International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 6288-6305
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Carbon Conscious Scheduling in Kubernetes to Cut Energy Use and Emissions

Nishanth Reddy Pinnapareddy*

Senior Software Enginner, Doordash Inc, USA.
* **Corresponding Author Email:** nishanth.pinnapareddy@gmail.com - **ORCID:** 0000-0002-5247-7144

Article Info:

DOI: 10.22399/ijcesen.3785
Received : 18 June 2025
Accepted : 15 August 2025

Keywords

Carbon-aware scheduling,
Kubernetes,
Carbon intensity,
Cloud sustainability,
Green computing

Abstract:

With the growing energy consumption of hyperscale data centers, cloud computing is also growing and the proliferation of cloud computing is increasing its greenhouse gas (GHG) emissions. Kubernetes, the strongly recommended container orchestration platform through the multi-cloud environments, heavily impacts the deployment of compute resources. Traditional scheduling focuses on throughput, latency, and resource usage and does not pay attention to the carbon intensity (gCO 2 /kWh) of the consumed electricity. This paper postulates the use of carbon-sensitive scheduling throughout Kubernetes to dynamically schedule operations according with low-carbon energy production times. By using realizable, predictive carbon intensity values available through APIs like ElectricityMap or WattTime, workloads can be either geographically scheduled to be run in regions with higher renewables penetration, or scheduled so that it runs during periods when the grid is predicted to be cleaner. Placement makes use of native Kubernetes mechanisms node affinity, taints and tolerations, and custom scheduling policy to optimise placement. This paper describes an architectural framework, system assessment methods based on carbon intensity trace data, and deployment case studies of innovative technologies at major technology companies and research laboratories. The results show that carbon-aware workload placement can save at least 10 and up to 30 percent of CO 2 emissions and do so without any perceptible performance impact. There are still difficulties of granularity of carbon data, interoperability standards, and adoption of the enterprise.

1. Introduction

In today's technological world, technology is deeply ingrained in every aspect of life. Whether watching movies online or running smart home devices, conducting business, or storing data, these functions depend significantly on data centers — huge facilities housing thousands of servers, networking gear, and continuous cooling infrastructure. These centers are massive electric guzzlers, much of that still coming from fossil fuel sources. The energy footprint of the data center is increasing with the demand for more internet services and cloud computing. Data centers suck down about 1–2% of electricity around the world now, and this number is expected to explode. That might not seem like much, but it's enough energy to power millions of homes. This increase in power consumption translates into increased greenhouse gas emissions when the energy comes from coal and natural gas, which are carbon-intensive. While this warms our planet, increasing its temperature and exacerbating the climate crisis, these emissions trap heat in the atmosphere. Consequently, the responsibility of the technology sector to reduce environmental harm is becoming apparent. Kubernetes is one of the main modern cloud infrastructure tools. Kubernetes is widely used across industries to scale and manage containerized applications. It allows the software to run over a group of machines in a network, called nodes, and deal with operations like load balancing and fault recovery. While this can be advantageous to uptime and speed, it can unwittingly increase carbon emissions when workloads are assigned to nodes running on dirty energy. By contrast, if Kubernetes can be tuned to prioritize nodes with cleaner electricity, Kubernetes can potentially reduce emissions. This is where this notion of carbon-aware computing is important.

Carbon-aware computing is a smarter, more energy-aware way to build information systems. Instead of merely organizing actions according to availability or performance, it considers the environmental impact of electricity at that particular location and

moment. Systems can determine when and where power is cleaner by analyzing its "carbon intensity"—its capacity to emit carbon dioxide per kilowatt hour of electricity generated. For example, a system might stave off no imperative rationing or course errands to a zone where solar or wind power is now accessible. By making this type of dynamic decision, this kind of decision-making simultaneously lowers emissions. It helps keep costs lower on the energy side since renewable power is typically less expensive during peak power generation times. This article considers how Kubernetes can be instrumental in supporting carbon-aware scheduling practices. It examines how these technical foundations can be integrated with real-time carbon data sources and the tools necessary to carry them out. It also gives real-world case studies of organizations currently using these techniques and provides insight into what worked, what didn't, and what lessons they learned. The aim is to provide developers, IT administrators, and cloud architects with real, usable advice for constructing greener infrastructure. Along the way, readers will learn about the critical relationship between digital systems and the environment. They will see how seemingly small improvements in workload management can collectively amount to very big steps in fighting climate change.

1.1. Background

Traditional Kubernetes Scheduling Explained

One of the popular platforms for managing containerized applications is Kubernetes, which has become a prominent platform in a very short time (Pamadi et al., 2024). In brief, a container is just a lightweight package of software that has everything to run an app. With the presence of Kubernetes, these containers can be deployed, monitored, and managed across a group of servers (known as nodes). The Kubernetes scheduler must be a key component. A few things factor into where it should run each container: how much CPU or memory is available, how busy each server is, and any rules the developer or system administrator has set. However, conventional Kubernetes lacks sensitivity to carbon-aware sourcing or emissions data, even though dual sourcing strategies in systems design increasingly emphasize sustainability and resilience (Goel & Bhramhabhatt, 2024). This is not about carbon emissions from decisions about resource use and performance. Because of this, Kubernetes may assign workloads to servers powered by coal-heavy electricity, even if they can be pulled from somewhere else with cleaner electricity. Of course, this method is fine for performance and availability, but it misses an opportunity to reduce environmental harm (Karkkainen, 2019).

The image below illustrates how the traditional Kubernetes Container Scheduling System (KCSS) processes a queue of containerized workloads without consideration of the carbon intensity of the underlying infrastructure. Kubernetes places tasks onto nodes based solely on factors like CPU, memory availability, and resource policies, without regard for how "clean" or "dirty" the electricity powering those nodes is.

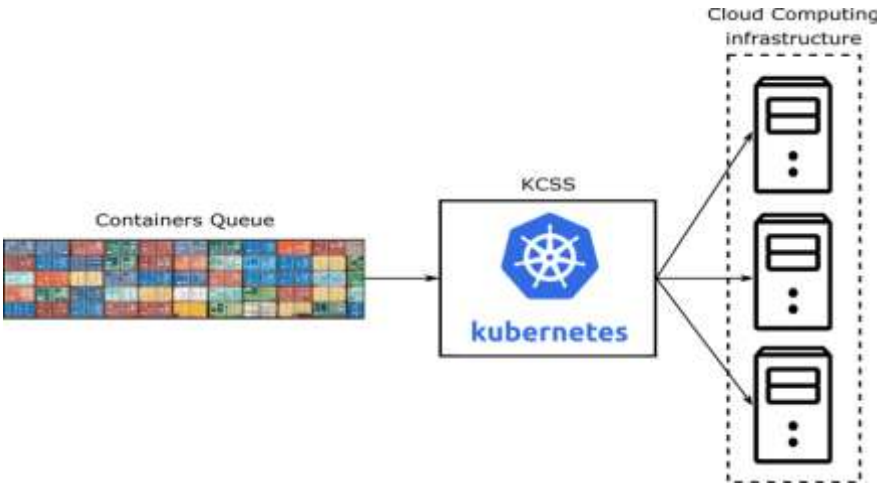

Figure 1: *Kubernetes-container-scheduling-strategy*

Cloud Infrastructure Energy Consumption

Modern cloud services operate in big data centers worldwide. These facilities are filled with servers that run 24 hours a day and consume massive amounts of power. Energy is used to power the servers, cool them, backup systems, and light the place. According to a 2020 study from the International Energy Agency (IEA), data centers consume about 200 terawatt hours of electricity around the world. That's nearly 1% of all electricity demand, more than the total electricity consumption of some small countries. Internet usage is globally on the rise, and companies are shifting to cloud platforms, so this demand is also increasing. Big tech

companies, Google, Amazon, and Microsoft, for example, run some of the biggest data centers (Jones, 2018). While many try to use renewable energy, the type of power used depends on location and time. Sometimes, these data centers draw their electricity from electricity produced by burning coal or gas, thereby contributing to higher emissions.

Understanding Carbon Intensity in the Grid

It's not always clear where electricity comes from. Clean energy, such as solar, wind, and hydro, can generate it, or fossil fuels such as coal and natural gas can. Electricity's carbon intensity refers to how much carbon dioxide emissions per unit of power (typically measured in grams per kilowatt-hour) are produced in the production of the power (Dhanagari, 2024). For example, in a given region, low carbon intensity may occur when a region gets most of its electricity from solar energy during the day. On the other hand, increased reliance on fossil fuel plants at night raises carbon intensity. Put another way, performing the same task at different times and locations can result in significantly different environmental impacts. Carbon-aware scheduling comes into play here. Systems like Kubernetes could run tasks in low-carb regions or wait until clean power is available by checking the carbon intensity of the electricity in real time (Dhanagari, 2024).

Table 1: *Comparison of Carbon Intensity by Energy Source*

Energy Source	Average Carbon Intensity (g CO_2/kWh)	Emission Level
Coal	820	Very High
Natural Gas	490	High
Oil	650	High
Biomass	230	Moderate
Solar PV	40	Very Low
Wind	12	Very Low
Hydroelectric	24	Very Low
Nuclear	16	Very Low

Data Centers and Tech Industry Sustainability

As the tech industry comes to terms with its role in tackling climate change and the pressure to account for its carbon footprint, major cloud providers are making significant environmental goals public (Turek et al., 2021). For example, Google Cloud wants to run on 100 percent carbon-free energy around the clock by 2030. Microsoft echoed the pledge, saying it aims to become carbon-negative by the same year. Meanwhile, Amazon Web Services (AWS) has committed to sourcing all its energy from renewable sources by 2025. While these targets are bold and commendable, achieving them will involve more than purchasing renewable energy. It requires new tools and smarter means of managing energy use across the increasingly complicated computing ecosystems.

Adopting carbon-aware computing strategies is one possible way to support these sustainability efforts. Making Kubernetes—the most popular container orchestration platform globally—carbon aware can help lower emissions. Carbon-aware scheduling allows Kubernetes to schedule a workload on infrastructure with access to inexpensive and cleaner electricity sources, and also schedules non-urgent tasks when carbon prices may be lower. While technically it may appear to be a simple shift in scheduling logic, embedding this into production environments requires integrating security and automation into the broader DevOps lifecycle, as seen in DevSecOps pipelines (Konneru, 2021). Organizations can meet sustainability commitments closely without performance tradeoffs by aligning compute operations to real-time energy availability.

1.2. Importance of Carbon-Aware Scheduling

Aligning Computing with Clean Energy Availability

The idea behind carbon-aware scheduling is that computer tasks are executed when and where electricity is less carbon-intensive (Buchanan et al., 2023). Data centers do not all use the same electricity, as the source varies throughout the day and across regions. Sometimes, clean sources such as wind or solar generate electricity, while at other times, it is produced from fossil fuels like coal or gas, which emit more carbon. By using real-time data about the carbon intensity of the electricity grid, tasks can be scheduled more efficiently. For example, if wind energy is cranking during the night in one region, Kubernetes could move some of this work there instead of to a region still powering up with coal. For a flexible task, like generating reports or running backups, a utility can delay the task by a few hours until cleaner electricity becomes available. The only downside is that this type of scheduling requires no significant changes in how

the cloud is used. While it can reduce carbon output, nothing is more climate-friendly than choosing the right time or place to run a task. As shown in the image below, a Green Data Center isn't just about hardware efficiency—it involves energy-efficient resources, optimization, regulatory compliance, and smart resource scheduling.

Figure 2: *Fundamentals of green data centers*

Reducing Performance without Losing Emissions

One of the nicest things about carbon-aware scheduling is that speed or quality does not have to be sacrificed (Guyon, 2018). This type of computing task — batch processing, file backup, data analysis, video encoding, or training machine learning models — isn't something that needs to run now. These workloads tend to be flexible; a delay of a couple of minutes or even a few hours causes little to no impact on users and business operations. Rather than executing these tasks when they are queued, Kubernetes can be configured to execute them when cleaner energy is available or execute them from a different data center in a region where energy is produced less carbon at that time. The strategy dramatically reduces carbon emissions with no adverse effect on performance. Additionally, in several instances, it can result in financial benefits. When they are abundant, solar and wind can be cheap, clean energy sources. As emphasized in scalable systems design, aligning resource utilization with efficient timing can enhance both operational value and sustainability (Sardana, 2022). Companies can lower their environmental footprint and decrease operating costs by aligning workload execution with these low-cost, low-carbon periods. As a result, carbon-aware scheduling provides a smart middle path that is good for the planet and the bottom line.

Table 2: *Examples of Flexible vs Non-Flexible Workloads*

Workload Type	Flexibility	Suitable for Carbon-Aware Scheduling?
Video Encoding	High	Yes
Batch Reporting	High	Yes
CI/CD Pipeline Builds	Medium	Yes (with delay tolerance)
Real-time Chat	Low	No
Fraud Detection Alerts	Low	No
Data Backups	High	Yes

Energy Efficiency Benefits and Financial Incentives

Businesses that don't go green are losing out on more than just the planet (Smith & World Economics Association, 2016). As expected, more and more companies are being asked to report their carbon emissions as part of their Environmental, Social, and Governance (ESG) goals. Carbon-aware scheduling gives businesses a real way to reduce their carbon impact from their cloud and show they're doing something about climate. Some cloud providers already charge different rates for different locations or times of high demand. It's more costly to run a

task when high energy demand is traditionally driven by higher carbon intensity. But scheduling jobs during low-demand, clean-energy windows can save companies money. This strategy mirrors principles seen in modern architecture, where context-sensitive workload placement supports cost and resource control (Chavan, 2022). Energy efficiency is another benefit. Out of the box, traditional data center operations will waste resources. For example, servers could be underused, or the workloads could be spread imperfectly. As microservices increasingly aim to scale without overspending, carbon-aware scheduling offers an approach that balances operational efficiency with environmental stewardship, promoting better utilization in cleaner, lower-cost regions (Chavan, 2023).

Widening Impact: Edge to Cloud

Carbon-aware scheduling is not just for big data centers. As computing reaches the 'edge'—smaller servers closer to users, such as in stores or offices—these ideas have the potential for much wider application. Consider, for example, a retail company operating in multiple cities: it could conduct updates or system checks at predefined time intervals dependent on the carbon intensity of the power grid of the given region (Wang et al., 2021). With smart devices and edge servers popping up in homes, factories, and offices, carbon-aware techniques allow tasks to be coordinated to benefit both the user and the environment.

1.3. Key Concepts and Terminologies

Understanding carbon-aware Kubernetes scheduling begins with a grasp of a few basic concepts. These terms and ideas have become the basis of how computing and environmental impact is intimately tied. This section explains the concept of carbon intensity, the functioning of real-time grid data, the operation of a carbon-aware scheduler, and how workload shifting fits in with the rest to reduce emissions through smarter scheduling. The use cases around these concepts aren't theoretical — they're being used right now to make cloud computing more sustainable (Hurwitz & Kirsch, 2020).

The image below illustrates the internal flow of the Kubernetes Pod Scheduling Context, from picking a pod from the scheduling queue to binding it to a node

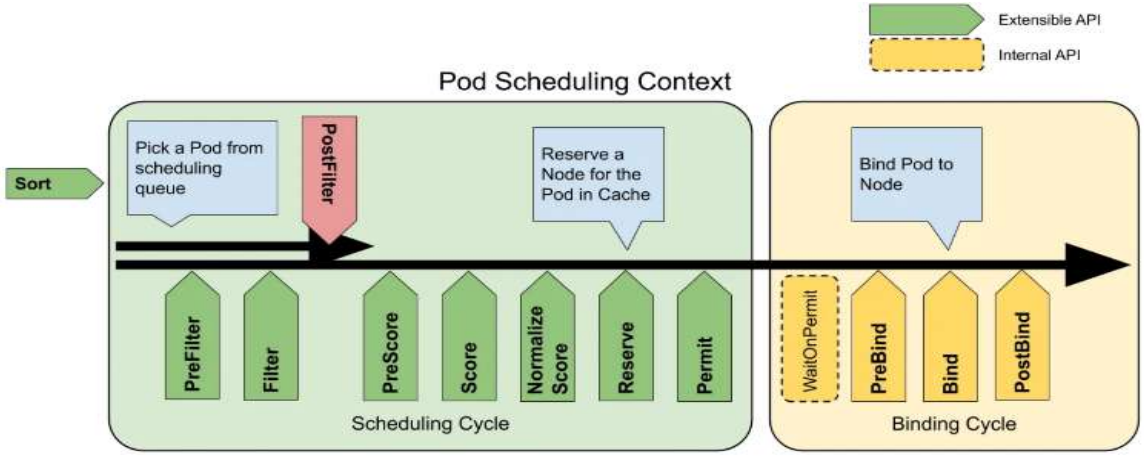

***Figure 3**: kubernetes-scheduling-*

The Carbon Intensity

Carbon intensity measures how much carbon dioxide is put in the atmosphere for each unit of electricity produced. It's generally given in grams of carbon dioxide per kilowatt hour (g CO_2/kWh). This number varies with the source of electricity production. High carbon intensity involves, for example, electricity generated by burning coal or gas, which creates a great amount of carbon. Electricity made by wind turbines, solar panels, or hydroelectric dams creates very low or no carbon emissions. Understanding these variations in energy sources is critical, especially as data systems increasingly rely on inference-driven automation, which demands real-time, efficient power allocation (Raju, 2017). Carbon intensity is always changing. The most important thing to know about carbon intensity is that it isn't even the same everywhere, and doesn't stay the same daily. At some points, certain parts may be very dependent upon coal during peak demand hours, while at other times, others might create excess solar energy through sunny afternoons. This enables them to schedule tasks when electricity is cleaner, thus reducing the overall carbon footprint of digital operations by tracking such changes in carbon intensity with computing systems.

Real-Time Grid Emissions Data

This is to support action based on carbon intensity. To do so, systems need access to real-time data about the electricity grid (Chen et al., 2023). This data provides information on where electricity comes from and how much carbon is being released to generate it. This kind of data was hard to obtain in

the past, but now, with many platforms, it is available as a service. These services compute the current carbon intensity at different locations by gathering data from grid operators, power plants, and energy markets. Thanks to their real-time emissions data, systems like Kubernetes can make smarter decisions. If the data fairly proves that a grid in one region is comprised mostly of clean energy that location can be chosen for running heavy computing workloads. Meanwhile, the system could postpone or reroute jobs if a region currently burns coal or gas to generate power. Access to this kind of data makes scheduling in this traditional way more responsible and environmentally aware.

Carbon Aware Schedulers

A carbon-aware scheduler is a system that decides when and where to run computing tasks based on information about the carbon intensity of the resources it has access to. In Kubernetes, scheduling prioritizes performance, availability, and resource use. It simply doesn't care if electricity is clean or dirty. Along with this intelligence comes the ability to include emissions data in the decision. As predictive analytics continues to influence infrastructure-level optimization, carbon-aware scheduling naturally fits into intelligent orchestration pipelines that enhance DevOps and sustainability (Kumar, 2019). However, there are many ways to integrate carbon-aware scheduling into Kubernetes. Sometimes, developers create custom schedulers that talk to the emissions data sources themselves. Others employ external tools that monitor the grid and augment Kubernetes node labels with carbon intensity levels. Next, workloads can be routed to 'cleaner' nodes using standard Kubernetes rules such as node affinity or taints; the idea remains the same each time: match workloads with the most environment-friendly computing options. The great aspect of carbon-aware scheduling is that it happens silently in the background. There's no need for users to modify their applications or workflows. Whenever it makes environmental sense, the system sits idle or chooses a better location, yet it does not sacrifice performance or reliability (Pedelty, 2016).

Workload Shifting in Practice.

Shifting workload is one of the most successful tactics for carbon-aware computing. Simply put, it involves shifting workloads, either in time or in space, to lower their carbon impact. Google Science Fair uses the words "time shifting" and "location shifting" to describe these concepts. This is a useful approach, especially for non-urgent tasks. Video rendering, backup processes, software updates, and training machine learning models are a few workloads that can be flexibly scheduled. They can be safely delayed by a few minutes or even hours so that they do not have to be completed immediately, and without affecting end users. The flexibility associated with these jobs makes them ideal for carbon-aware schedulers to 'bend' their jobs to run at a time that benefits from clean energy availability.

For example, in practice, a software development team can schedule their nightly testing jobs sometime in the early morning hours when wind energy is more available (Wizelius, 2015). Or it might mean relocating database backups to a data center in a region currently being powered by hydropower. These changes are often invisible to users but could result in substantial reductions in emissions over time.

Kubernetes Supporting Concepts

While Kubernetes doesn't natively handle carbon data, it does have a few built-in features that will assist with carbon-aware scheduling. Big tools give each node labels that describe its characteristics. For example, a node can be marked as a "low-carbon" node if it happens to be located in a clean energy region.

As of Kubernetes 1.6, affinity and anti-affinity rules allow us to configure workloads to prefer or avoid running on particular nodes depending on labels (Ungureanu et al., 2019). Taints and tolerations take another step by marking nodes to run only certain workloads. This allows organizations to hone in on exactly where their workloads go using up-to-date emissions data and these features.

Additionally, CI/CD pipelines, which automate software builds and tests, can become carbon-aware. The pipeline can wait until a cleaner energy window arrives before the jobs run. Easy to automate through scripts and APIs, this is a simple adjustment that can help us put carbon-aware practices into action very easily

Table 3: *Kubernetes Features Supporting Carbon-Aware Scheduling*

Feature	**Purpose**	**How It Helps Carbon-Aware Scheduling**
Node Labels	Tag nodes with metadata	Label nodes with "low", "medium", "high" carbon intensity
Node Affinity	Prefer/require workloads on specific nodes	Direct workloads to cleaner energy nodes

Feature	Purpose	How It Helps Carbon-Aware Scheduling
Taints and Tolerations	Restrict workloads from running on nodes	Reserve clean nodes for critical workloads
Custom Scheduler	Customize scheduling logic	Integrate carbon data into scheduling

1.4 Existing Approaches and Tools

Microsoft's Carbon-Aware SDK

Microsoft created one of the earliest and most practical carbon-aware computing tools (Ahmed et al., 2015). To help developers and ops teams reduce emissions by making smart decisions about where and when they run workloads, the company released an open-source Carbon-Aware SDK. The SDK can work with local machines, cloud environments. It will certainly be useful, as it can be integrated into existing DevOps workflows, CI/CD pipelines, and scheduling platforms. This SDK allows organizations to schedule computing tasks based on real-time or forecasted carbon intensity data. For instance, if a data center in Oregon runs on clean hydropower in the early morning hours, the SDK can manage the workloads flexibly enough to move them to that time window. The SDK supports different strategies — from postponing jobs until cleaner energy is available to redirecting jobs to regions with less carbon-intensive grids. As seen in other data-intensive industries like fleet telematics, where intelligent tools improve timing and efficiency, this SDK lowers barriers for software teams aiming to act on climate goals without heavy infrastructure changes (Nyati, 2018).

Electricity Map provides real-time emissions data.

Creating any carbon-aware system requires data, and accurate data at that. The leading platform providing this sort of information is Electricity Map. This service tracks and displays in real time the carbon intensity of electricity grids around the world. By collecting information from national grid operators, power plants, and weather forecasts, and processing it, it shows how clean or dirty the power is in each region. This data is available for developers and researchers to access via Electricity Map's API. Then, slapping this information on Kubernetes allows nodes to be tagged by region and carbon intensity to place workloads more intelligently. For example, Kubernetes might preferentially locate pods onto nodes in regions where electricity is produced by wind, solar, or hydroelectric power sources. Energy-conscious organizations already use Electricity Map to view their carbon footprint and automatically schedule decisions based on evolving energy patterns (de Vasconcelos Danen, 2018).

Forecasting with WattTime

WattTime is another important player in the carbon data space. Compared to ElectricityMap, a broad, real-time snapshot-focused platform, WattTime is more forecast-oriented and less real-time. The software also predicts how carbon intensity will and will not change shortly. This forecast data is particularly useful to systems that need to schedule tasks even several hours ahead or overnight. WattTime uses machine learning models trained on past grid behavior, weather forecasts, and plant operation data. This produces an accurate picture of when cleaner power will most likely be available. Kubernetes can use this data to schedule batch processing or data transfer jobs to avoid high carbon time as much as possible. At the same time, WattTime has partnered with environmental and policy groups to bring their data into public and private sector projects.

Custom Kubernetes Schedulers and Plugins.

While carbon-aware scheduling is not built into plain Kubernetes recipes, its flexible architecture allows custom schedulers and plugins to be written that deliver this functionality. Some organizations have started building their own pluggable into the Kubernetes control plane and are ready. Typically, these systems do this by monitoring the emissions data from a trusted source and relabeling each node with the most up-to-date carbon intensity scores. After nodes are labeled, Kubernetes' built-in scheduling policies—such as affinity rules or node selectors—can be used to control the placement of workloads onto 'cleaner' nodes. Sometimes, companies block some jobs from accessing high-carbon nodes unless a better alternative exists, with taints and tolerations. That way, Kubernetes can focus on sustainability while still having system performance and reliability. One method often used in practice is to have a separate monitoring system running outside Kubernetes that periodically updates the labels or scheduling rules. This implies that Kubernetes itself may not be aware of the carbon context, but it can still be influenced by updated metadata and indirectly become carbon-aware.

The image below demonstrates a practical example of custom scheduling in Kubernetes based on GPU availability. A user submits a POD request requiring GPU resources to the local Kubernetes cluster

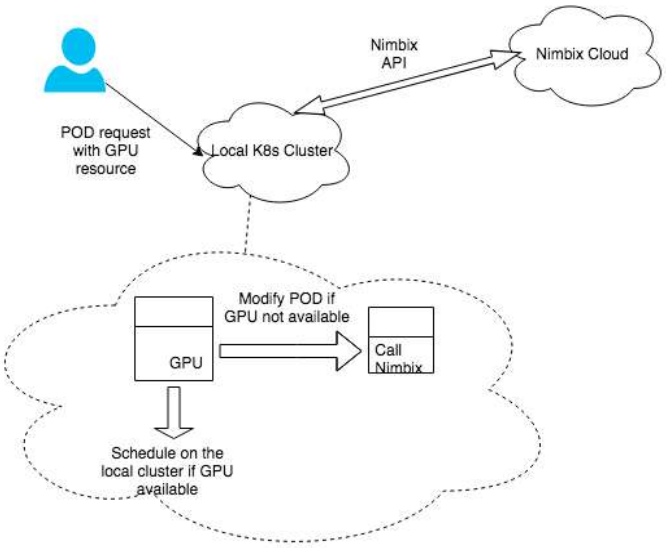

Figure 4: *Experimenting with custom Kubernetes scheduler*

Real-World Use Cases and Developer's Adoption

Companies are already using these tools for good. For example, Microsoft's Carbon Aware SDK (open-sourced in January 2020) is used by some businesses whose internal job schedulers are connected to that SDK, using live emissions data to work out when they run compute-intensive operations. Industry-specific, like video processing, can experience delayed rendering with minimal negative effects on end users. ElectricityMap data is also being used by research labs and universities to schedule when to schedule simulations or to back up when their grid is cleaner in their region. Developer teams are adjusting their CI/CD pipelines to be more carbon-friendly. Rather than having code changes immediately trigger build software, a few teams have even gotten away with delaying builds for an hour or two — just long enough to catch the lower emissions window. These small changes can yield significant emission reductions when applied on a large scale by a large team or over a long period.

Table 4: *Real-World Tools and Data Providers*

Tool/Service	Provider	Purpose	Integration Example
Carbon Aware SDK	Microsoft	Schedule based on emissions data	Used in internal CI/CD pipelines
ElectricityMap API	Tomorrow.io	Real-time global grid emissions data	Tagging nodes in Kubernetes
WattTime Forecast	WattTime	Predictive emissions forecasting	Scheduling jobs ahead for low-carbon windows
Kepler	Open Source	Power-level metrics in Kubernetes clusters	Augments scheduling with energy efficiency

1.5 Carbon-Aware Kubernetes Scheduling Architecture

Building on the Kubernetes Foundation

It's a powerful and flexible system for managing containerized applications (Khan, 2017). At its core, Kubernetes worries about where and how software containers are run on a cluster of machines called nodes. When it decides where to place each container, the built-in scheduler looks at available resources such as CPU, memory, and disk space. Kubernetes makes these decisions without explicitly considering environmental factors like carbon emissions or energy sources by default. Replacing the system is not necessary to make it carbon-aware; enhancements can be built on top of the existing architecture. As with advanced AI systems that integrate multiple data types for better contextual inference, Kubernetes includes extension points that allow developers to inject smarter scheduling logic (Singh, 2022). These consist of node labels, taints and tolerations, affinity rules, and custom and external controllers. By superimposing the benefits of these features on live carbon intensity data, a system can be designed that automatically favors nodes in low-emission regions and times, much like fusing diverse inputs to generate optimized 3D outputs in synthetic environments (Singh, 2022).

Integrating Carbon Intensity.

Real-time carbon intensity data is fundamental to achieving carbon-aware scheduling. This data can be obtained from external services such as ElectricityMap or WattTime, which monitor how clean the electricity in regions is at a given time. Their APIs return current carbon intensity values or forecasts based on the condition of the power grid. With this data, a background service could be created to inspect the carbon intensity of the regions where the cluster's nodes are currently running and use that information as part of Kubernetes admission control to guide placement decisions (Abdelmassih, 2018). Then, the nodes can be labeled by this service. For instance, a node in a solar region may be labeled as a 'low carbon genius node,' and a node in a coal-heavy region may be labeled as a 'high carbon genius node.' These are environmental tags that Kubernetes can use when scheduling.

The popular DMZ usage is for this process, which typically involves a lightweight controller or agent that runs outside of the core Kubernetes components. The node labels instead poll the emissions data regularly, after all five or ten minutes, for illustration, and update them based on the most up-to-date information. The labels automatically change if carbon levels change throughout the day, so long as the system is up to date.

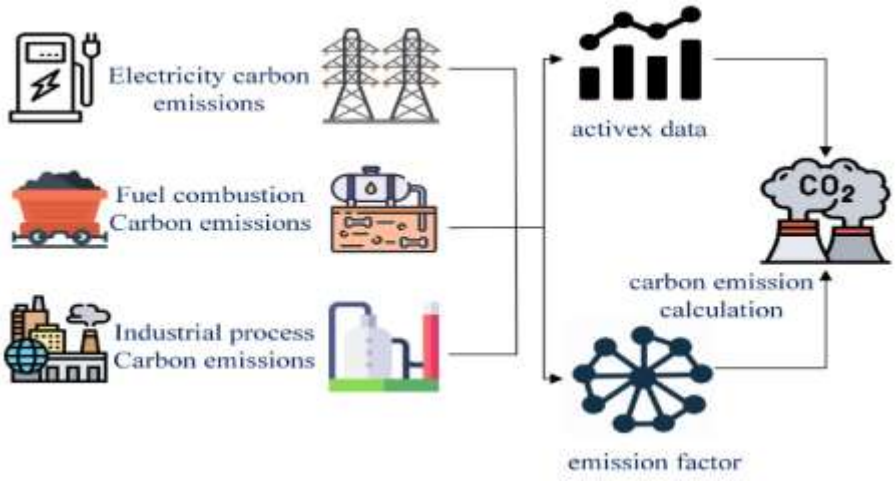

Figure 5*: Schematic diagram of the dynamic accounting framework for carbon emissions in high-energy-consuming industrial parks.*

Kubernetes Scheduling Features

After nodes are labeled with carbon intensity, Kubernetes' native scheduling tools can be leveraged to control workload placement. The most common method is pod affinities, where a pod (a group of containers) can be directed to run on a specific node labeled accordingly. For example, to use Node Affinity to prefer nodes labeled as 'low carbon', a workload can be configured with this preference. Kubernetes places the pod in such a node if present. However, if that isn't the case, it may either fall back to a less ideal alternative or wait till one becomes available. Even stronger is the even form of control provided by taints and tolerations. For example, a node can be tainted so that only pods with a special marker are allowed to start on it. This can be used to reserve clean energy nodes for workloads duly marked as carbon-sensitive. This ensures that less critical jobs are not using valuable low-carbon capacity and that there is room to allow higher-priority, sustainability-aware jobs.

Like most other SDKs, there are custom schedulers available for use if desired. Developers can extend that scheduling layer, building entirely separate scheduling services on top of or instead of the already existing Kubernetes scheduling logic. These schedulers pull in external data sources and also have their own rules. For instance, a custom scheduler could put a hold on starting a nonurgent job until carbon levels go down below a certain limit or distribute workloads across multiple clusters if it depends upon surrounding emissions. In more advanced setups, organizations can embrace both internal and external logic. Carbon data is collected and analyzed by an external monitoring service, node labels are applied, and this simple API is exposed. Kubernetes uses these labels, together with preconfigured affinity rules, to schedule those pods intelligently.

A Practical Example of Architecture at Work

To understand how this works in the real world, think about a video processing service that renders jobs each night at some cloud. They have Kubernetes clusters in three geographically distinct locations: one in California, one in Germany, and one in Singapore. These locations communicate with a carbon-aware scheduling service, which monitors grid carbon intensity for each location using data provided by ElectricityMap's API. The system updates the labels showing whether a region's

electricity is 'low,' ‘medium,' or 'high' carbon intensity on each node in each cluster every ten minutes. Kubernetes looks at nodes labeled "low-carbon" and launches the nightly batch jobs once it's ready to fire them up. California might be selected if it had extremely strong solar generation that night. This job could be routed to Germany if they are using wind energy and not otherwise.

This system is automatic and bases its decision solely on the current state of the world, not on any internal metrics. As seen in comparative automation methods like image captioning, where external context is key to accurate interpretation, the outcome here is a process that considers each computer's computing needs while being environmentally aware without requiring manual intervention or complex reconfiguration to achieve the goal (Sukhadiya et al., 2018).

Architecture Approach Summary

Creating a brand-new platform is not a prerequisite for carbon-aware Kubernetes scheduling. It extends the tools and options already built into Kubernetes. Adding real-time carbon data to the mix and adjusting how nodes are labeled and prioritized allows us to create a more sustainable system to run workloads. This results in a smarter Kubernetes that makes decisions not based on CPU usage or available memory but on the impact that task will have on the world via affinity rules, custom schedulers, or external controllers. With more companies seeking to diminish their environmental footprint, these architecture patterns offer a repeatable, scalable, and cost-effective means for cloud operations to be linked closely to climate targets.

2. Methodology

2.1 Designing the Research Setup

Since carbon-aware scheduling in Kubernetes is still in practice, a well-defined and realistic test environment is needed to understand how it works (Burns & Tracey, 2018). The methodology aims to simulate workloads as they behave inside the typical Kubernetes cluster, with and without the application of carbon-aware logic. Doing so allows for measuring whether emissions can be reduced by incorporating carbon awareness, and if so, by how much. It also aids in the identification of any tradeoff between workload delay and system complexity. These steps begin with a Kubernetes cluster deployment on any cloud provider like Google Cloud Platform, Microsoft Azure, or AWS, or in a local lab environment using tools like Minikube or KIND (Kubernetes in Docker). A multi-region setup with nodes in at least two or three locations is advised to obtain a more accurate, meaningful result. A potential region can assume different energy grid conditions and is represented by each node.

2.2 Integrating Over Carbon Intensity Data Sources

One important part of the methodology is pulling real or near-real-time data on each node region's carbon intensity. They integrate third-party services like WattTime or ElectricityMap. These services offer APIs that provide current carbon intensity levels in particular geographies, determined based on grid usage and power source weights. Simple custom script or small service built to poll these APIs periodically (every 5 min) and update Kubernetes node labels about the node's region carbonity by indicating it as "low," "medium," or "high" with a certain carbon intensity. For instance, if a node is in an area where plenty of electricity is produced from renewable sources at that moment, it might be termed as 'low carbon.' The Kubernetes scheduler uses these labels when placing workloads. For accuracy, timestamps and carbon intensity values are logged during every update (Lu et al., 2017). This data is used to evaluate the performance and environmental impact of scheduling decisions during the later analysis phase.

The image below illustrates the interrelated factors that contribute to carbon emissions, from economic scale and population to energy use and GDP per capita. It contextualizes why tracking carbon intensity (of GDP and energy) is crucial to effective emission reduction strategies.

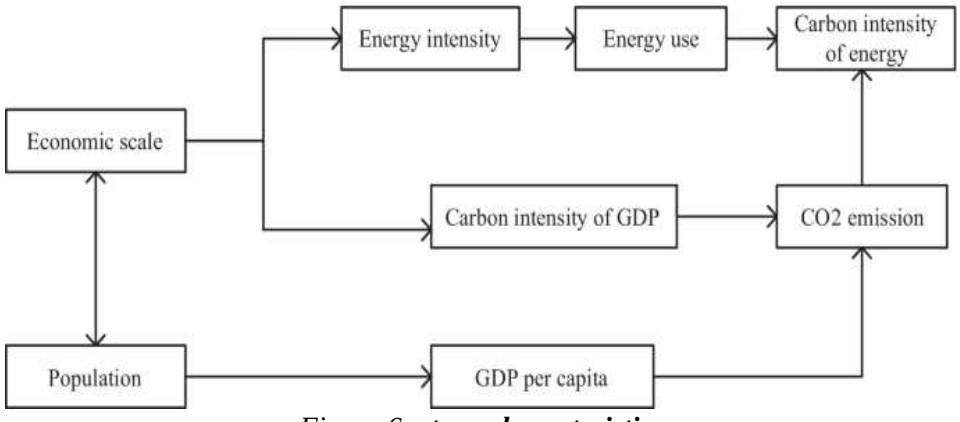

Figure 6: ***stage characteristics***

2.3 Scheduling Strategy and Experiment Design

The experiment will run in two phases. For first, the Kubernetes cluster runs with Traditional scheduling. In other words, tasks are scheduled entirely based on the resources available and independent of their carbon intensity. Most production environments today run the system as they would with any other platform. In the second phase, carbon-aware scheduling is triggered. It consists of applying node selection rules based on the carbon-intensity labels referenced above. For example, jobs may be constrained to only run on 'low carbon' nodes or prefer them when available. If no clean energy nodes can be found when needed, nonurgent jobs may be delayed until the external conditions improve. The use of phased experimentation, similar to approaches used in evaluating career feedback systems where tailored strategies are applied in stages, helps assess real-world outcomes (Karwa, 2023). Both phases employ the same workloads and scheduling intervals. The results are compared to measure total carbon emissions associated with the two approaches. To keep the comparison fair, all other settings—including CPU requests, limits on memory, and resource quotas—are left unchanged, reflecting the need for design consistency across parallel conditions (Karwa, 2024).

2.4 Metric Collected and Tools Used

Some of the experimental data includes several types of data. Next, the grid's actual carbon intensity at the time a job runs is monitored using emissions APIs, enabling the calculation of estimated emissions per task. Additionally, the duration of any job delays is assessed, along with any failures resulting from node unavailability or misconfiguration. System performance metrics such as CPU usage, job completion time, and node utilization are also recorded to verify that the carbon-aware system is still operable relative to its expectations. These metrics are collected with simple tools using Kubernetes monitoring, such as Prometheus and Grafana, and simple log collection, like Fluent or similar logging agents.

Technical metrics other than these are also noted as qualitative observations (Pendrill & Petersson, 2016). This encompasses the complex process of setting up the system, its maintainability, any unexpected behavior, and its limitations encountered during the tests. The performance of the carbon-aware system is then measured against three major criteria: reduction in estimated carbon emissions, performance of the workload in the carbon-aware system vs the original baseline system, and ease of implementation. These help determine when the benefits of production of carbon-aware schedules are worth the additional work and whether the proposed system can be adopted in real-world development and operations teams.

Table 5: ***Key Metrics Captured in the Experiment***

Metric	**Purpose**	**Tool Used**
Carbon Emissions per Task	Measure environmental impact	API Data + Logs
Task Completion Time	Compare system performance	Prometheus, Grafana
Node Utilization Rate	Measure resource efficiency	Kubernetes metrics
Job Delay Duration	Understand user impact of carbon-aware logic	Scheduler Logs
Error/Failure Rate	Assess operational reliability	Fluent, Logs

3. Results and Discussion

3.1. Case Studies and Real-World Applications

Industry Adoption by Major Cloud Providers

The tech industry now has a sector attempting to clean up its computing act, with several major cloud providers already deploying carbon-aware strategies (Lee et al., 2024). For example, Microsoft has adopted this by creating its Carbon Aware SDK, integrating it into some of its internal systems, and encouraging its developers to use this within their workflows. It is also a company aiming to go carbon-negative by 2030 and builds scheduling tools that enable workloads to be moved to carbon-aligned schedules based on the power grid they are running on. In particular, pilot projects were conducted through Microsoft Research, where software builds and batch processing jobs were delayed until lower-carbon windows appeared, based on forecast data from grid monitoring services.

Google, too, has invested heavily in clean energy tracking across its data centers and launched the 24/7 carbon-free energy matching concept. Google prefers to buy clean energy in real-time, matching the timing and location of its energy use. Google Cloud Platform's Kubernetes Engine doesn't offer carbon-aware scheduling natively yet, but it is building internal tools to route and delay workloads depending on regional energy conditions. They create a solid blueprint for how carbon-aware

principles can be baked into large-scale infrastructure. Amazon Web Services has also made progress in this direction. Unlike AWS, which hasn't yet released a dedicated, carbon-aware scheduler for Kubernetes, it does have carbon footprint tracking in its set of tools for its customers. It encourages the utilization of the cleanest available zones where possible. A path forward for customers who run container workloads on AWS Fargate or off EC2 instances is to use third-party emissions data to design intelligent schedulers that avoid regions with higher emissions intensity.

Academic Research and Open Source Projects

Outside corporate settings, research institutions and open-source communities have attempted carbon-aware computing, though with encouraging results. One striking example of this was seen in a collaboration between universities in Europe, where researcher's leveraged Kubernetes clusters located in separate countries to help test efficiently shifting workloads based on real-time electricity emissions data. The researchers found that 20 percent or more emissions could be slashed simply by delaying batch jobs for a few hours or sending them to a greener region, with virtually no performance loss.

Another community-led initiative, an open-source tool called the Kepler project (Kubernetes-based Efficient Power Level Exporter and Report helps monitor energy utilization in Kubernetes clusters. Although Kepler is primarily concerned with energy consumption, information can be derived from Kepler data to guide scheduling in support of carbon intensity sources. For example, it enables developers to build on top of Kepler's node-level power usage metrics with something like WattTime or ElectricityMap to favor nodes that are energy efficient and run on cleaner electricity. There is also a smattering of open-source plugins for Kubernetes trying to fill this space (Rahman et al., 2023). One example is custom schedulers that introduce logic for region-based emissions levels or integrations with the Carbon Aware SDK. Many of these are still early in development, but they all indicate a very strong interest in the developer community in building sustainable systems from the ground up.

As illustrated in the image below, achieving cost efficiency in cloud storage involves a six-step cyclical process.

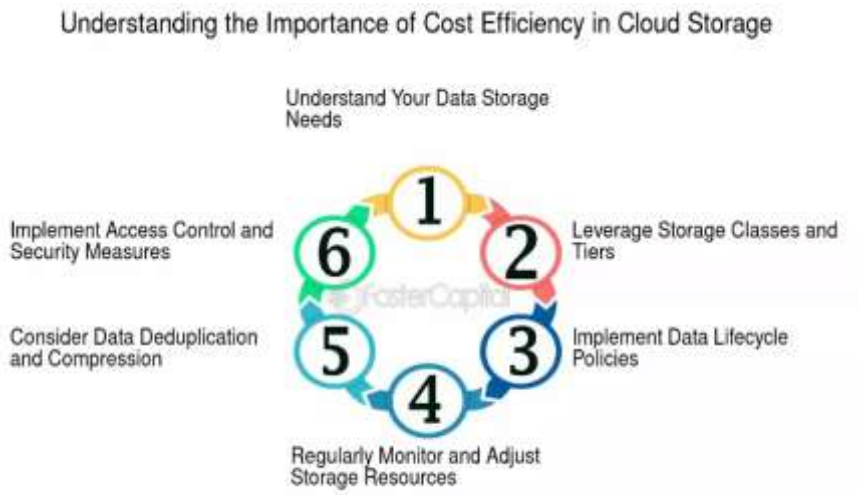

Figure 7: *Cost Efficiency through Cloud Computing*

Business use cases and operational improvements.

Carbon-aware scheduling is on its way to becoming a practical business tool. A real case is a digital media company that needs to run high-volume video rendering jobs every evening. The company integrated emissions data into its Kubernetes cluster. It changed its scheduling policies to delay rendering jobs by just one or two hours on high-carbon nights or run them earlier on low-carbon nights. They measured an estimated 18% drop in their emissions from computing jobs over several months and didn't see that impact on customer delivery times. A Fintech startup is one of them, and it runs batch data analytics on Kubernetes overnight. Following the introduction of carbon awareness to their workflow, they utilized node labeling and scheduling rules to steer clear of nodes situated in regions that routinely have high carbon intensity. In all regions, cleaning was delayed for jobs when emissions were especially high. Second, the team saw lower energy costs during off-peak hours – financial and environmental savings win-win.

A software development agency also experimented with carbon-aware CI/CD pipelines. They had a different way of running build and test jobs, instead of sprinting straight along when code was pushed. They checked the carbon intensity of the bit currently and ran scheduling logic to delay jobs

during dirty power. The delay worked Most of the time, and developers could override it if an urgent build were needed. This moved job timing significantly without any detrimental impact on developer productivity and release speed target (Meyer et al., 2017).

Lessons Learned from the Field.

Looking at all these examples reveals a few key lessons. The first good place to begin with carbon-aware scheduling is with flexible workloads. The lowest-hanging fruits are jobs that don't have to be run in real-time, e.g., batch processing, backups, and rendering. Second, real-time data is required. If accurate, up-to-date carbon intensity information isn't available, the scheduling system can't make reasonable decisions. The key to success is integrating with trusted APIs such as WattTime or ElectricityMap. Third, users want transparency and control. In all successful examples, developers or operators could override or bypass carbon-aware logic as necessary (Rethinagiri et al., 2015). Robot Flow combined automation with flexibility, allowing teams to implement the new system without feeling trapped readily. Finally, although the initial steps of carbon-aware scheduling can require much time and planning, the long-term benefits of fewer emissions and more energy cost management make it a worthy endeavor.

Limitations in Carbon Intensity Data

As one of the core building blocks of carbon-aware scheduling, access to region-specific and up-to-date, accurate carbon intensity data is important. However, acquiring high-quality data is still a major challenge. They say real-time emissions data from all countries or grid operators is unavailable. Because data may be delayed, incomplete, or average across large areas in certain regions, the data is not useful for making decisions about fine-grained scheduling. Services with great APIs, such as ElectricityMap and WattTime, are prone to downtimes or unreliable feed data if they rely on inputs from other energy providers. Furthermore, some of these APIs have fees for commercial usage, which could make them unsuitable for frequent use by small startups or academic projects. Carbon-aware workload scheduling becomes ineffective or even inaccurate if workloads are re-scheduled based on obsolete and inaccurate views of grid conditions, as without accurate and timely data, the assumptions made in carbon-aware scheduling become unreliable.

Kubernetes Technical Constraints

Although it's a flexible platform, Kubernetes does not natively understand metrics such as carbon. Thus, making Kubernetes carbon-aware may require customization or integration of external services. There are many effective techniques using node labeling, affinity rules, taints, and tolerations to influence scheduling, but ongoing updates and management are always required. If labels are not updated regularly with the current carbon conditions, they will cause the scheduler to make decisions based on stale data. It brings complexity to building a custom scheduler or integrating external logic into the kubelet or the control plane (Jakobović, 2023). This means developers must write and constantly update code to interact with emissions APIs, parse carbon intensity levels, and interpret them into scheduling rules. These tasks require more skills than just standard Kubernetes administration and will often involve a fair bit of heavy lifting on the engineering front. Moreover, organizations using managed Kubernetes services on cloud platforms usually lack control of some parts of the cluster, making the integration process more challenging.

As illustrated in the image below, Kubernetes offers numerous advantages that make it ideal for scalable, cost-effective deployments:

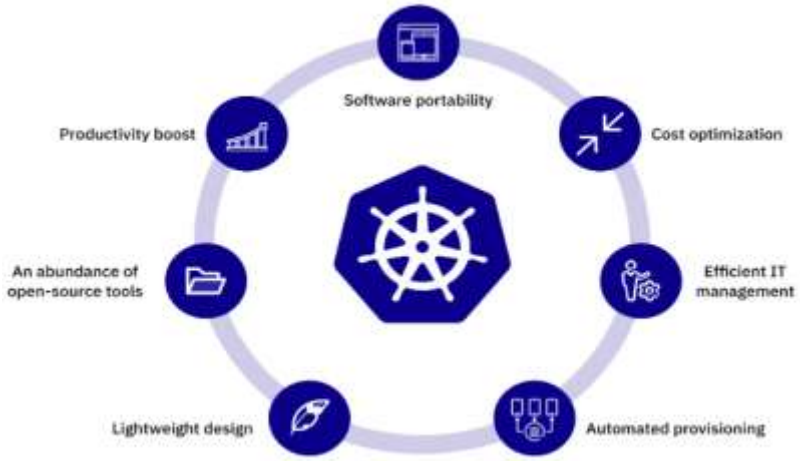

Figure 8: *Kubernetes and the Alternatives*

Operational Challenges and Cultural Resistance

It's one of the greatest hurdles from an operational point of view: getting teams to shift how they manage workloads and schedule work (Döbert, 2018). Most engineers are already accustomed to jobs being run on demand or within set time windows, usually optimized for speed and simplicity instead of environmental impact. Moving to such a carbon-aware model will likely entail delaying jobs or moving them to different parts of the world and will feel like a step backward on the control and responsiveness front. Additionally, they could result from organizational resistance due to fears of detrimental performance or reliability. For instance, consider a carbon-aware scheduler deciding not to run a task because the grid is currently powered by fossil fuels; there is always a chance that delivery times, user expectations, and system metrics could be affected due to the task not being executed immediately. If the delay is still just a few minutes, some teams would not be comfortable with an external data source hindering the run of their critical operations, even for a short amount. It takes time to earn trust in the system. In addition, given that the environmental benefits may not be transparent regarding business goals, carbon-aware practices can be difficult to align with business goals. In all but the simplest situations, it's unrealistic to think that decision-makers would give more weight to sustainability features over revenue drivers unless and until there are clear environmental savings, reputational advantages, or regulatory compliance requirements to comply with.

Lack of Standardization and Benchmarking.

A central limitation of the carbon-aware computing space is the lack of standardized tools, benchmarks, and best practices. There is currently no commonly accepted framework to measure the carbon impact of scheduling decisions inside Kubernetes. Still, there is a need to assess the outcome of a carbon-aware scheduling strategy. How the results are compared or how solutions can be reused across these organizations is hard because they use different metrics, data sources, and scheduling policies. Because there is no standardization, technology adoption slows down (Janssen et al., 2020). Often, developers and DevOps teams are left to build their solutions or use early-stage open-source projects that may not be well supported. However, the lack of standards also makes it hard to show in quantifiable terms how much carbon awareness adds up in terms of value, especially at the beginning, when savings might pale in comparison or accrue slowly. Fragmentation of the carbon-aware movement cannot take place without industry-wide collaboration and shared reporting formats. As each organization creates its own carbon labeling or scheduling system, the effort may be duplicated, and progression may be inconsistent.

Edge Cases and Inflexible Workloads.

Carbon-aware scheduling is not suitable for all workloads. Applications that need to respond in real-time, like video chats, live data processing, or online games, must respond in a timely manner, or it will affect the user experience. Carbon-aware strategies are hard to apply to these types of jobs unless they are supported by backup systems or regional replication. Some industries are tightly regulated, so it isn't easy to adjust where or when their data is processed. For instance, companies operating in financial services or healthcare may face legal restrictions on data residency or on the lag time for processing, which can make it difficult to offload workloads across regions or defer execution until later in the day.

Table 6 ***: Challenges of Carbon-Aware Scheduling***

Category	Challenge Description	Impact Area
Data Quality	Inconsistent or unavailable emissions data	Scheduling accuracy
Technical Integration	Complex setup and custom scheduler maintenance	DevOps overhead
Organizational Adoption	Team resistance to delays or job relocation	Workflow compliance
Legal and Policy Limits	Data sovereignty and compliance issues	Regional workload flexibility

3.2 Recommendations and Future Work

Practical Steps for Teams and Developers

The first and most important step for teams that want to start their carbon aware journey is figuring out which workloads are flexible in their systems (Patel et al., 2024). That doesn't mean every task has to run immediately and in the right place. Carbon aware scheduling is a good fit for batch jobs, reporting processes, daily data exports, backups, software builds, and media rendering because they can accommodate delays or be rerouted without impacting users. After discovering flexible workloads, developers can add real-time real-time carbon intensity insights to a Kubernetes environment. Reinventing the wheel is unnecessary. In many places worldwide, APIs from trusted services such as WattTime and ElectricityMap

provide up-to-date data on grid emissions. These APIs can be polled by a simple background service that will then label Kubernetes nodes with current carbon conditions. Next, it uses node affinity and other native Kubernetes scheduling knobs to prefer cleaner nodes and push out tasks on need.

It's also necessary to ensure the team understands why to curb carbon emissions during work. Developers and DevOps engineers should be encouraged to think about sustainability as another performance metric along with cost, latency, and uptime. However, these environmental factors become visible and actionable by integrating carbon metrics into dashboards, reports, and alerts. Being able to see, in simple visualizations, when and where workloads ran compared to how carbon-intensive it was helps build awareness and trust in the system.

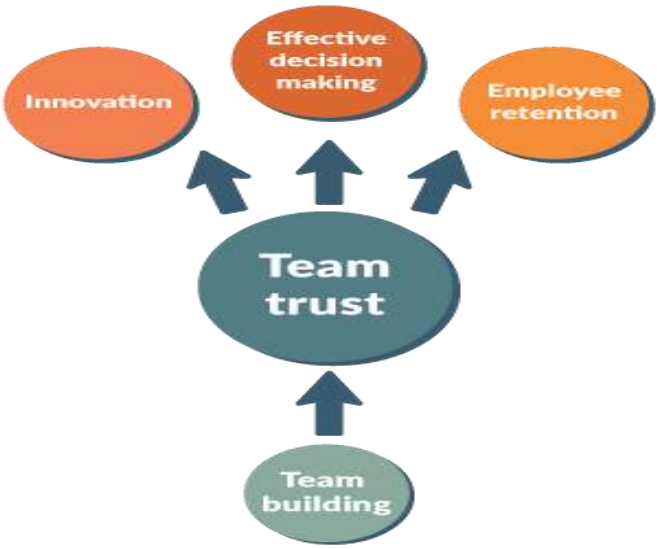

Figure 9: *why team building is important*

Integration with CI/CD Workflows

This carbon-aware scheduling gets even more effective when turned into the fabric of the software delivery process itself (Hallberg, 2024). Continuous Integration and Continuous Delivery (CI/CD) pipelines, often used to trigger large volumes of compute-heavy workloads, can lead to pipeline failures that become large wasted computing in the common case of many developers pushing code many times a day. Pipelines can delay builds or tests by a few minutes using carbon checks before starting a job, and the job will start once cleaner electricity is available. The lightweight scripts or plugins query emissions APIs once before running a job. This also allows developers to override delays if necessary without sacrificing using regulations. Ultimately, these changes can be measured, especially in large organizations with hundreds or possibly thousands of builds per day. Beyond CI/CD, tools like Terraform, Pulumi, or Helm can similarly be updated to deploy carbon-aware settings by default. Making sustainability a part of the provisioning process also means that it's applied to each new environment out of the gate, whether attaching node affinity rules to pods or installing a monitoring service for emissions data.

Future Research and Development Areas

While the carbon-aware scheduling foundations are now in place, many research areas remain where more research and design work will lead to higher efficacy and adoption. Forecast accuracy is amongst the most important areas. Systems will be able to plan rather than always react in real time and give more precise, localized predictions of carbon intensity. Even if an upcoming low-carbon window is hours away, better forecasting can help schedule the workload. A problem that remains ripe for improvement is the construction of Kubernetes-native carbon-aware controllers that can be quickly deployed, set up, and managed. At this stage, much of the work in this space is either custom-built or handled through first-stage open-source projects. More widespread use and trust might be fostered by embracing a well-supported community standard similar to what Prometheus and Fluent have done with monitoring and logging. There is still ample opportunity for designing hybrid systems that incorporate environmental scheduling alongside other sustainability-based techniques for resource selection, automatic resource scaling concerning environmental factors, or general energy efficiency techniques. However, carbon awareness could take another leap if these systems can be integrated with cloud provider APIs that expose regional energy mix data, even in multicolored or federated Kubernetes environments.

Opportunities and Standards Industry-Wide

However, carbon-aware scheduling will require more than technical innovation to scale truly: Cloud providers, standards bodies, and industry groups must work together. One promising path is developing shared sustainability metrics and benchmarks that developers and companies can apply to measure and report their progress. These metrics could be things like the average carbon savings per workload, carbon avoided via shifting workload, or the percent of compute tasks run in low-carbon windows (Sukprasert et al., 2024). Cloud providers should also provide built-in support for carbon-aware workload placement. If AWS, Azure, and Google Cloud offer regional carbon intensity data in their APIs and provide it to their Kubernetes services so that it is easily accessible for all types of users, then that will revolutionize the effort to adopt these. Education and advocacy are the key. There needs to be more sustainable computing and carbon-aware content in conferences, technical blogs, open source communities, and developer platforms. The more engineers feel the environmental impact of their choices, the more their carbon-blind behaviors will be replaced by more sustainable defaults (Schooling et al., 2023).

3. Conclusion

Timely and practical, carbon-aware Kubernetes scheduling is taking shape as a way to operate cloud computing in an environmentally sustainable way. The idea is simple at its core: moving the timing and location of computing tasks to when and where clean electricity can be consumed. There's less than seems in this adjustment, but it's hardly trivial. When applied to thousands of data centers and millions of workloads, it can significantly reduce associated carbon emissions from digital infrastructure. Because Kubernetes has the power to schedule and run a workload anywhere it can, integrating real-time carbon intensity data into its scheduling logic enables developers and infrastructure teams to choose where and when to run workloads smarter and save emissions without sacrificing performance. Currently, tools like ElectricityMap and WattTime provide live and future predictions of how clean the power grid is in different regions. With strong flexibility, Kubernetes is a good platform to react to this data. Developers can create environmentally aware systems with things they already use, like node labeling, affinity rules and taints, and custom schedulers. These tools are the first step towards a more responsible and carbon-aware computing model that makes for a powerful alternative when coupled with their open-source practical plugins and automation scripts.

Cloud workload emissions can be reduced by anywhere from 10% to 30%, depending on how flexible the workloads are and how much the carbon intensity of the local grid varies when implementing carbon-aware scheduling through case studies. However, even small changes like putting off a task by an hour or moving it off-site or to a different time zone can yield measurable results when applied consistently across a company's entire infrastructure. This approach, in addition, solves the problem of cost efficiency. When renewables are abundant, clean energy isn't just better for the planet — it's often cheaper, too. Besides reducing emissions, companies taking on this model also achieve energy savings and have a stronger hand in reporting on sustainability to stakeholders such as customers, partners, and regulators. Second, it is important to note that carbon-aware scheduling sacrifices neither performance nor stability. Batch processing, backups, machine learning training, and video rendering can be rescheduled with minimal impact on such flexible workloads. In cases where immediate executions are important, systems can be designed to include fallback rules to prevent critical tasks from being put off. Because the approach achieves this balance between sustainability and reliability, it is practical for real-life use.

Carbon-aware scheduling fits naturally within a broader sustainability effort as pressure mounts on the tech industry to reduce its environmental impact. This complements corporate ESG goals, net zero commitments, and ongoing investment in renewable energy procurement. Kubernetes is central to cloud-native development and operations today, making it a strategic and ethical priority to see that it evolves in such a way that it can support sustainable practices in the future. In the future, carbon-aware scheduling will only be successful with industry-wide collaboration to improve forecasting models, standardize emissions data integration, and simplify adoption through developer-friendly tools. The vision is that carbon-aware logic will eventually not be a specialty feature but a baseline expectation in operating the digital systems that underlie our world.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Abdelmassih, C. (2018). Container Orchestration in Security Demanding Environments at the Swedish Police Authority.

[2] Ahmed, K., Ren, S., He, Y., & Vasilakos, A. V. (2015). Online resource management for carbon-neutral cloud computing. In Handbook on Data Centers (pp. 607-630). New York, NY: Springer New York.

[3] Buchanan, W., Foxon, J., Cooke, D., Iyer, S., Graham, E., DeRusha, B., ... & Mathews, N. (2023). Carbon-aware computing.

[4] Burns, B., & Tracey, C. (2018). Managing Kubernetes: operating Kubernetes clusters in the real world. O'Reilly Media.

[5] Chavan, A. (2022). Importance of identifying and establishing context boundaries while migrating from monolith to microservices. Journal of Engineering and Applied Sciences Technology, 4, E168. http://doi.org/10.47363/JEAST/2022(4)E168

[6] Chavan, A. (2023). Managing scalability and cost in microservices architecture: Balancing infinite scalability with financial constraints. Journal of Artificial Intelligence & Cloud Computing, 2, E264. http://doi.org/10.47363/JAICC/2023(2)E264

[7] Chen, H., Wang, R., Liu, X., Du, Y., & Yang, Y. (2023). Monitoring the enterprise carbon emissions using electricity big data: A case study of Beijing. Journal of Cleaner Production, 396, 136427.

[8] de Vasconcelos Danen, J. P. (2018). Economic potential of human motion for electricity production in gymnasiums (Master's thesis, Universidade NOVA de Lisboa (Portugal)).

[9] Dhanagari, M. R. (2024). MongoDB and data consistency: Bridging the gap between performance and reliability. Journal of Computer Science and Technology Studies, 6(2), 183-198. https://doi.org/10.32996/jcsts.2024.6.2.21

[10] Dhanagari, M. R. (2024). Scaling with MongoDB: Solutions for handling big data in real-time. Journal of Computer Science and Technology Studies, 6(5), 246-264. https://doi.org/10.32996/jcsts.2024.6.5.20

[11] Döbert, T. (2018). Doing More With Less: Techniques to Manage Team's Increasing Workload.

[12] Goel, G., & Bhramhabhatt, R. (2024). Dual sourcing strategies. International Journal of Science and Research Archive, 13(2), 2155. https://doi.org/10.30574/ijsra.2024.13.2.2155

[13] Guyon, D. (2018). Supporting energy-awareness for cloud users (Doctoral dissertation, Université de Rennes).

[14] Hallberg, M. (2024). Carbon-aware Scheduling in Kubernetes.

[15] Hurwitz, J. S., & Kirsch, D. (2020). Cloud computing for dummies. John Wiley & Sons.

[16] Jakobović, K. (2023). PROPOSAL FOR DEVELOPMENT OF A CUSTOM KUBERNETES OPERATOR (Doctoral dissertation, Algebra Univerity).

[17] Janssen, M., Weerakkody, V., Ismagilova, E., Sivarajah, U., & Irani, Z. (2020). A framework for analysing blockchain technology adoption: Integrating institutional, market and technical factors. International journal of information management, 50, 302-309.

[18] Jones, N. (2018). How to stop data centres from gobbling up the world's electricity. nature, 561(7722), 163-166.

[19] Karkkainen, B. C. (2019). Information as environmental regulation: TRI and performance benchmarking, precursor to a new paradigm?. In Environmental law (pp. 191-304). Routledge.

[20] Karwa, K. (2023). AI-powered career coaching: Evaluating feedback tools for design students. Indian Journal of Economics & Business. https://www.ashwinanokha.com/ijeb-v22-4-2023.php

[21] Karwa, K. (2024). Navigating the job market: Tailored career advice for design students. International Journal of Emerging Business, 23(2). https://www.ashwinanokha.com/ijeb-v23-2-2024.php

[22] Khan, A. (2017). Key characteristics of a container orchestration platform to enable a modern application. IEEE cloud Computing, 4(5), 42-48.

[23] Konneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. International Journal of Science and Research Archive. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient

[24] Kumar, A. (2019). The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management, 6(6), 118-142. Retrieved from https://ijcem.in/wp-content/uploads/THE-CONVERGENCE-OF-PREDICTIVE-ANALYTICS-IN-DRIVING-BUSINESS-INTELLIGENCE-AND-ENHANCING-DEVOPS-EFFICIENCY.pdf

[25] Lee, B. C., Brooks, D., van Benthem, A., Gupta, U., Hills, G., Liu, V., ... & Yu, M. (2024). Carbon connect: An ecosystem for sustainable computing. arXiv preprint arXiv:2405.13858.

[26] Lu, X., Ota, K., Dong, M., Yu, C., & Jin, H. (2017). Predicting transportation carbon emission with urban big data. IEEE Transactions on Sustainable Computing, 2(4), 333-344.

[27] Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., & Fritz, T. (2017). The work life of developers: Activities, switches and perceived productivity. IEEE Transactions on Software Engineering, 43(12), 1178-1193.

[28] Nyati, S. (2018). Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. International Journal of Science and Research (IJSR), 7(10), 1804-1810. Retrieved from https://www.ijsr.net/getabstract.php?paperid=SR24203184230

[29] Pamadi, E. V. N., Khan, S., & Goel, E. O. (2024). A comparative study on enhancing container management with Kubernetes. International Journal of Advanced Research and Interdisciplinary Scientific Endeavours, 1(3), 116-133.

[30] Patel, P., Gregersen, T., & Anderson, T. (2024). An agile pathway towards carbon-aware clouds. ACM SIGENERGY Energy Informatics Review, 4(3), 10-17.

[31] Pedelty, M. (2016). A song to save the Salish Sea: Musical performance as environmental activism. Indiana University Press.

[32] Pendrill, L., & Petersson, N. (2016). Metrology of human-based and other qualitative measurements. Measurement Science and Technology, 27(9), 094003.

[33] Rahman, A., Shamim, S. I., Bose, D. B., & Pandita, R. (2023). Security misconfigurations in open source kubernetes manifests: An empirical study. ACM Transactions on Software Engineering and Methodology, 32(4), 1-36.

[34] Raju, R. K. (2017). Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR), 6(2). https://www.ijsr.net/archive/v6i2/SR24926091431.pdf

[35] Rethinagiri, S. K., Palomar, O., Sobe, A., Yalcin, G., Knauth, T., Gil, R. T., ... & Milojevic, D. (2015). ParaDIME: Parallel distributed infrastructure for minimization of energy for data centers. Microprocessors and microsystems, 39(8), 1174-1189.

[36] Sardana, J. (2022). Scalable systems for healthcare communication: A design perspective. International Journal of Science and Research Archive. https://doi.org/10.30574/ijsra.2022.7.2.0253

[37] Schooling, J., Geddes, R., Frawley, D. D., Mair, R. J., O'Rourke, T. D., Powrie, W., ... & Threlfall, R. (2023). The Role of Funding, Financing and Emerging Technologies in delivering and managing infrastructure for the 21st Century.

[38] Singh, V. (2022). Advanced generative models for 3D multi-object scene generation: Exploring the use of cutting-edge generative models like diffusion models to synthesize complex 3D environments. https://doi.org/10.47363/JAICC/2022(1)E224

[39] Singh, V. (2022). Integrating large language models with computer vision for enhanced image captioning: Combining LLMS with visual data to generate more accurate and context-rich image descriptions. Journal of Artificial Intelligence and Computer Vision, 1(E227). http://doi.org/10.47363/JAICC/2022(1)E227

[40] Smith, R., & World Economics Association. (2016). Green capitalism: the god that failed. London: College Publications.

[41] Sukhadiya, J., Pandya, H., & Singh, V. (2018). Comparison of Image Captioning Methods. INTERNATIONAL JOURNAL OF ENGINEERING DEVELOPMENT AND RESEARCH, 6(4), 43-48. https://rjwave.org/ijedr/papers/IJEDR1804011.pdf

[42] Sukprasert, T., Souza, A., Bashir, N., Irwin, D., & Shenoy, P. (2024, April). On the limitations of carbon-aware temporal and spatial workload shifting in the cloud. In Proceedings of the Nineteenth European Conference on Computer Systems (pp. 924-941).

[43] Turek, T., Dziembek, D., & Hernes, M. (2021). The use of IT solutions offered in the public cloud to reduce the city's carbon footprint. Energies, 14(19), 6389.

[44] Ungureanu, O. M., Vlădeanu, C., & Kooij, R. (2019, July). Kubernetes cluster optimization using hybrid shared-state scheduling framework. In Proceedings of the 3rd International Conference on Future Networks and Distributed Systems (pp. 1-12).

[45] Wang, Y., Qiu, J., & Tao, Y. (2021). Optimal power scheduling using data-driven carbon emission flow modelling for carbon intensity control. IEEE Transactions on Power Systems, 37(4), 2894-2905.

[46] Wizelius, T. (2015). Developing wind power projects: theory and practice. Routledge