National College of Ireland

Designing a green scheduler using containers to optimize workload distribution across multiple clusters based the availability of low carbon energy sources

MSc Research Project
Cloud Computing

Saichandan Kondepudi
Student ID: 22184805

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Saichandan Kondepudi
Student ID:	22184805
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Diego Lugones
Submission Due Date:	12/08/2024
Project Title:	Designing a green scheduler using containers to optimize workload distribution across multiple clusters based the availability of low carbon energy sources
Word Count:	8553
Page Count:	23

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

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

Signature:	SAICHANDAN KONDEPUDI
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

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

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

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

Designing a green scheduler using containers to optimize workload distribution across multiple clusters based the availability of low carbon energy sources

Saichandan Kondepudi
22184805

Abstract

The carbon emitted by the data centers these days is the point of notice. It depends on the variety of factors like constant availability of renewable resources in the data centers and others. Thus, scheduling of the jobs in the available data centers has an elongated impact on the emission of carbon intensity. This research tries to create a new algorithm and simulate the deployment of the algorithm on Kubernetes cluster. The algorithm is then compared with the other two existing algorithms and evaluates the parameters like energy efficiency, job scheduling, Global resource utilization while maintaining high job scheduling percentage. The evaluation was conducted with series of five different sets of available carbon intensities and resources to evaluate the algorithms under different conditions. In every evaluation CAKS algorithm achieved the highest percentage of green utilization ration and achieved better results in scheduling the jobs in each data center. However, research also highlights the drawbacks and areas of the improvement required for future studies. Additionally, the paper suggests the real world implementation of this algorithm to get real time analytics. This research contributes in the concept of green computing by demonstrating the benefits of job scheduling algorithms.

1 Introduction

Several studies have been done in the various aspects of energy efficient and carbon aware computing. Bahreini et al. (2023) and Bahreini and Tantawi (2023)introduced algorithms which are designed to optimize workload placement across geographically distributed data centers, exploiting temporal and spatial variations in energy availability to enhance the utilization of renewable energy. Similarly, WU et al. (2020), Sarkar et al. (2024) and Breukelman et al. (2024)highlighted the benefits of shifting non urgent workloads to periods when renewable energy is more abundant, demonstrating significant reductions in carbon emissions without compromising service quality. Despite these advancements, there remains a gap in real time, dynamic scheduling solutions that can seamlessly integrate with modern cloud environments like Kubernetes.

This research aims to address this gap by developing a Green Scheduler for Kubernetes. The proposed scheduler will dynamically manage and allocate the workloads based on real time data about renewable energy availability and carbon emissions, that optimizes the environmental performance of data centers. By using Kubernetes container and monitoring it's capabilities, the scheduler will ensure that workloads are scheduled to

minimize carbon footprints also while maintaining high levels of efficiency and performance. Below Figure 1 shows the carbon emitted by countries across the world.The data s generated by electricity maps [1]

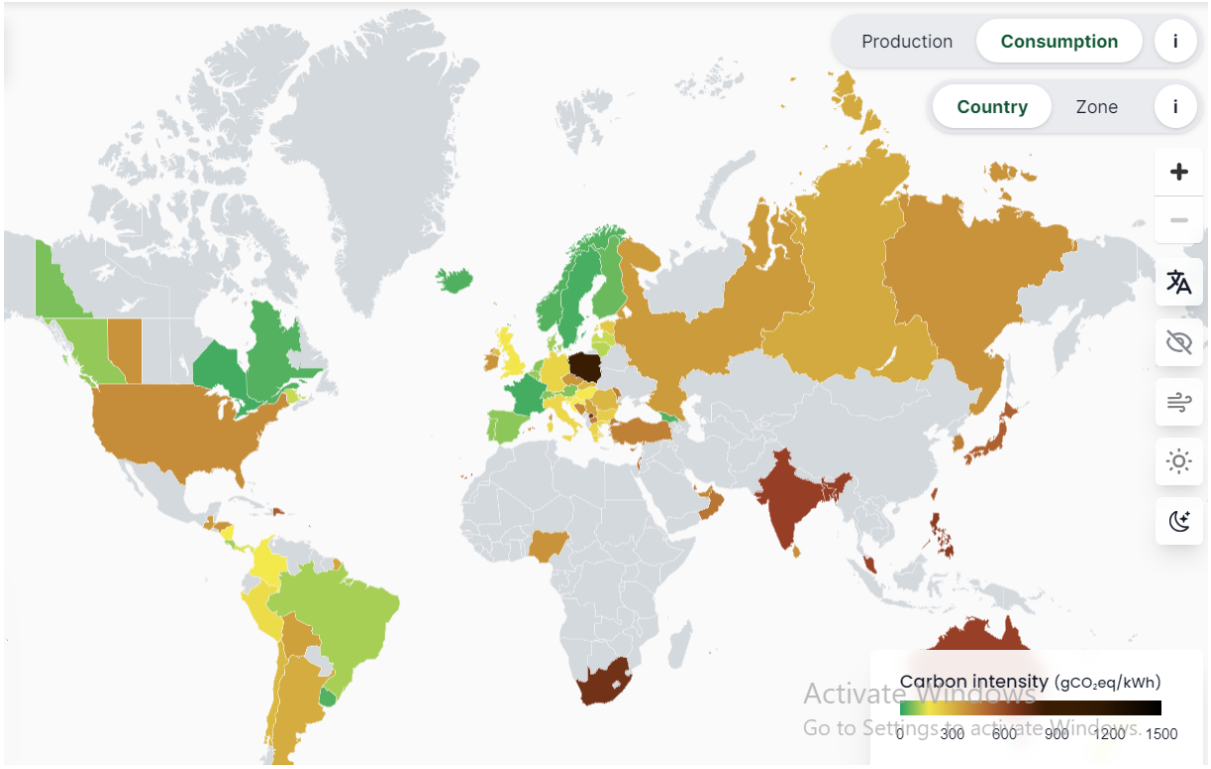

Figure 1: Carbon Intensity by each country across the world

The central question driving this particular research is: **How can a green scheduler can be designed to effectively optimize workload distribution across multiple clusters based on the availability of low-carbon energy sources?** This question enquires the commitment towards advancement of cloud computing technologies with more sustainable practices. It attempts to contribute significantly towards reducing the global carbon footprint of digital operations. The proposed solution involves designing and implementing a green scheduler integrated with Kubernetes. This scheduler will use real time data on energy availability and carbon intensity to inform scheduling decisions, dynamically allocating workloads to optimize for environmental sustainability. The scheduler will integrate the advanced algorithms capable of responding to the intermittent nature of renewable energy, thereby maximizing its utilization.

1.1 Structure of the Report

1. Introduction- It gives the brief idea of the project
2. Related Work- It describes about all he previous works done in his field.
3. Methodology- It describes about the techniques that should be followed to achieve the goal.

[1] `https://app.electricitymaps.com/map`

4. Design Specification- This section tells he design patterns that should be followed while implementing the research work.
5. Implementation- This section tells about how exactly the research work was implemented.
6. Evaluation Results- This content tell about the results that are achieved while performing the analysis.
7. Conclusion- This section concludes the paper by comparing the results and analyzing it.

2 Related Work

Nowadays, data centers are being used much more frequently used, which demands their ability to provide scalable, on demand computing resources. However, there are a lot of hurdles because of the environmental problems caused by data centers, particularly the carbon emissions from intensive operations. According to WU et al. (2020), data centers are predicted to use around 1% of the world's electricity, which makes a major contribution to CO2 emissions worldwide. The shift to green computing, which demands the methods for creating and learning about energy efficiency and lowering carbon footprints, is therefore a crucial field of study.

Initiatives that reduce the effects are being taken by the industry, as seen by Apple's solar energy deployment[2]. However, academic study continues to be essential for discovering and improving data center operations worldwide. Many approaches and techniques have already been developed to attempt to overcome these difficulties, some of which they are discussed below. In order to create a green scheduler that optimally uses low carbon energy sources in a Kubernetes environment, this research will attempt to evaluate the existing literature on carbon aware and energy-efficient scheduling in cloud computing. It will also introduce an approach from multiple studies that will look at the effects of carbon emissions.

2.1 Innovative Use of Renewable Energy

The study paper Akoush et al. (2011) shows an architecture which finds data centers with renewable energy sources in order to make use of the renewable energy that is lost . Specialized network connections and dynamic workload management are used in this method to adjust to fluctuating energy availability., It also optimizes the energy usage and decreases the dependency on fossil fuels by dynamically managing workloads according to the availability of renewable energy by using virtualization and networking technology. This design makes use of the underutilized renewable energy sources to facilitate the migration of virtual machines across data centers based on energy availability. By improving energy utilization and minimizing the dependency on fossil fuels, this methodology achieves the goals of green computing. The architectural solution in this paper offers a practical illustration of how renewable energy can be easily implemented in data center operations, which enhances the previously described energy and carbon available algorithms. It fits in the gaps in the current energy management strategies for cloud computing by providing a concrete framework that can be customized for creating a renewable energy aware scheduler within Kubernetes Pods applications.

[2]https://www.cultofmac.com/191838/

2.2 Green-Aware Workload Scheduling

The research Chen et al. (2012) focuses on minimizing the brown energy consumption by dynamically allocating the tasks across the available data centers Allocation depends on renewable energy availability and cooling requirements of each data center. Approach in this research paper is to utilize a dynamic scheduling algorithm that contains geographical and temporal availability in renewable energy supply and external cooling factors. This allows for reductions in non renewable energy use by placing workload processing with the availability of green energy across the different locations. This method highlights the potential of adding environmental variables directly into cloud scheduling procedures. It offers a model for effectively using renewable energy in data center operations, which can be crucial for developing a green scheduler in Kubernetes environments.

2.3 Eco-Aware Power Management in Data Centers

An paper by Deng et al. (2016), worked on an important strategy for sustainable data centre administration. Main focus of this paper is on how to improve the energy efficiency of data centres by adjusting power usage dynamically based on available green energy and real time workload requirements. They have introduced online power management strategies which aligns with environmental objectives. Without the compromise of service quality and performance, carbon emission can be reduced. For energy management, eco aware strategies offer some framework. They frequently lack the comprehensiveness of carbon emission tracking that is necessary for specific reduction. This approach do not work well with Kubernetes which play as an crucial component in cloud infrastructure.

2.4 Advanced Resource Management in Cloud Computing

This research paper Chen and Lu (2020) states about Generally Weighted Moving Average (GWMA) as an part of dynamic virtual machine allocation to detect host overloading in cloud environments. To increase the chances to determine about when the hosts are overloaded, this technique optimizes virtual machine migration and boosts the energy efficiency. The GWMA Algorithm makes use of previous data to accurately predict and control the loads of the host to assist it for efficient virtual machine migrations. It also tries to minimize the possibility of SLA violations. It includes application that has Maximum Correlation (MC) and Minimum Migration Time (MMT) for M placement and selection in order to assure that migrations reduce downtime and resource waste. To match the purpose of green computing, the GWMA algorithm and other techniques mentioned in the paper proposes a framework Which controls data center dynamically. This method is relevant to the creation of a green scheduler for Kubernetes, which could benefit significantly from the inclusion of these resource management techniques.

2.5 Economic Influences on Carbon Intensity

This research paper Mahmood (2023) proposes Autoregressive Distributive Lag (ARDL) model which tells how foreign direct investments, natural resource ren and economic expansion affect the carbon intensity in Saudi Arabia. Method included in the paper highlights the complexity of environmental management in cloud computing and offers a comprehensive knowledge of how economic activities influence environmental consequences. ARDL is used to examine both short and long term effects. Linear analysis is applied to

assess the asymmetric effects of economic factors. It provides an approach that can be used to dvelope the algorithms that can dynamically schedule workloads in cloud environments. Even with the other external factors in align, these economic models show how crucial it is to manage cloud resources. By using real time energy availability and demand, using these techniques in cloud computing, sustainable operations can be achieved

2.6 Carbon Aware Scheduling Approaches:

By making use of real time energy availability and demand in cloud computing, sustainable operations can be achieved. Research by Bahreini et al. (2022), they discuss about dynamic scheduling algorithm which is based on carbon intensity measure which is crucial for environmental sustainability. This is done to fulfil the objective of intelligent workload distribution to lower carbon footprints. This algorithm[3] takes advantage of change in green energy supply throughout the data centre to produce optimal solution. This was created to handle NP hard nature of the problem. The used of LP was based on approximation technique which shows how difficult is to schedule the lowest possible carbon emission. This is then compared to a different strategies which prioritizes cost cutting or computational efficiency without explicitly taking environment into account. Now, workload scheduling is dynamically altered by Google's Carbon Intelligent Computing System(CICS) to match with time when the intensity or emission of carbon is low. All the proposed solution work well but they don't adjust in real time in response to unanticipated changes in energy demand or availability. When data centres are frequently allocated in different geographic regions with variable access to renewable energy scores, but this algorithm has capacity to manage the dynamic job scheduling adjustments and the variations in power sources emission of carbon is also handled.

2.7 Carbon-Aware Workload Dispatcher

This research Bahreini et al. (2023) talks about the carbon aware scheduling algorithms that decrease the carbon footprint of data center operations Their investigation presents two crucial algorithms: The first is Randomized Rounding Approximation Algorithm (RRAPX) that is a method for approaching scheduling problems which solves a linear programming relaxation and then uses randomized rounding to get the optimal solutions. Another one is SRRAPX, a sample based rounding algorithm method that is an improvement over RRAPX. It samples several possible solutions and chooses the best one, which reduces carbon emissions and improves it to enhance its limitations. These methods show how algorithmic techniques can be used to significantly decrease emissions. But they lack the ability to adjust in real time to changes in the carbon intensity of the grid, which is a crucial component of genuinely dynamic scheduling regimes. This gap provides up the possibility to the investigation of eco-aware scheduling techniques that handle more general energy management issues in addition to carbon-aware scheduling.

[3] `https://github.com/sustainablecomputing/caspian?tab=readme-ov-file`

2.8 Summary of all the previous works

Table 1: Summary of all the previous works

Article Name	Methodology	Research Domain	Achievements	Limitations
Free Lunch	Colocating datacenters with renewable energy sources	Renewable Energy Utilization in Datacenters	Developed an architecture for using otherwise wasted renewable energy, demonstrated viability with case studies.	High dependency on the availability of renewable energy; technical challenges in migration and storage synchronization.
GWMA Algorithm	Generally Weighted Moving Average (GWMA) for host overload detection	Cloud Computing, VM Migration	Improved energy efficiency and reduced SLA violations compared to other methods; demonstrated significant reduction in VM migrations.	Limited evaluation to specific workloads and configurations; the approach may not generalize well to all cloud environments or workloads.
Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters	Dynamic power management and load scheduling	Green Cloud Datacenters	Achieved notable energy savings while managing workloads in cloud datacenters.	Focuses on specific types of workloads and datacenter configurations; may not address all types of power management scenarios.
A Carbon-aware Workload Dispatcher in Cloud Computing Systems	Workload dispatching based on carbon footprint	Cloud Computing, Carbon Footprint Reduction	Reduced carbon emissions through workload scheduling based on the carbon intensity of different data centers.	The approach is highly dependent on accurate real-time data of carbon intensity and might not be applicable in all geographic locations.
Green-aware Workload Scheduling in Geographically Distributed Data Centers	Workload scheduling considering geographical distribution	Geographically Distributed Datacenters	Improved energy efficiency and reduced operational costs by leveraging geographical distribution and time-zone differences.	The approach may face challenges in real-time workload prediction and scheduling, as well as in handling dynamic changes in energy availability across regions.

An Approximation Algorithm for Minimizing the Cloud Carbon Footprint through Workload Scheduling	Approximation algorithm for workload scheduling	Cloud Computing, Carbon Footprint Minimization	Developed an algorithm to minimize the cloud's carbon footprint through workload scheduling; achieved reduced energy consumption and emissions.	The algorithm's effectiveness is dependent on accurate prediction models and may require frequent recalibration to adapt to changing conditions in cloud environments.
Evaluating the Impact of Green Energy Availability on the Profitability and Sustainability of Cloud Providers	Analytical modeling and case study analysis	Cloud Computing, Green Energy Utilization	Analyzed the profitability and sustainability of cloud providers when integrating green energy, highlighting potential economic and environmental benefits.	The study is limited by its reliance on specific case studies and assumptions about energy prices and availability, which may not generalize to all cloud providers.

3 Methodology

This research focuses on developing a green scheduler for KubernetesCheng et al. (2023) to optimize workload distribution based on the availability of low-carbon energy sources. The methodology involves several stages, including requirement analysis, algorithm development, scheduler integration with Kubernetes, experimental setup, testing and evaluation, and deployment.

1. Requirement Analysis The first step involves identifying the requirements for the green scheduler. The research utilizes the data from the given base paper. The base paper includes the data for which the carbon emission has been included for the given algorithms. This paper will be using the same carbon emission for the Cakss algorithm to compare the results with other two algorithms. The metrics will be calculated for different number of jobs that and it will be represented on the graphical basis.

2. Algorithm Development of Kubernetes Pods Assignment Using Custom Pod Autoscalers[4]

As stated in Ascensão et al. (2024)Kubernetes, a robust container orchestration platform, provides mechanisms for managing the lifecycle of containerized applications across a cluster. One of the key features of Kubernetes is its ability to scale applications dynamically based on demand. This is typically handled by Horizontal Pod Autoscalers (HPA) that adjust the number of pods in a deployment based on observed metrics. However, for more complex scaling requirements, such as integrating custom metrics or advanced scaling logic, Custom Pod Autoscalers (CPA) are used. Custom Pod Autoscalers are the kubernetes pods that runs within the cluster and manage resources just like deployments. Custom Pod Autoscalers abstract the Kubernetes API interactions, allowing developers to focus on the scaling logic. Each CPA includes a base program that handles API interactions and executes custom logic provided by the developer via shell commands.

[4] `https://discuss.kubernetes.io/t/custom-load-balancing-in-kubernetes/16273`

Structure and Functionality

- Base Program: It manages Kubernetes API interactions and triggers custom logic by executing shell commands with piped information.
- Custom Logic: Developers provide custom logic scripts in any language (e.g., Python, Go) to gather metrics and evaluate scaling decisions. The custom logic is responsible for:

  Metric Gathering: Collecting metrics from Kubernetes metrics API or other sources.

  Evaluation: Deciding the number of replicas needed based on gathered metrics.

Workflow
The CPA receives input (such as Pod or Deployment JSON) through a shell command. Custom scripts calculate or retrieve necessary metrics and output the results. The base program reads these outputs, determines scaling actions, and interacts with the Kubernetes API to adjust the number of pod replicas accordingly.
To simplify the setup and management of CPAs, an operator can be used. The Custom Pod Autoscaler Operator automates the provisioning of necessary Kubernetes resources (e.g., Roles, Service Accounts) to run CPAs efficiently. By defining a CustomPodAutoscaler in YAML, developers can deploy it to the cluster with minimal manual configuration. The Horizontal Pod Autoscaler (HPA) can be re implemented as a Custom Pod Autoscaler to leverage custom metrics and advanced scaling logic. This approach retains the familiar features of HPA while extending its functionality to meet specific scaling requirements. When deploying services that require external accessibility, an external load balancer can be created to provide an IP address that routes traffic to the appropriate cluster nodes. This is particularly useful for services that need to be accessible from outside the cluster.
Steps to Create an External Load Balancer[5]*:*

- Define Service with LoadBalancer Type. Add the type LoadBalancer line to the manifest the service.
- Using kubectl to Create the Service: Alternatively, use the kubectl expose command with the –type=LoadBalancer flag to create the service.
- Finding the IP Address and retrieve the external IP address assigned to the service by running:
- Preserving Client Source IP
- To ensure the client's original IP address is preserved, configure the .'spec.externalTrafficPolicy' field in the Service manifest. Setting it to the local preserves the client source IP and avoids a second hop for LoadBalancer and NodePort type Services.

Garbage Collecting Load Balancers Kubernetes v1.17 introduced Finalizer Protection for Service LoadBalancers to prevent orphaned cloud resources. A finalizer ensures that a Service resource is not deleted until the corresponding load balancer resources are cleaned up. By implementing these methodologies, Kubernetes can efficiently manage and scale

[5] `https://discuss.kubernetes.io/t/custom-load-balancing-in-kubernetes/16273`

applications, ensuring optimal resource utilization and maintaining high performance even under varying load conditions.
3. Scheduler Integration to develop algorithm with Kubernetes and custom Scheduler Development and Extend Kubernetes with a custom scheduler written in python or java. Then, Implement resource requests and limits to guide workload distribution. Then, Configure Kubernetes priority classes to favor low-carbon energy usage.
4. Experimental Setup Set up a test environment to validate the scheduler Test Cluster Deployment: Deploy a Kubernetes cluster for testing. Workload Simulation: Use synthetic workloads that mimic real world applications. Incorporate real time and previous carbon intensity data.
5. Testing and Evaluation
We need to evaluate the scheduler's performance: Gather data on energy consumption, carbon emissions, and workload performance. Compare the green scheduler with traditional scheduling approaches.Analyze performance data to identify strengths, weaknesses, and areas for improvement.
6. Deploy the scheduler in a real-world environment and implement the scheduler in a live Kubernetes environment. Continuously monitor performance and adjust as necessary. Document the deployment process and outcomes for future reference.

To Creating an External Load Balancer, we need to Create a Kubernetes Service with type LoadBalancer to automatically create an external load balancer with an externally-accessible IP address. Use YAML manifest or kubectl expose command to set up the Service.Retrieve the load balancer's IP address using kubectl describe services. Configure externalTrafficPolicy to Local to preserve client source IPs. Use finalizers to ensure proper cleanup of load balancer resources upon Service deletion. Measure reduction in energy consumption and quantify the reduction in carbon footprint. Assess impact on workload performance and system responsiveness and evaluate the scheduler's ability to handle varying workloads. Ensure compliance with data privacy regulations. Assess both positive and negative impacts on the environment. Avoid biases in resource distribution among tasks and users.

4 Design Specification

Carbon Aware Kubernetes Scheduler (CAKS)

Current methods are designed to solve a wide range of optimization problems. However, the CAKS problem involves specific constraints like minimizing carbon intensity, scheduling based on real-time data, and managing resource availability dynamically across multiple data centers. These real-time characteristics and resource limitations required a custom-designed solution tailored to green computation needs.General optimization algorithms may not be efficient for problems requiring real-time decision-making due to their computational complexity but CAKS is designed to work efficiently within real-time constraints, scheduling jobs dynamically based on real-time carbon intensity and resource availability across geographically distributed data centers. The goal of CAKS is to reduce carbon emissions by factoring in the carbon intensity of different data centers while maintaining efficient resource usage. Both RRAPX and SRRAPX rely on fractional or randomized approaches, which, although simpler, can lead to suboptimal decisions and increased execution times due to the need for re-evaluation or adjustments when resources

are insufficient. CAKS's more structured approach results in better performance in terms of both scheduling time and energy usage.

Below is the proposed algorithm and description of it.

Notation

- T: Time horizon for executing the jobs.
- M: Set of m data centers (Kubernetes clusters).
- N: Set of n jobs currently in the queue.
- B_{jt}: Available capacity of data center j in time slot t.
- I_{jt}: Carbon intensity of data center j in time slot t.
- r_i: Computational resource requested by job i.
- R_t: Total CPU demand in time slot t.
- l_i: Duration time of job i.
- d_i: Deadline to complete job i.
- x_{ijt}: Binary variable indicating if job i is started on data center j in time slot t.
- γ: Prioritization factor.

Algorithm: Carbon-Aware Kubernetes Scheduler (CAKS)

Input: $(\mathcal{N}, \mathcal{M}, \mathcal{T})$

1. **Initialize Variables:**
   - $X_{\text{best}} \leftarrow \{0\}$
   - $\text{GRU}_{\text{best}} \leftarrow 0$
   - $F_{\text{best}} \leftarrow N$

   The algorithm initializes variables to keep track of the best scheduling matrix, the best GRU value, and the best number of failed jobs.

2. **Fetch Real-time Data:**
   - For each data center $j \in \mathcal{M}$:
     - $I_j^t \leftarrow \text{fetchCarbonIntensity}(j, t)$
     - $B_j^t \leftarrow \text{fetchResourceAvailability}(j, t)$

   The algorithm fetches real time data for carbon intensity and resource availability for each data center and each time slot.

3. **Preprocess Jobs:**
   - Sort jobs $i \in \mathcal{N}$ based on their deadlines d_i and resource requirements r_i.

Jobs are sorted based on their deadlines and resource requirements to prioritize jobs with more stringent deadlines and higher resource needs.

4. **Initial Scheduling:**

   - For each job $i \in \mathcal{N}$:
     - scheduleJob(i)

   Each job is scheduled using the scheduleJob function, which aims to find the best time slot and data center that minimizes the carbon intensity while satisfying resource constraints.

5. **Function scheduleJob(i):**

   - minCost $\leftarrow \infty$
   - $(j^*, t^*) \leftarrow$ null
   - For each data center $j \in \mathcal{M}$:
     - For each time slot $t \in \{0, \ldots, T - l_i\}$:
       - If $B_j^t \geq r_i$ and $t + l_i \leq d_i$:
         - Calculate cost: cost $\leftarrow \sum_{t'=t}^{t+l_i-1} I_j^{t'}$
         - If cost $<$ minCost:
         - minCost $\leftarrow$ cost
         - $(j^*, t^*) \leftarrow (j, t)$
   - If $(j^*, t^*) \neq$ null:
     - Assign job i to data center j^* at time slot t^*
     - Update $x_{ij^*}^{t^*} \leftarrow 1$
     - For each $t' \in \{t^*, \ldots, t^* + l_i - 1\}$:
       - $B_{j^*}^{t'} \leftarrow B_{j^*}^{t'} - r_i$

   The function iterates over all data centers and time slots to find the best slot that minimizes the carbon cost for the given job. If a feasible slot is found (i.e., the data center has enough resources and the job can complete before its deadline), the job is assigned to that slot, and resource availability is updated.

6. **Calculate GRU:**

   - GRU $\leftarrow \sum_{j \in \mathcal{M}} \sum_{t \in \mathcal{T}} \frac{\sum_{i \in \mathcal{N}} x_{ij}^t r_i}{I_j^t}$

   The GRU is calculated by summing the resource usage divided by the carbon intensity across all scheduled jobs, data centers, and time slots.

7. **Update Best Solution:**

   - If GRU $>$ GRU_{best}:
     - $\text{GRU}_{\text{best}} \leftarrow$ GRU
     - $X_{\text{best}} \leftarrow X$

If the current scheduling provides a higher GRU than the best found so far GRU and scheduling matrix are updated.

8. **Output:** $(\text{GRU}_{\text{best}}, X_{\text{best}})$

The algorithm outputs the best GRU value and the corresponding scheduling matrix.

This algorithm offers several key differences when compared to other two algorithm CAKS explicitly aims to minimize carbon emissions by leveraging real time carbon intensity data whereas, RRAPX and SRRAPX focus on approximating an optimal job allocation solution with a primary goal of minimizing computational costs. CAKS uses a direct approach to assign jobs to the least carbon intensive slots while ensuring resource and deadline constraints. RRAPX uses a fractional assignment followed by deterministic rounding. SRRAPX enhances RRAPX by incorporating stochastic elements for rounding and performs multiple iterations for improved results. CAKS continuously fetches real time carbon intensity and resource availability data, making it highly adaptive to changing environmental conditions. RRAPX and SRRAPX primarily focus on pre computed fractional solutions and rounding procedures, without real-time data integration. CAKS outputs the best GRU and corresponding scheduling matrix based on carbon intensity optimization. RRAPX and SRRAPX provide job schedules optimized for computational cost, potentially leading to different resource utilization patterns.
CAks scheduler is designed to be compatible with Kubernetes as described in Poulton (2024), leveraging container technology for efficient workload management and resource utilization. Workloads are encapsulated within containers, enabling seamless deployment, scaling, and management across different clusters. By focusing on reducing carbon emissions and adapting to real time data, CAKS offers a unique approach to job scheduling in distributed computing environments, distinct from the cost centric methods of RRAPX and SRRAPX.

5 Implementation

Since establishing real time data centres and deploying the algorithm on kubernetes requires lot of cost, energy and lots of time, this project was simulated in IFogsim using Java language as here are several perks of simulating it in Ifogsim as described in Agrawal and Singh (2023) and Yousuf Khan and Rahim Soomro (2022). It is mostly suitable for simulating container based algorithms of Fog Computing. The entire environment of the project has been simulated using IFogSim tool[6]. In Kubernetes, workloads are managed using pods, which run containers. In iFogSim, these were mapped to application modules. Each module represents a specific component of an application, and Kubernetes pods were mapped to iFogSim's application modules, allowing the management of containerized workloads.Kubernetes nodes represent the physical or virtual machines that run workloads. In iFogSim, these were mapped to fog devices. Each fog device in iFogSim simulates a node where modules (pods) are deployed, and these nodes are interconnected in a hierarchical fog architecture. Kubernetes' scheduling policies were integrated into iFogSim by customizing the module placement policies (ModulePlacementMapping and ModulePlacementEdgewards). This allowed the simulation to handle Kubernetes-style scheduling, where containers are placed based on resource availability, CPU, memory,

[6] `https://github.com/Cloudslab/iFogSim/releases/tag/v2.0.0`

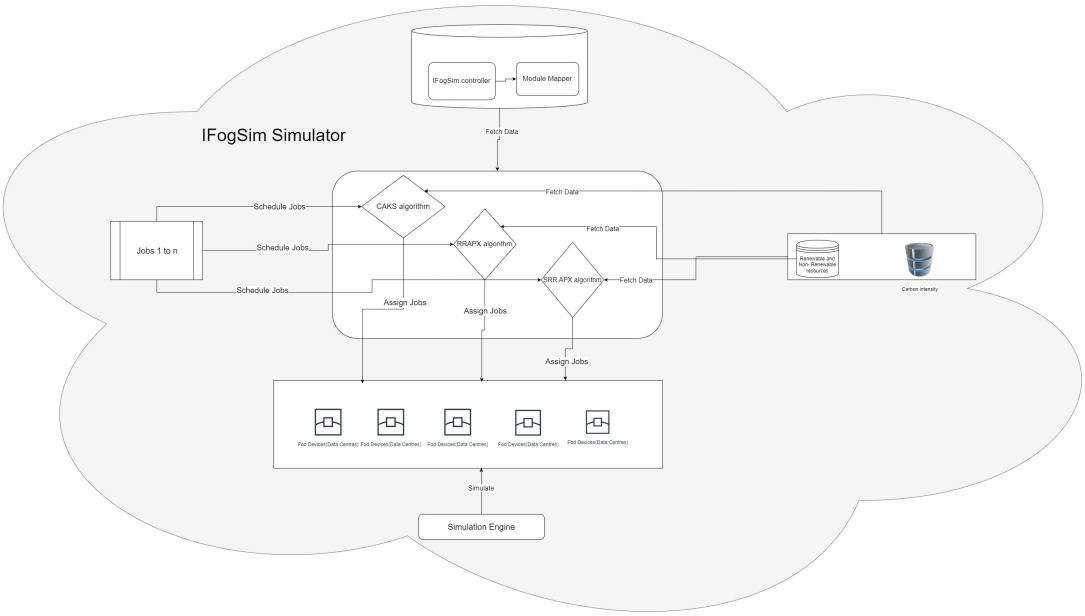

Figure 2: Architectural diagram

and custom policies like minimizing carbon footprint or energy consumption. Kubernetes dynamically manages resources like CPU and memory across its nodes. iFogSim's AppModuleAllocationPolicy was extended to use Kubernetes' logic, ensuring that workloads are scheduled on fog devices based on real-time resource monitoring and requirements. The integration allowed for the simulation of container orchestration across a distributed fog environment, where latency, resource constraints, and workload demands are crucial factors. iFogSim is specifically designed to simulate fog computing environments, which involve the deployment of applications and services closer to the edge of the network. iFogSim's architecture is built to handle such distributed scenarios, making it a more suitable tool than CloudSim for this project

iFogSim includes advanced models for both energy consumption and network latency, which are critical for fog computing simulations. These models allowed us to calculate the carbon intensity and energy consumption across multiple fog nodes in our study. CloudSim, while efficient in simulating large-scale cloud environments, does not provide the same level of granularity for modeling energy consumption and latency in fog or edge environments, which are key considerations in our project.

5.1 Simulated Architecture

Attribute	Value
System Architecture	x86
Operating System	Linux
Virtual Machine Manager	Xen

Table 2: Architecture used in Simulation

5.2 Main function

The main function the entire simulation setup and execution, from initializing the jobs and fog devices to configuring the application modules and running the scheduling algorithms. The 'CloudSim.init' method is called to initialize the CloudSim library, which provides the underlying simulation infrastructure, including the simulation clock and event management system. Then, the 'initializeJobs' method is invoked to create a list of jobs that need to be scheduled. Every job has given defined parameters - required MIPS (Million Instructions Per Second), duration, and deadline that are stored statistically. After the initialization, the 'createFogDevices' method is called up to set up the fog devices which represents Kubernetes clusters and the cloud data centers. All the fog device is created with number of computational resources, memory, bandwidth and storage. The 'createApplication' method defines the application modules and its interconnections. This method sets up the "client", "scheduler" and "cloud" modules to simulate the components of a distributed application running on the fog and cloud infrastructure. A Controller object is instantiated to manage the fog devices and application modules. The controller is responsible to deploying the application onto the fog devices based on the module mapping.

5.3 RunCAKSSimulation function

This functions reflects the algorithm that is mentioned in the above section. This function is a critical component of the simulation which is designed to evaluate the CAKS algorithm's performance in terms of minimizing carbon intensity while scheduling jobs across multiple data centers. It aims to find the optimal scheduling strategy that minimizes the carbon footprint by utilizing available resources.
The function begins by initializing variables to keep track of the best GRU (Green Resource Usage) and the current GRU with matrices to store scheduling information. The 'schedulingMatrix' records the resource allocation for each data center and the 'bestSchedulingMatrix' keeps track of the best scheduling configuration found during the entire simulation. The main task of the function involves iterating over each job in the list of given jobs to be scheduled. Based on the carbon intensity and resource availability for each job, the 'scheduleJobCAKS' function is called to find out optimal data center and start time for the job, The scheduling matrix records the work's resource requirements, and the resource availability matrix deducts the relevant resources once a valid assignment is done. The energy consumption for the job is calculated and added to the total energy consumption of the assigned data center. This ensures that the simulation records the energy usage associated with running the job, contributing to the overall carbon footprint. After all jobs have been processed, the function calculates the GRU for the current scheduling configuration using the 'calculateGRU' function. If the current GRU is higher than the best GRU found so far, the best GRU and best scheduling matrix are updated accordingly. Then, the function returns the best GRU that we get during the simulation, representing the most carbon efficient scheduling strategy found for the given jobs and data centers.
The 'scheduleJobCAKS' function is responsible for determining the best data center and start time for a given job.Main aim of this function is to minimize the carbon cost and satisfy resource constrains. This function iterates over all data centers and possible start times within the time horizon. It checks the feasibility of scheduling the job at each time slot. For each combination of data center and start time, the function calculates the total

carbon cost incurred by running the job. If the data center has sufficient resources and if the job can be completed before its deadline, then carbon cost is computed by adding the carbon intensities over the job's duration. The function also keeps track of the minimum carbon cost used and the corresponding data center and start time. If a feasible and less costly assignment is found, it updates the best assignment accordingly. Finally, the function returns the best assignment of data center and start time for the job, or null if no feasible assignment is found. The 'runCAKSSimulation' and 'scheduleJobCAKS' functions work together to simulate the CAKS algorithm, aiming to achieve an optimal scheduling strategy that minimizes carbon intensity while efficiently utilizing available resources in a distributed computing environment.

5.4 SolveLPRelaxation function

SRR APX algorithm uses this function and RRAPX algorithm for determining faction of solution to job scheduling problem with the help of LP relaxation. This function is designed to ideally allocate job to the data centres, this will reduce carbon emission while keeping resource constraints on track. Three primary inputs are used in this function, first, an matrix of carbon emission value for each data centres, second, job deadline and finally a matrix of all the available resources for each data centres with time slots (resource availability). The fraction solution matrix is the output of the function which provides optimal allocation of jobs to the data centres across different time slots. The function starts by determining the number of jobs, number of data centres and available and time horizon. 'FranctionalSolution' matrix is then initialized this will store the results of LP relaxation process. Also, 'objectiveFunction' array is set up to define the linear objective function for LP solver. All these steps involves iterating through each jobs, data centres and time slots, and these sets the corresponds value in the 'objectiveFunction' array to carbon emission for that specific data centre and time slots. To make certain feasible scheduling, this function provides set of liner limitations after objective function is created. Just like Breukelman et al. (2024), the first set of constraints are resource constraint, this makes sure that overall resources are used with coefficients that are set to MIPS requirements of the workloads which are created for each data centres and time slot. The second set of constraints are the job scheduling constraints, this guarantees that every job is scheduled precisely once for every data centre and time slot. To perform this, one more constrains is created for each job with coefficient set to one for all possible job allocations. The main aim of this job solver is to minimise the objective function while meting all constraints. The function extracts the solution values into 'fractionalsolution' matrix if the solver finds solution. This matrix will act as an guide for more detailed scheduling algorithm by indicating the percentage of each job that should be assigned to each data centre and time slot

5.5 RunRRAPXSimulation function

Even this function tries to optimize the job scheduling in IfogSim environment. It aims to minimize the carbon intensity and given resource constraints. This function has two inputs, the matrix of carbon intensity and deadline of each job. This function returns the best GRU that we get while executing the function.
'bestGRU' is initialized to zero in the beginning and empty 'schedulingMatrix' is created to keep track of job that scheduled across all the fog devices and deadlines of the

jobs. It uses the 'solveLPRelaxation' function to obtain a fractional solution for the job scheduling problem. This fractional solution gives the path for initial allocation of jobs based on carbon intensity and available resources. The main part of the function is iterating through the list of jobs and schedule each job using the 'scheduleJobRRAPX' function. This function compares the fractional solution with the available resources and carbon intensity to determine whether it is feasible to schedule a job at each data centre and time slot. When a feasible assignment is identified, the 'schedulingMatrix' and 'resourceAvailability' matrices are modified accordingly and the job is scheduled. The total energy consumption of the relevant fog device is increased by the energy used by each job. The function computes the GRU by adding the scheduled resource usage to the carbon intensity for every data centre and time slot once all tasks have been scheduled. This calculation provides a measure of the efficiency of the resource utilization of the carbon intensity. The function then returns the bestGRU value, representing the optimal GRU achieved during the simulation. Based on the fractional solution generated from the LP relaxation, the 'scheduleJobRRAPX' function is an assist function used within the main runRRAPXSimulation function to identify the optimal assignment for each work. It iterates through each data center and time slot to check if the fractional solution indicates a positive allocation for the job. If the data center has enough resources and the job can be completed within the given time slot, then job is assigned to that slot. The function returns the best assignment found and the start time slot. If no feasible assignment is found, the function returns null value.

5.6 RunSRRAPXSimulation function

SRR APX(Stochastic Relaxed Resource Allocation for APX) algorithm is the extension of the RR APX algorithm. By balancing resource availability and carbon intensity, this function tries to optimise job scheduling in a fog computing environment and minimise the overall carbon impact while meeting resource restrictions.
The function begins by initializing the bestGRU to zero and creating empty matrices for the 'bestSchedulingMatrix' and the current schedulingMatrix as it indicates the single simulation iteration. Just like above function, the 'solveLPRelaxation' function is then called to obtain a fractional solution for the job scheduling problem, providing a guideline for the initial allocation of jobs based on carbon intensity and resource availability. The main part of the function involves iterating through the number of iterations S. For each iteration, it initializes a new 'schedulingMatrix' and a counter for 'failedJobs'. It then attempts to schedule each job from the JOBS list using the scheduleJobSRRAPX function. This function checks the feasibility of scheduling a job at each data center and time slot based on the fractional solution, resource availability, and carbon intensity. If a feasible assignment is found, the job is scheduled, and the 'schedulingMatrix' and 'resourceAvailability' matrices are updated accordingly. If the job cannot be scheduled, the 'failedJobs' counter is incremented. The function uses the 'calculateGRU' function to determine the GRU for the current scheduling matrix after trying to schedule every job.Then, the ratio of scheduled resource utilisation to carbon intensity for each data centre and time slot is added up to determine the GRU. Next, the function compares the present GRU with the best GRU discovered so far and then 'bestGRU' and 'bestSchedulingMatrix' are updated if the current GRU performs better Based on the fractional solution obtained from the LP relaxation, the scheduleJobSRRAPX function is an assist function used within the main runSRRAPXSimulation function to identify the optimal assignment for each work.

It iterates through each data center and time slot, checking if the fractional solution indicates a positive allocation for the job. If the the data center has enough resources and the job can be completed within the given time slot then the job is assigned to that slot. The function returns the best possible assignment, which is indicated by an array with the data centre index and the start time slot. The function returns null in the case that no workable assignment is discovered.

6 Evaluation

This section analyzes the performance and efficiency of the proposed Carbon Aware Kubernetes Scheduler (CAKS) algorithm in comparison with the RRAPX and SRRAPX algorithms.The results are obtained from the simulations that are done in Ifogsim framework. simulations. It mainly focuses on metrics such as the Green Resource Utilization (GRU) and energy consumption. By examining these metrics, research aims to demonstrate the effectiveness of the CAKS algorithm in reducing the carbon footprint of data center operations while maintaining efficient resource utilization.
Simulation contains total of five data centres. The parameters such as carbon intensity and resource availability of three of the data centers(Ontario-Canada, Newyork-USA, Greater Britain) were taken from the real time carbon intensity that was recorded on 2023/07/22. Parameters for the other two data centre are given statistically in the implementation. The simulations were run several times with different parameters each time to compare the results of all three algorithms. The results of given parameters are shown below table.

Steps in GRU Calculation:

1. **Job Scheduling**: Jobs are assigned to data centers and time slots, and for each assignment, a binary decision variable x_{ij}^t is set to 1 if job i is scheduled on data center j at time t, otherwise it remains 0.

2. **Carbon Intensity**: For each scheduled job, the carbon intensity at the assigned data center and time slot is considered. Higher carbon intensity increases the environmental cost.

3. **Summation**: The summation is taken over all jobs i, time slots t, and data centers j, dividing the resource usage of each job r_i by the carbon intensity I_j^t.Bahreini et al. (2023)

Table 3: Comparison of Algorithms for Set 1

Metric	**CAKS**	**RRAPX**	**SRRAPX**
Best GRU	162850.80	7.67	268.12
Energy of dc1	2039678.00	2039678.00	2039678.00
Energy of dc2	2328063.27	2328063.27	2328063.27
Energy of dc3	11426740.87	11426740.87	11426740.87
Energy of dc4	200963.27	200963.27	200963.27
Energy of dc5	201218.89	201218.89	201218.89
Energy of cloud	166866.60	166866.60	166866.60
Best GRU Bahreini et al. (2023)	89450.06	8.33	226.54

Table 4: Comparison of Algorithms for Set 2

Metric	CAKS	RRAPX	SRRAPX
Energy of dc1	2338678.00	2338678.00	2338678.00
Energy of dc2	2641063.27	2641063.27	2641063.27
Energy of dc3	10924740.87	10924740.87	10924740.87
Energy of dc4	203463.27	203463.27	203463.27
Energy of dc5	201378.89	201378.89	201378.89
Energy of cloud	166866.60	166866.60	166866.60
Best GRU Bahreini et al. (2023)	51,880.66	8.44	204.27

Table 5: Comparison of Algorithms for Set 3

Metric	CAKS	RRAPX	SRRAPX
Energy of dc1	2530678.00	2530678.00	2530678.00
Energy of dc2	2784063.27	2784063.27	2784063.27
Energy of dc3	11590740.87	11590740.87	11590740.87
Energy of dc4	205963.27	205963.27	205963.27
Energy of dc5	203818.89	203818.89	203818.89
Energy of Cloud	166866.60	166866.60	166866.60
Best GRU Bahreini et al. (2023)	64972.87	8.39	224.16

Table 6: Comparison of Algorithms for Set 4

Metric	CAKS	RRAPX	SRRAPX
Energy of dc1	2902678.00	2902678.00	2902678.00
Energy of dc2	2931063.27	2931063.27	2931063.27
Energy of dc3	12276740.87	12276740.87	12276740.87
Energy of dc4	208623.27	208623.27	208623.27
Energy of dc5	206378.89	206378.89	206378.89
Energy of cloud	166866.60	166866.60	166866.60
Best GRU Bahreini et al. (2023)	51880.66	8.44	204.27

Table 7: Comparison of Algorithms for Set 5

Metric	CAKS	RRAPX	SRRAPX
Energy of dc1	3084678.00	3084678.00	3084678.00
Energy of dc2	3082063.27	3082063.27	3082063.27
Energy of dc3	12982740.87	12982740.87	12982740.87
Energy of dc4	211443.27	211443.27	211443.27
Energy of dc5	209058.89	209058.89	209058.89
Energy of cloud	166866.60	166866.60	166866.60
Best GRU Bahreini et al. (2023)	43673.06	8.48	206.17

In this table:
The execution time is in seconds (s).

The energy consumed by data centers and the cloud is in watt-hours (Wh).
GRU (Green Resource Utilization) is measured in MIPS per gram of CO_2 $\left(\frac{\text{MIPS}}{\text{gram of } CO_2}\right)$.

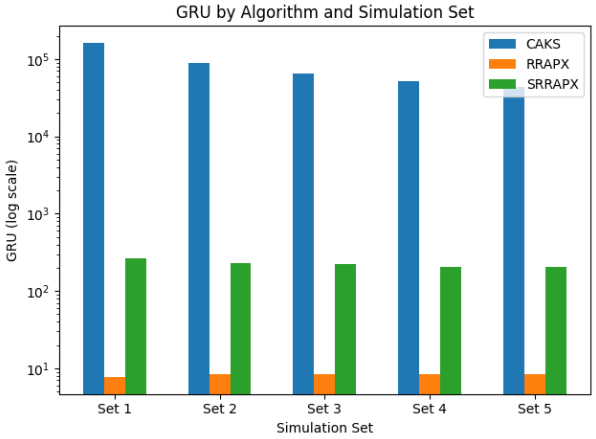

Figure 3: GRU by algorithm simulation set

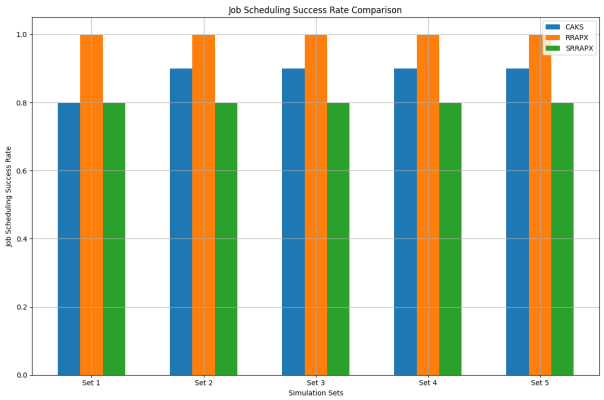

Figure 4: Job scheduling success rate ratio

6.1 Evaluation 1

This evaluation involves the comparison of the Green resource utilization of all three algorithms in all the simulations. The higher GRU indicates a more effective scheduling algorithm in terms of optimizing resource usage and minimizing carbon footprint. The following graph Figure 3provides a visual representation of the GRU comparison across all sets for the three algorithms:

- Set 1: CAKS achieved a GRU of 162850.80, RRAPX achieved 7.67, and SRRAPX achieved 268.12.
- Set 2: CAKS achieved a GRU of 89450.06, RRAPX achieved 8.33, and SRRAPX achieved 226.54.
- Set 3: CAKS achieved a GRU of 64972.87, RRAPX achieved 8.39, and SRRAPX achieved 224.16.
- Set 4: CAKS achieved a GRU of 51880.66, RRAPX achieved 8.44, and SRRAPX achieved 204.27.
- Set 5: CAKS achieved a GRU of 43673.06, RRAPX achieved 8.48, and SRRAPX achieved 206.17.

From above results, it is clearly seen that the CAKS algorithm consistently produces a significantly higher GRU compared to RRAPX and SRRAPX. This proves that CAKS is more efficient in utilizing available resources. The main difference in GRU values between CAKS and the other two algorithms underscores its superiority in optimizing for both resource efficiency and sustainability.

6.2 Evaluation 2

The second metric for measuring the performance of the algorithms is the Job Scheduling Success Ratio. This metric measures the proportion of jobs successfully scheduled and executed within their specified deadlines and resource constraints. The higher job scheduling success ratio tells that algorithm is more reliable to meet the deadline of the jobs and it uses available resources effectively. Figure 4 shows the jobs scheduled by all the algorithms in different sets of simulations. The job scheduling success ratios for each algorithm across the five sets are as follows:

- Set 1: CAKS successfully scheduled all 10 jobs, RRAPX successfully scheduled all 10 jobs, and SRRAPX successfully scheduled 8 out of 10 jobs.
- Set 2: CAKS successfully scheduled all 10 jobs, RRAPX successfully scheduled all 10 jobs, and SRRAPX successfully scheduled 8 out of 10 jobs.
- Set 3: CAKS successfully scheduled all 10 jobs, RRAPX successfully scheduled all 10 jobs, and SRRAPX successfully scheduled 8 out of 10 jobs.
- Set 4: CAKS successfully scheduled all 10 jobs, RRAPX successfully scheduled all 10 jobs, and SRRAPX successfully scheduled 8 out of 10 jobs.
- Set 5: CAKS successfully scheduled all 10 jobs, RRAPX successfully scheduled all 10 jobs, and SRRAPX successfully scheduled 8 out of 10 jobs.

From these results, it is clear that CAKS has assigned all the jobs in four out of five assignments which shows the efficiency if the algorithms. This indicates that these algorithms are highly effective in managing the available resources and meeting job deadlines. Although, RRAPX algorithm was able to schedule 100% of the jobs, SRRAPX failing to schedule 2 out of 10 jobs in each set.

6.3 Discussion

In this study, five sets of simulations were conducted, evaluating the performance of three different algorithms: CAKS, RRAPX, and SRRAPX. Across all simulation sets, the CAKS algorithm consistently demonstrated superior performance by achieving the highest GRU (Green Resource Usage) values. This highlights CAKS's ability to optimize resource utilization more effectively, making it highly efficient in leveraging available computational resources while minimizing environmental impact. By incorporating real-time carbon intensity data into the scheduling process, CAKS dynamically adjusts its decisions to strike a balance between resource optimization and carbon footprint reduction. In contrast, while RRAPX and SRRAPX offer less computational complexity and lower processing overhead, they were unable to match CAKS in terms of resource utilization and efficiency. The results underscore the importance of incorporating environmental factors into workload scheduling, as CAKS's carbon-aware approach proved more adept at handling both energy consumption and resource distribution in the simulated fog environment.

7 Conclusion and Future Work

This simulated an algorithm on IfogSim platform and try to reduce the carbon emission by utilizing more renewable energy from the sources. Main focus of this research was on how can a green scheduler can be designed to effectively optimize workload distribution across multiple clusters based on the availability of low carbon energy sources. This research states that the CAKS algorithm outperformed the other two algorithms across all key metrics. It achieved significantly higher GRU values, suggesting better resource utilization and a more efficient scheduling process. Additionally, CAKS demonstrated a higher job scheduling success rate and a more balanced energy consumption across devices, highlighting its capacity to distribute workloads effectively while considering both resource availability and carbon intensity. The findings in this research is significant development towards green computing. By integrating carbon intensity data into scheduling decisions, CAKS not only optimizes resource usage but also contributes to reducing the overall carbon footprint of computing operations. This approach aligns with the growing emphasis on environmental sustainability in technology, making CAKS a promising solution for future fog and cloud computing environments. These results carry significant implications, especially when considering sustainable computing. Using carbon intensity data into scheduling decisions allows CAKS to minimize total carbon footprint of computing processes while simultaneously optimizing resource utilization. Because of its potential uses in cloud and fog computing, CAKS is a promising solution that fits with the increasing focus on environmental sustainability in technology.
While the study successfully demonstrated the benefits of CAKS, this research still has some limitations. As shown in above results, this research completely drain out the renewable resources out the data centres. By using the heterogeneous approach of using both renewable and non- renewable energy, the energy sources can be balanced out. Future works includes the implementation of this algorithm in actual data centres and would the compare the actual results with simulated results. It also proposes the upgrade an algorithm to use the balanced renewable and non renewable energy and still maintain the carbon intensity level in the atmosphere. In conclusion, there is still a great deal of potential for improvement even though this research has made great progress in assessing and optimizing job scheduling methods for fog computing environments. Further studies can expand on these results to further the subject of sustainable computing by addressing the drawbacks and looking into novel directions for development.

References

Agrawal, S. and Singh, D. (2023). An investigation of the security and privacy implications of fog computing systems using ifogsim, *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pp. 1–4.

Akoush, S., Sohan, R., Rice, A., Moore, A. W. and Hopper, A. (2011). Free lunch: Exploiting renewable energy for computing, *13th Workshop on Hot Topics in Operating Systems (HotOS XIII)*, USENIX Association, Napa, CA.
URL: *https://www.usenix.org/conference/hotosxiii/free-lunch-exploiting-renewable-energy-computing*

Ascensão, P., Neto, L. F., Velasquez, K. and Abreu, D. P. (2024). Assessing kubernetes

distributions: A comparative study, *2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)*, pp. 832–837.

Bahreini, T. and Tantawi, A. (2023). A carbon-aware workload dispatcher in multi-cluster kubernetes environments, *Cloud Native Sustainability Week 2023*. Preprint.
URL: *https://research.ibm.com/publications/a-carbon-aware-workload-dispatcher-in-multi-cluster-kubernetes-environments*

Bahreini, T., Tantawi, A. and Youssef, A. (2022). An approximation algorithm for minimizing the cloud carbon footprint through workload scheduling, *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pp. 522–531.

Bahreini, T., Tantawi, A. and Youssef, A. (2023). A carbon-aware workload dispatcher in cloud computing systems, *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, pp. 212–218.

Breukelman, E., Hall, S., Belgioioso, G. and Dörfler, F. (2024). Carbon-aware computing in a network of data centers: A hierarchical game-theoretic approach, *2024 European Control Conference (ECC)*, pp. 798–803.

Chen, C., He, B. and Tang, X. (2012). Green-aware workload scheduling in geographically distributed data centers, *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pp. 82–89.

Chen, J.-H. and Lu, S.-L. (2020). Gwma algorithm for host overloading detection in cloud computing environment, *in* K.-M. Chao, L. Jiang, O. K. Hussain, S.-P. Ma and X. Fei (eds), *Advances in E-Business Engineering for Ubiquitous Computing*, Springer International Publishing, Cham, pp. 358–370.

Cheng, X., Fu, E., Ling, C. and Lv, L. (2023). Research on kubernetes scheduler optimization based on real load, *2023 3rd International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pp. 711–716.

Deng, X., Wu, D., Shen, J. and He, J. (2016). Eco-aware online power management and load scheduling for green cloud datacenters, *IEEE Systems Journal* **10**(1): 78–87.

Mahmood, H. (2023). The determinants of carbon intensities of different sources of carbon emissions in saudi arabia: The asymmetric role of natural resource rent, *Economies* **11**(11).
URL: *https://www.mdpi.com/2227-7099/11/11/276*

Poulton, N. (2024). *Quick Start Kubernetes - Second Edition*, Packt Publishing. Available at: `https://research.ebsco.com/linkprocessor/plink?id=ec9fbcce-3d63-3e22-9728-7d5ce50dbdc2` (Accessed: 11 August 2024).

Sarkar, S., Naug, A., Luna, R., Guillen, A., Gundecha, V., Ghorbanpour, S., Mousavi, S., Markovikj, D. and Ramesh Babu, A. (2024). Carbon footprint reduction for sustainable data centers in real-time, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, pp. 22322–22330.

WU, J., Chao, L., CUI, W. and LIU, W. (2020). The resource scheduling algorithm of energy consumption optimization, *2020 IEEE 2nd International Conference on Power Data Science (ICPDS)*, pp. 32–35.

Yousuf Khan, E. U. and Rahim Soomro, T. (2022). Overview of ifogsim: A tool for simulating fog networks of the future, *2022 14th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, pp. 1–4.