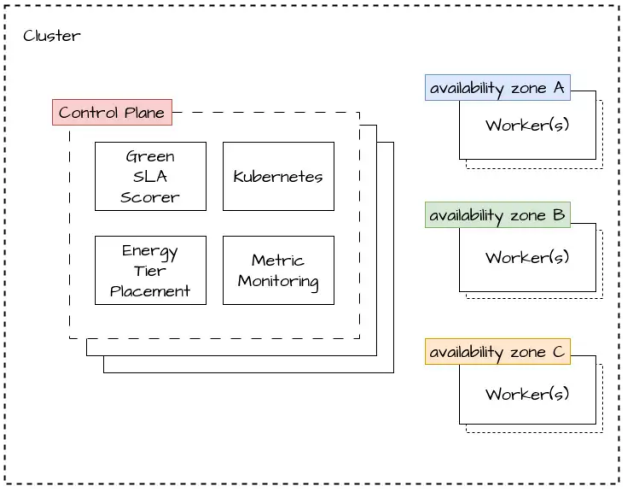


Figure 1. Tentative proposed components of the multi-tiered cloud orchestration system, illustrating the integration of Kubernetes, tiered resource allocation, real-time energy metrics, and green SLA socring.

E. Assumptions and Prerequisites

1) **Data Center Proximity to Energy Sources** It is assumed that data centers are strategically located near grid energy sources, with direct physical connections to these sources to facilitate rapid energy transfer. Proximity to the grid allows each data center to access real-time energy supply data, enabling energy-aware orchestration to dynamically adjust workloads based on availability and energy conditions.

2) **Distributed Energy-Aware Orchestrators with Synchronized Communication** The energy-aware orchestrator is deployed in each data center to handle localized workload scheduling, tier management, and energy monitoring. These distributed orchestrators periodically communicate with one another to share updated information on energy availability, workload distribution, and overall efficiency metrics across the network. This synchronized communication in the distributed environment is essential for maintaining a coherent and efficient energy management strategy across multiple data centers, balancing load and maximizing use of renewable energy sources at a network level.

F. Simulate workloads

To simulate fluctuating workloads that mirror real-world energy consumption patterns, Google's Cluster

Workload Traces[4] from their public repository can be a valuable resource. These traces contain information on job characteristics (e.g., CPU and memory requirements, job duration, and priority) collected from production clusters, which is ideal for creating diverse task profiles.

1) Use Google Cluster Traces to classify tasks by priority (e.g., business-critical vs. energy-efficient) and determine resource demands, simulating realistic workloads with varying urgency and intensity.
2) Create workload patterns that reflect time-based variations, with business-critical tasks peaking during business hours and energy-efficient tasks filling off-peak times.
3) Integrate power traces to model renewable energy availability and grid load, enabling the scheduling system to prioritize tasks based on real-time power conditions.

G. Monitoring and Measurement

To measure the effectiveness of the proposed system, we would like to monitor it for the following metrics:

- Latency, response time, and energy consumption reduction
- Resource utilization - CPU, memory, including energy consumption
- Task failure rates and scheduling efficiency under changing grid conditions

H. Green Metrics Calculation

To evaluate the effectiveness of our system in reducing energy consumption and carbon emissions, we propose calculating a set of green metrics that track energy usage and carbon savings in real time. These metrics can help quantify the environmental impact of our workload scheduling algorithms. The metrics used in our system are inspired by widely accepted energy efficiency models in cloud computing, such as Power Usage Effectiveness (PUE) and Carbon Usage Effectiveness (CUE), which provide a foundational approach to measuring energy efficiency and environmental impact. For example, as discussed by T. N. F. dos Reis et al.[15] in their analysis of energy consumption in Green Cloud Computing, PUE evaluates how efficiently a data center uses energy, while CUE measures the carbon emissions produced per unit of energy consumed. These metrics allow for the comparison of energy efficiency between data centers and help ensure the reduction of CO2 emissions during cloud operations.

1) **Carbon Emission Reduction (CER)** measures the total reduction in carbon emissions achieved by shifting workloads from conventional energy sources to renewable energy-powered tiers. This metric is calculated using the difference in carbon intensity between conventional and renewable sources and the duration for which the workloads are executed in the respective tiers.
   - **CEF** = $(CarbonIntensity_{conventional} - CarbonIntensity_{renewable})WorkloadDuration$
   - $CarbonIntensity_{conventional}$ is like coal, natural gas, oil
   - $CarbonIntensity_{renewable}$ is like solar, wind, hydro
2) **Energy Efficiency Score (EES)** evaluates how efficiently the system utilizes energy resources, calculated as the ratio of workload output to total energy consumed. This metric helps track improvements in energy efficiency when workloads are executed in energy-optimized tiers.
   - **EES** = $WorkloadOutput/TotalEnergyConsumed$
   - $WorkloadOutput$ is like the number of tasks completed, amount of data processed, job completion rate, etc
   - $TotalEnergyConsumed$ is like energy consumed by GPU/CPU
3) **Renewable Energy Utilization (REU)** tracks the percentage of workloads processed using renewable energy sources. This metric ensures that the system adheres to Green SLAs that prioritize renewable energy usage.
   - **REU** = $(WorkloadsOnRenewableEnergy/TotalWorkloads)100$
4) **Cost-Carbon Tradeoff Score (CCTS)** combines cost savings with carbon reductions to provide a hybrid metric that balances environmental and economic goals. The score is a weighted sum of cost savings from energy-efficient scheduling and the carbon savings from using renewable energy sources.
   - **CCTS** = $CostSavings + CarbonSavings$

These metrics can be periodically calculated using real-time data from the integrated APIs and internal monitoring tools (e.g., AWS CloudWatch, PowerAPI). The results are stored in a database and visualized using a dashboard to provide insights into the system's performance and adherence to Green SLAs. The green metrics are also fed back into the workload orchestrator, allowing it to adjust scheduling decisions dynamically based on current energy conditions and sustainability goals.

I. Successful Outcome:

- **Reduces Energy Consumption and Carbon Emissions:** Achieves measurable reductions in energy usage and carbon footprint while meeting Green SLA targets.
- **Maintains Performance SLOs:** Ensures that high-priority tasks consistently meet latency and performance SLOs without significant breaches, despite energy optimization efforts.
- **Improves Data Center Efficiency:** Increases data center utilization, especially in renewable-energy-powered tiers, contributing to overall cost savings.
- **Balances Trade-offs Effectively:** Proves that trade-offs between performance, cost, and sustainability are well-managed, with minimal impact on customer satisfaction.

[4]https://github.com/google/cluster-data

IV. Experiment Setup

A. Evaluation of Energy Data APIs

To facilitate the dynamic allocation of workloads based on real-time energy data, we evaluated multiple Energy Data APIs that can provide data to enable the system to make energy-aware scheduling decisions. Specifically, we are interested in APIs that provide real-time information on carbon intensity, renewable energy availability, and energy consumption. This will allow us to prioritize workload execution in regions or tiers that are powered by cleaner energy sources.

These APIs are integrated into a dedicated service that periodically queries each data source, aggregates the information, and provides it to the workload orchestrator. This integration aligns with recent research on energy-aware scheduling, such as the work on G. L. Stavrinides[21]. These studies emphasize the importance of leveraging real-time energy metrics, including carbon intensity, to make dynamic scheduling decisions that enhance resource efficiency and minimize environmental impact.

1) **Electricity Maps API**[5] provides real-time carbon intensity data by region, allowing the system to shift workloads to regions with lower carbon emissions. We utilize HTTP GET requests (Python) to fetch carbon intensity data periodically, which informs the scheduling decisions for workloads across our multi-tiered system. The data is then stored for real-time analysis by the orchestrator.
   - **Carbon intensity (gCO2/kWh):** This is the primary variable that measures how much CO2 is emitted per unit of energy in a given region.
   - **Renewable energy percentage:** The share of energy sourced from renewables in the energy mix.
2) **WattTime API**[6] offers real-time emissions data, enabling our system to adapt workload schedules according to when cleaner energy is available. By integrating the WattTime API, we ensure that the system can dynamically adjust to fluctuations in energy availability and carbon intensity, maximizing the use of renewable energy.
   - **Real-time carbon intensity (gCO2/kWh):** Provides emissions data specific to energy consumption in the region.
   - **Marginal carbon emissions (gCO2/kWh):** Represents the emissions added by using an additional unit of electricity.
3) **AWS CloudWatch and Azure Monitor** provide real-time energy consumption metrics for cloud environments hosted in AWS and Azure, respectively. These tools allow us to track the energy usage of specific workloads and adjust resource allocation to more energy-efficient tiers when appropriate. For workloads hosted on AWS or Azure, energy consumption data is periodically collected using the respective SDKs and used to guide workload placement decisions.
   - **Energy consumption metrics (kWh):** Measures the total energy consumption of specific workloads.
   - **Cost of energy usage ($):** Provides detailed metrics on energy cost, which can be used in combination with carbon metrics.
   - **CPU/Memory utilization:** Used to assess resource efficiency in handling workloads.

   These APIs are integrated into a dedicated service that periodically queries each data source, aggregates the information, and provides it to the workload orchestrator. The orchestrator uses this data to dynamically schedule workloads across tiers based on real-time energy and carbon metrics, ensuring adherence to Green SLAs while minimizing carbon emissions.

[5]https://www.electricitymaps.com/free-tier-api

[6]https://legacy-docs.watttime.org/#register-new-user

B. Cloud Platform Simulation and Load Balancing Experiments

Our experimental investigation leverages CloudSim, a robust simulation framework developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. This framework enables us to model and simulate cloud computing infrastructures and services, providing a controlled environment for testing our proposed multi-tiered resource management approach.

C. CloudSim

CloudSim is a simulation framework designed specifically for modeling and simulating cloud computing infrastructures and services. It is developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. We will be leveraging this framework to model the cloud computing environment - data centers, virtual machines (VMs), resource provisioning, cloud service broker policies, and scheduling algorithms.

CloudSim's comprehensive feature set aligns perfectly with our research objectives. Here's an overview of its main features:

- **Data Center Modeling**: The framework provides tools to simulate complete data center environments,including servers (hosts), storage, and network capabilities across multiple availability zones and network interconnections. It helps simulate complex scenarios like resource allocation in multi-cloud environments.
- **Workload Management**: Built-in support for modeling container-like workloads and their resource requirements allows us to emulate real-world cloud applications.
- **VM Management** It offers mechanisms to manage the lifecycle of virtual machines, allowing the simulation of VM allocation, migration, and provisioning.
- **Energy-Aware Operations**: CloudSim's native capabilities for energy consumption simulation enable us to

integrate and test power management techniques, making it ideal for our carbon-aware scheduling investigations.

- **Resource Orchestration**: The framework implements flexible scheduling and load balancing mechanisms that we can extend to incorporate our custom environmental metrics.
- **Energy and Power Management** The framework includes capabilities to simulate energy-aware cloud data centers, allowing experimentation with power management techniques to minimize energy consumption.
- **User-Friendly API**: CloudSim provides a user-friendly API that simplifies the task of modeling various aspects of cloud infrastructure, including storage, compute, and network resources.
- **Scheduling and Resource Allocation**: The framework allows users to test various scheduling and resource allocation algorithms to compare their performance under different scenarios.

D. Experimental Setup

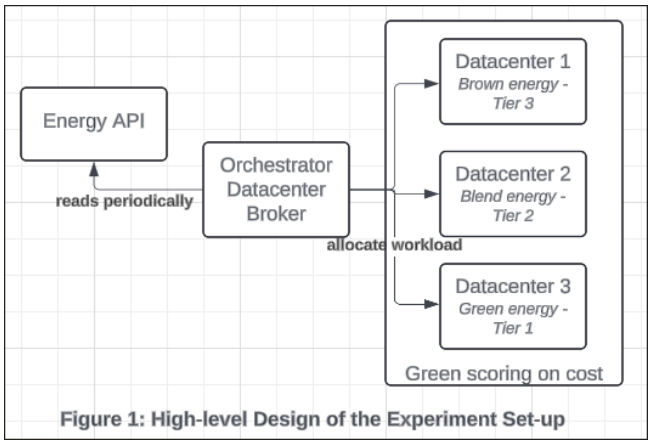


Figure 1: High-level Design of the Experiment Set-up

The BorgPowerMain class is designed to simulate a cloud computing environment using the CloudSim framework. The experiment focuses on creating a datacenter with multiple hosts, each having different configurations, and dynamically allocating resources based on power characteristics. The setup involves the following key steps:

1) Initialization: The CloudSim library is initialized with a specified number of users and simulation parameters. Fossil-free energy percentages are loaded from a CSV file to simulate the availability of green energy over time.

2) Datacenter Creation: Three types of datacenters are created: high-resource, medium-resource, and low-resource datacenters. Each datacenter has hosts with different CPU, memory, and MIPS configurations. The DatacenterFactory class is used to create these datacenters with specific characteristics.

3) Broker Creation: A DatacenterBroker is created to manage the virtual machines (VMs) and cloudlets (tasks) in the simulation. The broker is responsible for submitting VMs and cloudlets to the appropriate datacenters based on the power characteristics.

4) VM and Cloudlet Initialization: Lists for VMs and cloudlets are initialized. The Borg dataset is loaded, and VMs and cloudlets are created based on the dataset rows. Each VM and cloudlet is configured with specific resource requirements.

5) Dynamic Allocation: A monitoring thread is set up to periodically check the fossil-free energy percentage and update the datacenter selection accordingly.

6) PowerAwareDatacenterBroker: Class used to dynamically allocate VMs and cloudlets to the datacenters with the optimal power characteristics.

7) Simulation Execution: The simulation is started, and the CloudSim framework manages the execution of VMs and cloudlets. The simulation runs for a specified period (e.g., 12 hours), during which the broker dynamically allocates resources based on the power data.

8) Results Collection: After the simulation completes, the results are collected and printed. This includes information about the cloudlets' execution status, the datacenters they were executed on, and the resource utilization. Key Components

9) Datacenter Factory: Creates datacenters with hosts having different configurations.

10) PowerAwareDatacenterBroker: Manages the dynamic allocation of VMs and cloudlets based on power characteristics.

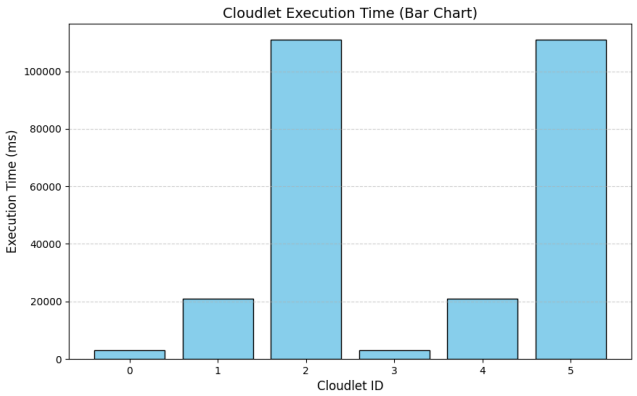
11) Borg Dataset: Provides the data for creating VMs and cloudlets with specific resource requirements.

12) Fossil-Free Energy Percentage: Simulates the availability of green energy over time and influences the datacenter selection.

13) Example Host Configurations:

- Host 1: 2 CPUs, 4 GB memory, 1000 MIPS
- Host 2: 4 CPUs, 8 GB memory, 2000 MIPS
- Host 3: 8 CPUs, 16 GB memory, 4000 MIPS
- Host 4: 1 CPU, 1 GB memory, 500 MIPS

We executed our simulation experiments using Java 21 runtime environment using Maven build system and within a containerized GitHub Actions workflow. The setup could run on GitHub's free-tier compute resources, demonstrating that our simulation framework can run effectively on any standard commodity hardware, making atleast our experiments easily reproducible by other researchers without requiring specialized infrastructure; though applying these algorithms in production would require a real data center or cloud compute infrastructure with power aware features described in other sections.
This simulation platform enables us to conduct controlled experiments that would be challenging to perform in physical cloud environments. We can evaluate how different orchestration strategies affect both computational efficiency and environmental impact while maintaining precise control over all system variables. The framework's energy modeling capabilities are particularly valuable for validating our hypotheses about carbon-aware scheduling and resource management similar to popularly available compute orchestration platforms like Kubernetes, Slurm, etc. The graph here shows the overall execution time for 6 different requests submitted to the system.



Through this experimental setup, we aim to demonstrate how next-generation cloud orchestration systems could incorporate energy emission metrics into their resource management decisions while maintaining application performance requirements. CloudSim provides the ideal foundation for developing and validating these new algorithms, allowing us to optimize for both computational efficiency and environmental impact in a controlled, reproducible environment.

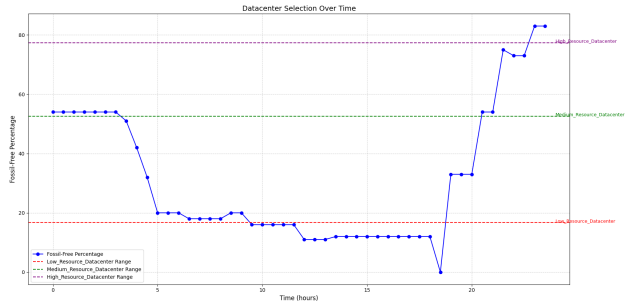
V. Experiment Results

This is our github[7] where we are working on our project. We are working on a fork of Cloudsim[18] for our experiments and our implementation of our project. We will simulate our workloads and our plans within this program.

Currently we have gathered energy metrics from Electricity Maps API and collected real-time information on power usage. There are various metrics collected by this, but we are focusing on fossil free power and renewable power usage for now. In our current model we have created a broker to select data centers based on fossil free power. Currently we have it set up so that if the fossil free percentage is over 70% then a high resource data center will be used, if the power is between 35% and 70% then a medium resource data center will be used, and if the fossil free usage is below 45% then a low resources data center will be used.

The project isn't perfect and it has had a few hiccups, for the WattTime API we are required to pay for a premium version to obtain all of the features and information it provides. It has been implemented, but it has limited what we can obtain and use out of it. Another annoying part of this API is that the provided token expires every 30 minutes and you would have to go in and change the code every 30 minutes with the new API token for this API to work. In the end we are using Electricity Maps API because it provides enough information for our needs so far. The implementation token for this API doesn't expire and we can continue to make requests which is very useful.

Cloudsim runs on it's own clock and doesn't do real time simulation, so it can't get live data from the APIs to use for the simulation since it completes over 24 hours of simulation in 1 second. To address this issue we gathered data from electricity maps api of california over a period of 24 hours and we are using that data to put it into our cloudsim simulation through an entity and run the datacenter changes based on that data of over 24 hours with the cloudsim clock simulation timings. The graph depicts the simulation of the Datacenter selection based on the fossil free percentage.



This data was then fed to the Cloud simulator to vary the energy consumption. We parse the incoming Cloudlets mapped from the Borg Dataset to determine which Datacenter the request should be routed to, based on the percentage of fossil-free power usage and renewable power usage.

We also assigned a cost to each Datacenter such that if it was associated with fossil fuel, a higher penalty was assigned to it versus if the Datacenter was running on green sources of power, a lower penalty was assigned to it. This was done to calculate a green metric.

This is based on the hypothesis that since the Datacenters can't be rewired to different power sources in real time, assuming that some Datacenters draw energy from green power sources, the workloads could be shifted to those Datacenters to calculate the green cost.

We simulated the above model and ran two simulations.

1) One which assumes a roundrobin allocation to Datacenters that don't have energy-based routing.
2) Another that assumes that workloads can be directed to Datacenters that have a higher percentage of green energy availability.

Here is a brief description of the logic in code.

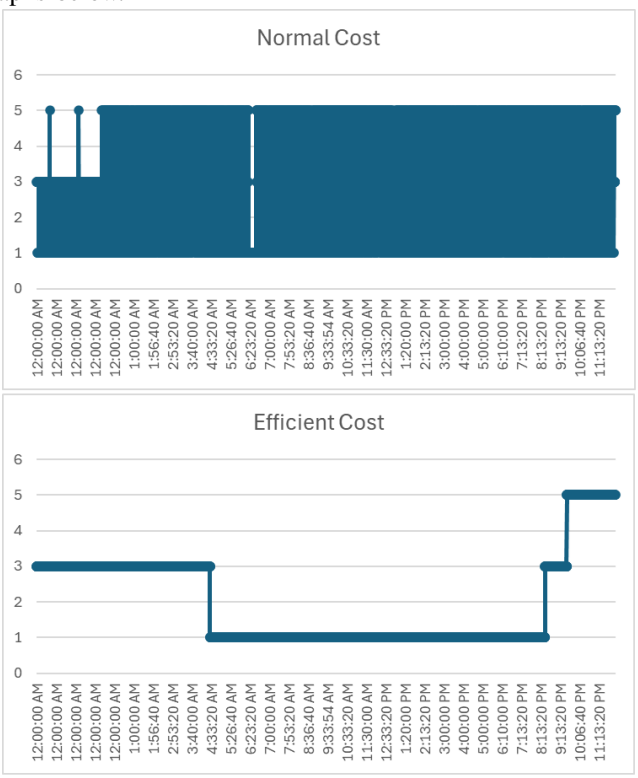
1) Iterating Through the Batch: The method iterates through each row in the batch list. Each row contains data necessary to create a VM and a cloudlet.
2) Creating or Retrieving VMs: For each row, the method checks if a VM with the given instanceIndex already exists in the vmMap. If the VM does not exist, a new Vm object is created with the specified parameters (e.g., mips, pes, memory, etc.).
3) Creating Cloudlets: A new Cloudlet object is created for each row using the specified parameters (e.g., cloudletId, cloudletLength, pes, cloudletFileSize, cloudletOutputSize, etc.). The UtilizationModelStochastic is used for the CPU, RAM, and bandwidth utilization models, indicating that the resource utilization will be stochastic (random).
4) Setting Cloudlet Properties: The brokerId is set as the user ID for the cloudlet, and the VM ID is set as the

[7]https://github.com/swghosh/green-cloud-sim

guest ID. The submission time for the cloudlet is set using the time from the dataset row.

5) Adding Cloudlets to the List: The created cloudlet is added to the cloudletList, which is used later in the simulation to manage and process cloudlets.

Based on the hypothesis above, we noticed that the cost of running the round-robin route was much less efficient than the cost of running with the green power sources. See the graphs below.



VI. Model for Distributed Load Balancing

Future exploration can include a distributed implementation for load balancing, where power stations actively publish alerts when power usage thresholds are breached, or renewable power availability improves significantly. The algorithm could function as follows:

1) Initialization: Each data center subscribes to power station alerts through a publish-subscribe model. Power stations provide real-time updates on power availability and carbon intensity.
2) Alert Reception: When a power station emits an alert (e.g., high renewable availability or usage threshold exceeded), the system captures the event in real time.
3) Decision Making: If an improvement alert is received (e.g., high renewable availability), workloads are redistributed to data centers in regions with better energy metrics.
4) If a threshold breach alert is received (e.g., high non-renewable usage), non-urgent workloads are deferred or migrated to alternate regions with better energy profiles.
5) Load Balancing: A decentralized orchestrator at each data center evaluates workload priorities and energy profiles. It dynamically reallocates tasks based on alerts, Service-Level Objectives (SLOs), and current resource availability.
6) Feedback Loop: Metrics like latency, carbon savings, and resource utilization are continuously monitored. Insights are fed back into the algorithm to refine decision-making over time.

VII. Future Work

The results and findings of this study lay the groundwork for further exploration and development in sustainable cloud computing practices. However, several areas remain open for refinement and expansion to enhance the efficacy and applicability of the proposed system. This section outlines potential directions for future work:

1) Real-Time API Integration and Adaptability: A critical next step involves improving the system's real-time adaptability by leveraging advanced APIs capable of providing live, granular data on energy availability, carbon intensity, and grid conditions. Collaborations with industry stakeholders to standardize and enrich energy data APIs can significantly improve the accuracy and responsiveness of the orchestration system.
2) Integration of Predictive Analytics: Incorporating predictive models and machine learning techniques into the orchestration system can enable anticipatory resource allocation. Predictive analytics could forecast workload demands and renewable energy availability, allowing the system to schedule tasks proactively. Reinforcement learning models, in particular, hold promise for adapting to dynamic workload changes and optimizing energy efficiency over time.
3) Multi-Cloud Collaboration: Expanding the orchestration system to operate seamlessly across multiple cloud providers is a compelling area for future work. This approach would enable the system to leverage the unique energy profiles and capabilities of different providers, maximizing overall sustainability. Establishing interoperability standards and protocols for energy-aware scheduling across cloud platforms will be essential to achieving this goal.
4) Real-World Implementation and Testing: Finally, deploying the proposed system in live cloud environments will provide invaluable insights into its practical viability and performance. Conducting real-world experiments across diverse use cases and industries can validate the system's efficacy and uncover new challenges and opportunities for improvement.

By addressing these future directions, the proposed system has the potential to revolutionize cloud computing by establishing it as a cornerstone of sustainable

technology practices, balancing performance, cost, and environmental impact.

VIII. Conclusion

The experimental results validate the effectiveness of integrating energy-aware scheduling and resource allocation strategies in cloud environments. By leveraging real-time fossil-free energy metrics, the proposed system validated the potential for reducing carbon emissions and improving energy efficiency in a multi-cloud environment. The adoption of tiered data center configurations, combined with dynamic workload orchestration, can enable an optimized balance between performance and sustainability.

Key findings include:

1) Improved resource utilization across high, medium, and low-energy tiers, showcasing the practical applicability of energy-aware cloud orchestration in diverse workload scenarios.
2) Simulation of real-world energy consumption patterns using CloudSim, although an promising framework proved to be quite challenging since there are a few gaps in the underlying implementation.

While the results are promising, challenges such as API limitations and the need for further real-time adaptability were identified. Future work could focus on refining the orchestration algorithms, incorporating more comprehensive energy metrics, and expanding the system's real-time capabilities to enhance operational precision.

References

[1] Ahmed, K.M.U.; Bollen, M.H.J.; Alvarez, M. *A Review of Data Centers Energy Consumption and Reliability Modeling.* IEEE Access 2021,9, 152536–152563.

[2] Tapan Chugh, Srikanth Kandula, Arvind Krishnamurthy, Ratul Mahajan, and Ishai Menache. 2023. *Anticipatory Resource Allocation for ML Training.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 410–426. https://doi.org/10.1145/3620678.3624669.

[3] Jianshu Liu, Shungeng Zhang, and Qingyang Wang. 2023. *μConAdapter: Reinforcement Learning-based Fast Concurrency Adaptation for Microservices in Cloud.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 427–442. https://doi.org/10.1145/3620678.3624980.

[4] Qiu, Haoran, Weichao Mao, Chen Wang, Hubertus Franke, Alaa Youssef, Zbigniew T. Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. *"AWARE: Automate workload autoscaling with reinforcement learning in production cloud systems."* In 2023 USENIX Annual Technical Conference (USENIX ATC 23), 387-402. 2023. https://www.usenix.org/conference/atc23/presentation/qiu-haoran.

[5] Pradeep Ambati, Noman Bashir, David Irwin, and Prashant Shenoy. 2021. *Good Things Come to Those Who Wait: Optimizing Job Waiting in the Cloud.* In Proceedings of the ACM Symposium on Cloud Computing (SoCC '21). Association for Computing Machinery, New York, NY, USA, 229–242. https://doi.org/10.1145/3472883.3487007.

[6] Kuo Zhang, Peijian Wang, Ning Gu, and Thu D. Nguyen. 2022. *GreenDRL: managing green datacenters using deep reinforcement learning.* In Proceedings of the 13th Symposium on Cloud Computing (SoCC '22). Association for Computing Machinery, New York, NY, USA, 445–460. https://doi.org/10.1145/3542929.3563501.

[7] Zhiheng Zhong, Minxian Xu, Maria Alejandra Rodriguez, Chengzhong Xu, and Rajkumar Buyya. 2022. *Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions.* ACM Comput. Surv. 54, 10s, Article 217 (January 2022), 35 pages. https://doi.org/10.1145/3510415.

[8] Y. Hu, R. Ghosh, and R. Govindan, *"Scrooge: A cost-effective deep learning inference system,"* in Proc. ACM Symp. Cloud Comput., 2021.

[9] Hanafy, Walid A. et al. *"Data-driven Algorithm Selection for Carbon-Aware Scheduling."* in HotCarbon Workshop on Sustainable Computer Systems, 2024

[10] Binghao Chen, Han Zhao, Weihao Cui, Yifu He, Shulai Zhang, Quan Chen, Zijun Li, and Minyi Guo. 2023. *Maximizing the Utilization of GPUs Used by Cloud Gaming through Adaptive Co-location with Combo.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 265–280. https://doi.org/10.1145/3620678.3624660

[11] John Thiede, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. *Carbon Containers: A System-level Facility for Managing Application-level Carbon Emissions.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 17–31. https://doi.org/10.1145/3620678.3624644

[12] Yuqiu Zhang, Tongkun Zhang, Gengrui Zhang, and Hans-Arno Jacobsen. 2023. *Lifting the Fog of Uncertainties: Dynamic Resource Orchestration for the Containerized Cloud.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 48–64. https://doi.org/10.1145/3620678.3624646

[13] Dan Zhao, Siddharth Samsi, Joseph McDonald, Baolin Li, David Bestor, Michael Jones, Devesh Tiwari, and Vijay Gadepally. 2023. *Sustainable Supercomputing for AI: GPU Power Capping at HPC Scale.* In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 588–596. https://doi.org/10.1145/3620678.3624793

[14] S. Tyagi and P. Sharma, *"Scavenger: A Cloud Service For Optimizing Cost and Performance of ML Training,"* 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Bangalore, India, 2023, pp. 403-413, doi: 10.1109/CCGrid57682.2023.00045

[15] T. N. F. Dos Reis, M. M. Teixeira and C. De Salles Soares Neto, *"Relation of Energy Consumption in Green Cloud Computing with Big Data,"* 2023 IEEE Green Technologies Conference (GreenTech), Denver, CO, USA, 2023, pp. 280-284, doi: 10.1109/GreenTech56823.2023.10173828.

[16] M. Vasconcelos, D. Cordeiro, G. da Costa, F. Dufossé, J. -M. Nicod and V. Rehn-Sonigo, *"Optimal sizing of a globally distributed low carbon cloud federation,"* 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Bangalore, India, 2023, pp. 203-215, doi: 10.1109/CCGrid57682.2023.00028.

[17] M. Jay, V. Ostapenco, L. Lefevre, D. Trystram, A. -C. Orgerie and B. Fichel, *"An experimental comparison of software-based power meters: focus on CPU and GPU,"* 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Bangalore, India, 2023, pp. 106-118, doi: 10.1109/CCGrid57682.2023.00020.

[18] Remo Andreoli, Jie Zhao, Tommaso Cucinotta, and Rajkumar Buyya, CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments, arXiv:2408.13386 [cs.DC], 2024. https://arxiv.org/pdf/2408.13386

[19] Noman Bashir1, Tian Guo2, Mohammad Hajiesmaili1, David Irwin1 and Prashant Shenoy1, Ramesh Sitaraman1, Abel Souza1, and Adam Wierman3. 2021. Enabling Sustainable Clouds: The Case for Virtualizing the Energy System. In ACM Symposium on Cloud Computing (SoCC '21), November 1–4, 2021, Seattle, WA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3472883.3487009

[20] Zhang, Kuo, Peijian Wang, Ning Gu, and Thu D. Nguyen. "GreenDRL: Managing Green Datacenters Using Deep Reinforcement Learning." *Symposium on Cloud Computing (SoCC '22)*, November 7–11, 2022, San Francisco, CA, USA. ACM, New York, NY, USA.

[21] G. L. Stavrinides and H. D. Karatza, "Energy-Aware Scheduling of Real-Time Workflow Applications in Clouds Utilizing DVFS and Approximate Computations," 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 2018, pp. 33-40, doi: 10.1109/FiCloud.2018.00013.