SUMMARY

In a time where cloud computing is increasing in popularity, it is important to consider the environmental impact which occurs as a result. Many industry leaders are aware of the environmental effects and are pledging to save energy and minimize carbon emissions where possible.

The motivation behind this paper is to investigate which potential savings in energy and carbon emissions can be implemented by extending the Kubernetes scheduler, which is a popular tool for the orchestration of workloads and servers. The increase of Artificial Intelligence aligns with the need for more computational power and data centers. As a result, orchestration platforms such as Kubernetes could potentially have a positive impact in reducing the energy used from the main grid in computations by providing it with more knowledge about the current state of renewable energy in its scheduling process.

In our contribution, we propose a system architecture for simulating data centers as part of a microgrid by extending an existing microgrid simulation framework to work in real time, utilizing KWOK to simulate nodes and pods, and provide a worktrace runner to emulate realistic worktraces. Additionally, a Kubernetes Scheduling Framework Plugin is proposed which aims to utilize renewable energy available, by using the state of the microgrids in its scoring.

By looking at realistic simulated solar data from the Technical University of Denmark, we have utilized a microgrid simulation framework to study the effects of our contribution in real time. We are able to study the effects of introducing renewable energy production and storage as knowledge for the scheduler to select the most optimal node and reducing the non-renewable energy used. Furthermore, a method of evaluation can be done by providing a real worktrace containing scheduled jobs, resource usage and computation time, and thus it is possible to compare and evaluate against the default Kubernetes schedulers performance.

We have created a system architecture that allows running realistic worktraces on a Kubernetes cluster with simulated nodes and pods, combined with a real time microgrid simulation that updates its state accordingly. This is achieved using Kubernetes WithOut Kubelet (KWOK), which makes it possible to have a Kubernetes cluster with normal control components, but simulate the nodes and pods. Using the Kubernetes Scheduling Framework, we have created a plugin that gets the current state of each nodes associated microgrid, and uses that information in the scheduling decision. When a pod has been scheduled, the plugin informs the microgrid simulation using the same API.

The goal of our scheduling plugin was to utilize renewable energy when available, by giving a higher score to nodes on microgrids with high current renewable output and battery charge. This was tested by running a two week long simulation based upon a worktrace provided by Azure with ≈ 900 nodes. Our proposed Kubernetes plugin was able to utilize 20.34% more renewable energy in the microgrid compared to the default Kubernetes scheduler. This is however a single example of scheduling plugin designed with a very specific goal in mind. Permutations or different potential alternative goals are discussed in Future Works, as this project could be extended and improved in many different ways.

Renewable Kubernetes Scheduling Simulating Microgrids With Nodes

Simon Malgo Pronk Andersen, Laurits Christian Bang Mumberg
Department of Computer Science
Aalborg University, Denmark
smpa20@student.aau.dk
lmumbe19@student.aau.dk

***Abstract*—Data centers are expected to consume $3 - 13\%$ of the global electricity in 2030. Thus, different measures such as reducing the PUE value and utilizing renewable energy sources should be explored to limit the environmental impact of cloud computing. By configuring data centers as microgrids, where they are able to operate both in connected and disconnected mode from the main grid, utilizing electrical storage and renewable energy production, carbon reductions might be possible. Workload schedulers, by default, are not yet fitted with the capabilities of knowing the source of energy and thus cannot act accordingly.**

Therefore, we propose a way of using the Scheduling Framework for Kubernetes to create a plugin that takes these factors into consideration when scheduling a workload. Additionally, we propose a framework for simulating microgrids and nodes, where the effects of scheduling are observable.

By simulating a large number of microgrids and configuring them with servers (also known in this paper as nodes), our plugin determines the best possible microgrid with regard to renewable energy production and battery charge to schedule a workload. The workload is then run on the simulated node, updating the energy consumption in the microgrid in real time.

Based on a real Azure worktrace used in our experiments, we have shown that our proposed plugin is able to utilize 20.34% more renewable energy in comparison with the default Kubernetes scheduler. Further study could introduce different workload types and specific hardware requirements, a non-heterogeneous node specification or different, better-fitting renewable energy sources tailored to the geographical region.

***Index Terms*—Kubernetes Scheduling, Scheduling Framework, Microgrids, Microgrid Simulation, Cloud Computing, Renewable Energy, Kubernetes Plugin**

I. Introduction

Data centers were estimated to consume $1.1 - 1.5\%$ of the global electricity which amounts to $203 - 273TWh$ in 2010 [1]. In comparison to 2010, it is expected that the electricity consumption of data centers in regard to the global electricity is expected to grow between $3 - 13\%$ in 2030 [2].

Most data centers are already operating with a Power Usage Effectiveness (PUE) value of near 1.0 which leaves little to no room for PUE improvements [3]. As such, looking for optimizations in the energy source [4] could be the next step to lessen the environmental impact of cloud computing. By imagining data centers as part of microgrids, the energy consumed could become more environmentally friendly due to the reliance of renewable energy sources.

In our preliminary research paper [5], we explored the potential for an environmentally friendly scheduling policy using a custom plugin for Kubernetes and microgrids.

The results of our preliminary study were not without their limitations, due to the quality of the microgrid simulation and experiments conducted. However it served as a proof-of-concept and laid the foundation for future research into scheduling workloads in Kubernetes through combining data centers and microgrids in a more renewable way.

Our study makes the following contributions.

1) **Microgrid Simulation Framework**. We have extended and used the open-source framework *python-microgrid* [6], [7] which can simulate a number of running microgrids. Our extension allows the addition of configurable modules called nodes to each microgrid which can have dynamic power consumption in correlation to the workload scheduled on said node.
2) **Scheduling Framework Plugin**. We have created a Kubernetes plugin that utilizes the microgrid state received from the simulation to make scheduling decisions based upon renewable energy availability and battery. Additionally, the plugin posts information about scheduled jobs to the simulator to keep it updated in real time.
3) **Evaluate Solution with Workload Trace from Azure**. Using a workload trace from Azure with clearly defined hardware specifications, we distribute data centers across 152 simulated microgrids and run the trace on a KWOK cluster to investigate the effects of our plugin.

II. Preliminaries[1]

A. Microgrid

As per the U.S. Department of Energy, a microgrid is defined:

> *... as a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid.* [8], [9]

Microgrids can vary in size, ranging from a single residential home with a small generation capacity, to a large community of hundreds residential homes with a very large generation capacity [10]. A unique feature of microgrids is the ability to switch between being connected to the energy grid or completely isolated relying on its own power generation [10].

[1] Subsection II-A and Subsection II-C are re-used background information from the preliminary research by *Andersen et al.* [5].

B. Data Centers

Like microgrids, data centers also vary in size depending on their hardware configuration and type of workloads. Artificial Intelligence (AI) requires more computational power [11] than hosting a simple web application and naturally requires additional hardware resources. A Cloud Native Computing Foundation (CNCF) project like Kepler [12] which can monitor power consumption of a cluster can be utilized but only after the data center is set up.

Enterprise Infrastructure Planning Tools (EIPTS) [13] can assist when designing data centers by providing estimates for power usage at both idle and $0 - 100\%$ resource utilization of different workload types and hardware configurations.

1) Energy Source: When data centers consume electricity from the grid, the environmental impact of the electricity can be significantly different based on how it is produced. This is supported by Figure 9 in Appendix B, where we can see the percentage of renewable energy sources in the grid compared to the carbon intensity for each region. The carbon intensity is the gCO_2eq/kWh meaning the amount of CO_2 created per kWh. Therefore, to make data centers more environmentally friendly, measures such as choosing providers that provide electricity from sustainable sources can be taken rather than relying on the uncertainties of the grid. Additional local measures can also be taken, such as installing and using Photovoltaic Panels (PVs) to power the data center which can make it less reliant on the grid when local renewable energy is available.

C. Kubernetes

Kubernetes [14] is an open source container orchestration tool originally developed at Google, but now maintained by the CNCF [15]. It takes care of a lot of the work needed to run a distributed system, including configuration, scaling and self-healing [16]. It utilizes the concept of containers, which are software packages containing everything needed to run an application including the code, the required runtime and libraries, making them independent of the hosts infrastructure. In Kubernetes, these containers are deployed to nodes as pods, where a pod contains one or more containers. Kubernetes is dependent on a few essential components where some run on the node and others on the main control plane. The components on the control plane manage the overall state of the cluster, with each of them having their own set of responsibilities. The component that is of the biggest interest to this project is the scheduler [17]. The scheduler is responsible for binding unassigned pods to nodes, which is also referred to as scheduling within Kubernetes.

D. Extending the Scheduler

The Kubernetes scheduler can be extended in a variety of ways [18]. In the preliminary study, these ways were explored to find the best match for this project. Extending the scheduler with a plugin using the scheduling framework was deemed to be the best option, due to its support from the Kubernetes SIG, its ability to still use functionality from the default scheduler, and variety of extension points [19]. In addition, it allows assigning a high weight to our plugin, while still letting default plugins influence the score when the state of multiple microgrids are very similar.

E. Problem Statement

By simulating data centers as part of microgrids $G = \{G_1, G_2, ..., G_i\}$ in different geographical regions where G_i is a single microgrid containing at least one node, is it possible, using the scheduling framework to create an intelligent scheduling plugin with a scoring algorithm that utilizes the renewable energy state of microgrids and a real-life workload trace.

The goal of this paper is to extend the preliminary research done by *Andersen et al.* [5] by greatly scaling the experiment, using realistic data, and making the microgrid simulation update in real time based upon the workload.

III. Methodology

Our Methodology section is structured as follows, Subsection III-A introduces the microgrid simulation framework utilized in our experiments. Subsection III-B, Subsection III-C and Subsection III-D introduce the dataset utilized for our renewable energy production used in our microgrids, and how we sized the renewable generation installations for each microgrid and their energy storage. In Subsection III-E we cover how we utilized Kubernetes Without Kubelet (KWOK) to simulate nodes and pods, which we use in our workload experiments. Finally, Subsection III-F highlights the entire architecture of our experiment setup, and covers how each part is connected.

A. Python Microgrid Simulation

In our preliminary research, we discussed two software to simulate a microgrid, namely python-microgrid [6], [7] and Homer Energy Pro [20]. The latter worked fine as an introductory tool for microgrids and how they should be configured for non-electrical engineers. However, the pitfalls of using this tool were primarily:

1) The location specific weather data used in the configured renewable energy source was outdated.
2) The simulation could not run in real-time and thus the effects of scheduling a workload could not be observed on the battery state.
3) The microgrid could only be configured with approved modules from Homer, which meant that our need for custom modules such as servers with varying electricity consumption was not supported.

As such we have decided to use the open-source framework python-microgrid, where we have the opportunity of adding our custom modules to represent a node, which can dynamically affect the battery state through the simulation. Another advantage of using python-microgrid over Homer Energy Pro, is the ability to run the simulation in real-time. This makes it possible to observe the changes and impact our node has on the microgrid whenever it receives a workload.

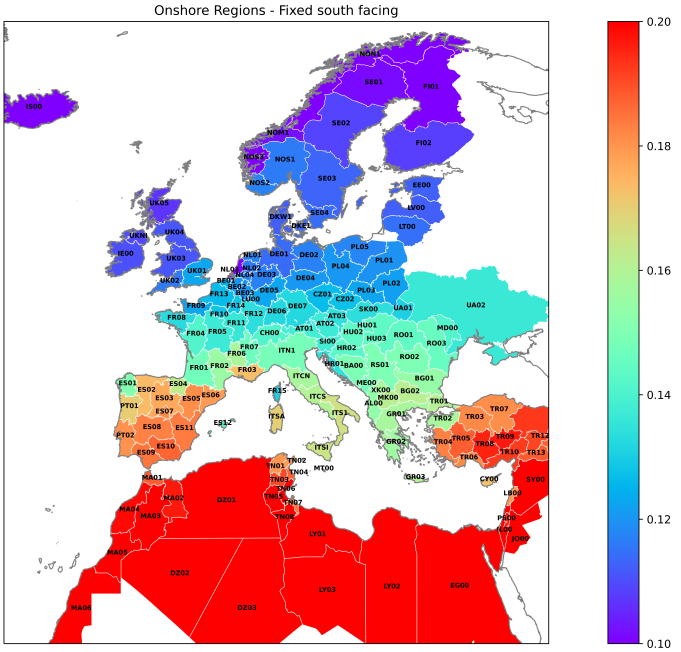


Figure 1. Overview of installation regions with mean annual utilization value [23].

B. Renewable and Carbon Emission Data

To ensure that our renewable energy module can produce realistic results in the microgrid, realistic data is necessary. Renewable energy sources can be volatile due to the dynamically changing weather and can as such create data variance, which aligns with our need of having varying data, which in our case will be used for distributing workloads.

To this end, we have opted to use a data set from a collection tied to the Technical University of Denmark (DTU) [21], [22], which includes solar data across the European region. Thus we consider and model PVs as the renewable source in this study. The specific data set [23] we utilize from DTU spans over 38 years from January 1st 1982 to December 1st 2019 consisting of hourly solar data across 49 countries and 152 regions (Figure 1) in percentages ranging from $0.0 - 1.0$, where the values are equal to the current utilization, with 1.0 being the maximum possible output. The utilization value is defined as a percentage of the maximum output of a renewable energy source which is produced at a given time. The mean value of the utilization can be seen in the y-axis in Figure 1, which is higher on average in the more sunny regions of the south. Due to the sheer length of the dataset, it has been pruned to only contain data from January 1st 2016 to December 1st 2019 and transformed to 10-minute intervals using forward fill. In total, there are 17761 data points for each region where each row contains a timestamp and the utilization value at that timestamp. The utilization value of four random regions over a 3-day period can be seen visualized in Figure 2.

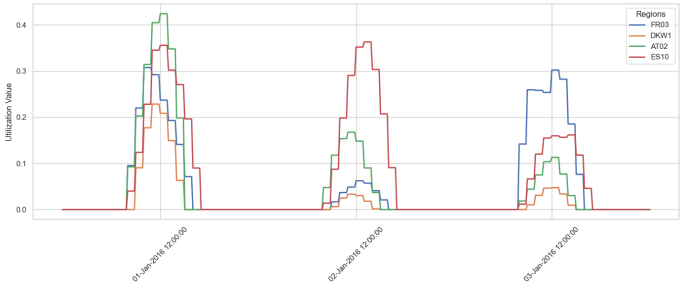


Figure 2. A 3-day utilization value variation for regions FR03, DKW1, AT02, and ES10.

C. Renewable Generation Capacity

In our case, we want to define the generation installations for each region in the data set[2]. This is due to our need of selecting the appropriate installation capacity to sustain the consumed electricity of our workloads. Each microgrid is in our case configured with a PV which is responsible for the renewable energy production. Several nodes are configured such that their power consumption ranges from $120 - 370W$ for each node, totaling a load of $720 - 2200W$ in each microgrid which should be partially sustained by the PV.

The utilization value is denoted as the variable UV and the data set assumes a generic PV module. Based on the daily energy consumption of our nodes and the UV rarely nears 1.0, we have defined the maximum generation capacity (P_{pv}) of the PV being $600W$ for each node with the maximum possible output being $3600W$. Thus, the calculation of how much renewable power (Watts) is generated at a given time, e.g. Total Generation Capacity (TGC) can be seen in Equation 1.

$$TGC = UV * P_{pv} \tag{1}$$

D. Energy Storage

As renewable energy production with PVs can be quite stochastic, it is important to consider storage systems which can save the renewable energy for later, if it is not consumed immediately. Thus we have also decided to implement an electrical battery storage in each microgrid in the simulation. Correctly sizing the battery can be challenging, due to the multiple variables involved. However, the following Equation 2 from [24] introduces a formula in Ampere-hours (Ahr) for exactly this purpose.

$$M_{batt} = \frac{A_d * E_L}{N_{batt} * N_{inv} * DoD * V_s} \tag{2}$$

In Equation 2 [24], A_d is the number of autonomous days, meaning days where the battery can supply energy without relying or recharging from the renewable source or grid. E_L is the estimated daily energy demand, N_{batt} is the battery efficiency, N_{inv} is the inverter efficiency. Depth of Discharge

[2]Excluding regions ending with 00 as this is a country-level aggregation [23] (except the regions with a single installation) and Ukraine due to lack of recent data.

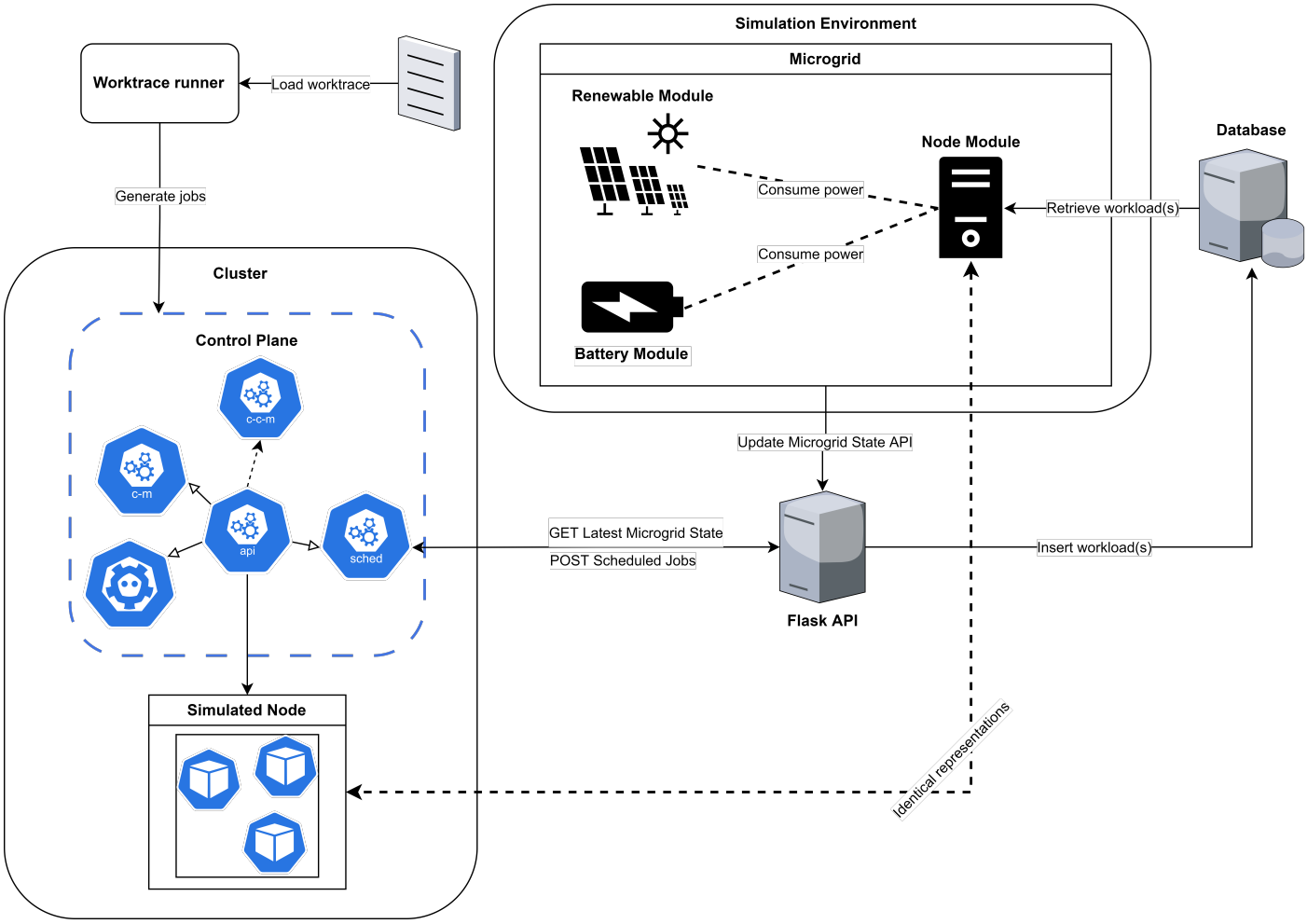


Figure 3. System Architecture

(DoD) is the allowed percentage of the battery which is discharged compared to the maximum capacity and V_s is the system voltage.

Based on Equation 2, we have defined the variables in the formula as the following. A_d is defined by the average job length in our workload trace [25], which is 10.2 hours $\approx$ 0.24 days. E_L is defined as the daily maximum power consumption of our nodes, which is $2184W * 24h$ and is thus $52416Wh$. N_{batt} is set to 0.9 and N_{inv} is set to 1.0 due to it not being definable in the python-microgrid library. The DoD is defined as 0.6 while the V_s is configured to 230 volts. As such, the battery capacity can be calculated as seen in Equation 3.

$$101.29Ahr \approx 23296.7Wh = \frac{0.24 * 52416}{0.9 * 1.0 * 0.6 * 230} \quad (3)$$

For simplicity and to understand the reasoning behind the scheduling choices, we utilize a homogeneous microgrid architecture which includes a fixed battery size disregarding geographical location and utilization values. For future research, the effects of having varying battery sizes could be a topic to explore, as we introduce in Subsection VIII-D.

E. KWOK and Simulated Pods

KWOK is as the name implies a toolkit for running clusters with simulated nodes and pods [26]. It is maintained by the Kubernetes SIG, and was created with the goal of having a lightweight platform for large scale Kubernetes experiments. It is compatible with all tools that use the normal Kubernetes Application Programming Interface (API) like kubectl, but also third party libraries such as the python Kubernetes client library [27].

KWOK simulates the lifetime of pods and nodes using stages. Each stage represents a part of the life cycle, with pods having stages for being *pending*, *running*, *completed* to name a few. These stages are defined in YAML as a Kubernetes resource, primarily consisting of a *selector* and a *next* field. The selector contains conditions that need to be met for the stage to take effect. Changes are then applied based upon what is specified in *next*. Stages can vary in complexity, with KWOK providing a few default options. Simulating pod failure is also possible, with the possibility of giving stages a chance to apply, if multiple meet the conditions.

Additionally, it is possible to set a delay from when a selector matches, to when the changes in *next* are applied. To make it

possible for us to simulate an existing work trace, we have added a delay between the default *running* and *succeeded* stages. The delay length is read from an annotation on each specific pod, which allows us at pod creation to define how long we want it to run.

F. System Architecture

To grant a better understanding and overview of our architecture, the following subsections will explain the design choices supported by Figure 3 which visualizes the overall architecture.

1) Kubernetes Plugin: The plugin is primarily responsible for assigning the simulated jobs to the simulated nodes. It utilizes two of the available extension points. The first one is *Score*, which is where nodes are ranked based upon the state of the microgrid they are a part of. This information is accessed through the Flask API. The second extension point is *PostBind*. This is the very last endpoint in the scheduling cycle, and is called after a pod has been successfully bound to a node. This is purely an informational endpoint, as no more decisions can be made. It is used to inform the Flask API that a pod has been assigned to a node, which should result in a new entry into the database. Algorithm 1 highlights the pseudocode for our plugin. On line 1, the prerequisites for scoring are defined. Line 3-5 gets the information about all microgrids G, the location of the candidate node N and finally the information about the microgrid (G_l) at the location (l) of the node (N). Line 7 calculates the relative difference (R_{diff}) in renewable energy production and energy consumption of a microgrid (G_l). Line 9 calculates a score via a sigmoid function, which clamps the result between $0 - 100$, due to Kubernetes needing a score in this range. Line 11 is where the weight (w) is defined and line 12 is where the final score is calculated and returned.

Algorithm 1 Pseudocode for Scheduling Framework Plugin

1: Given: pod P to schedule, candidate node N
2: **function** SCORE(N, P)
3: $G \leftarrow$ API Call
4: $l \leftarrow N$.labels["location"]
5: $G_l \leftarrow G[l]$
6:
7: $R_{diff} \leftarrow \frac{G_l.\text{renewable} - G_l.\text{load}}{G_l.\text{load}}$
8:
9: $load\_score \leftarrow \frac{100}{1.0 + e^{-5 \cdot R_{diff}}}$
10:
11: $w \leftarrow 0.5$ ▷ Weight can be set between 0-1
12: $score \leftarrow load\_score \cdot w + G_l.\text{battery} \cdot (1 - w)$
return $score$
13: **end function**

2) API & Database: To create and maintain an information link between our scheduler and the microgrid simulation, we have created a Flask [28] API. As a result, whenever a workload gets scheduled to the appropriate simulated node in KWOK, the workload is also scheduled on the appropriate simulated node in the microgrid. This is due to our API, which is responsible for the identical representation of nodes in both KWOK and the microgrid. Additionally, the API always expose the current state of all microgrids, which can be retrieved and used by the scheduling process. To store the scheduled workloads, we utilize a SQLite [29] database, where each scheduled job is stored.

3) Worktrace Runner: The worktrace runner is a python script that creates jobs through the kubeapi-server using the Kubernetes client library [27]. It contains a function that starts a simulated job given a name, a running duration, and resource requirements for the CPU and memory. Combining this with a parser for a given worktrace makes it possible to emulate a cluster doing real work.

4) Microgrid Simulation: Each simulated microgrid is located in each region in Figure 1, to accommodate the renewable energy data that is used. As such, we have a total of 152 simulated microgrids containing a data center with 6 nodes, where each simulated microgrid is configured with the components and their amount in Table I based on the python-microgrid framework.

In total, there are 912 simulated nodes (each with an 16-core processor and 128 GB of RDIMM) distributed across the 152 simulated microgrids, resulting in 6 simulated nodes in each data center. The technical specification of the nodes can be found in Appendix C. The simulation structure can be defined as the following.

*a) **Simulation Initialization**:* Load the renewable energy data set, and initialize the different modules from the library which each microgrid will be configured with. Then all microgrids are created for each region.

*b) **Simulation Run**:* Happens in time steps, where the time between each step is configurable. The simulation per microgrid can be divided into 5 parts for each time step.

1) *Retrieve the scheduled jobs, if any, from the database.*
2) *Run the jobs in the appropriate node in the microgrid for the specified completion time and consume the electricity*

Table I
MICROGRID COMPONENT OVERVIEW

Component	Description
Microgrid Location	The ISO 3166 A-2 [30] code for the country and region number.
Battery Module (1#)	The configured battery module.
Node Module (6#)	The nodes which will run scheduled jobs and consume energy.
Renewable Energy Module (1#)	The renewable energy source which will generate power to the microgrid and charge the battery.
Grid Connection (1#)	The main grid which powers the microgrid and charge the battery whenever the renewable production is insufficient.

in the microgrid accordingly.

3) *Expose the current status of the microgrid through an API which will be used by the scheduler for the future workloads.*
4) *Log the current and previous states of the microgrid to a CSV-file.*
5) *Go to the next time step and repeat the process indefinitely.*

c) ***Simulation End****:* This occurs whenever the data sets used in the simulation reaches their end, is manually stopped or a critical error crashes the simulation. In all cases the simulation should be restarted.

The detailed specification for the simulated microgrid can be found in Appendix A.

IV. Experiments

In the following section we will cover the experiments conducted based on our Section II Preliminaries and Section III Methodology. More concretely, we will first cover the worktrace which was used for all experiments in Subsection IV-A.

A. Azure Worktrace

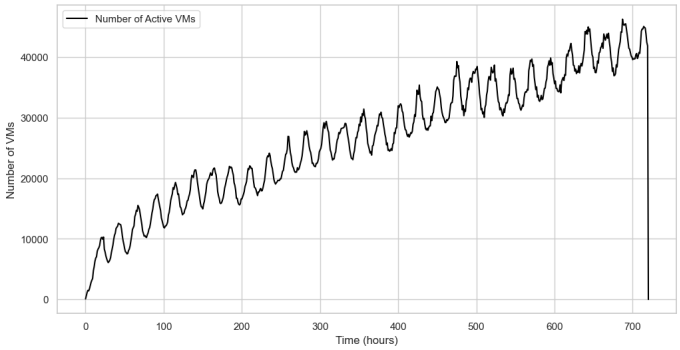
Azure provides two sanitized worktraces recorded in 2017 and 2019 of Virtual Machine (VM) workloads at one geographical region. They are accompanied by a paper that explores the possibility of predicting characteristics of VMs to better utilize resources [25].

The worktrace is useful to us because of its size, but also the amount of relevant data it contains. It includes timestamps for the start and end, memory- and CPU requirements, and information about the average and peak CPU utilization. By aggregating the average CPU utilization for all jobs running on a simulated node, combined with energy consumption statistics for appropriate hardware, we hope to be able to make a rough but feasible estimation on each nodes energy consumption. This amount is then fed to the microgrid simulator to update its state. Figure 4 highlights some of the important characteristics of the worktrace which we are interested in. Figure 4a displays the number of active VMs over time in the worktrace, Figure 4b displays the distribution of VM lifetimes in minutes, with a larger number of them having a relatively short lifetime and finally Figure 4c is the average CPU utilization of VMs over time in the worktrace.

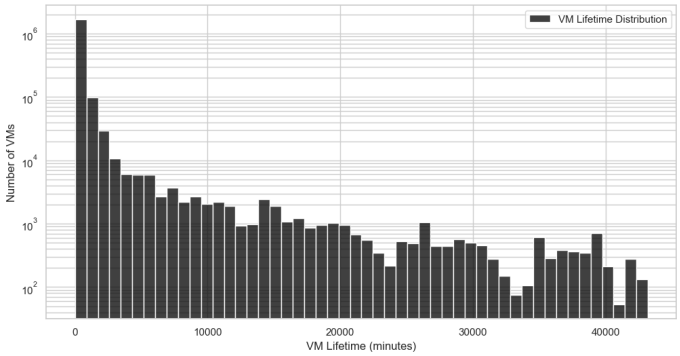
The paper includes information about a simulated run they performed to evaluate their predictions. It states that they ran 336k VM arrivals in a cluster of 880 servers with 16 cores and 112 GBs of RAM over a period of 1 month. By modeling our experiment setup based on this, we hope to achieve a realistic hardware-to-workload ratio.

B. Running The Experiment

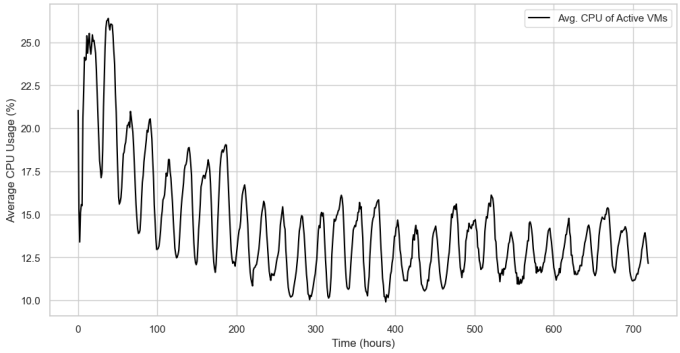
The experiment contains two weights that can be configured to affect the simulation behavior. The first of these is the weight of the scheduler plugin, compared to the other default plugins. This decides how important the result of each plugin is in the scheduling decision, by multiplying their score with the weight. For our experiment run, this value was set to 10, giving our plugin a big influence on the decision. The second weight is



(a) The number of VMs over time.



(b) Distribution of VM lifetimes.



(c) Average CPU utilization of VMs.

Figure 4. Azure VM Worktrace Characteristics

responsible for balancing the relation between renewable energy and the batteries State of Charge (SOC) when scoring. This can be seen in Algorithm 1 on line 11 and 12 as w. Like the example, the weight is set to 0.5 in the actual simulation run.

In the accompanying paper to the worktrace, the amount of jobs is scaled down to 20%, while keeping the same distribution of jobs. This is then run on 880 nodes. In our experiment the worktrace is similarly scaled, but we instead use 912 nodes to evenly distribute them across the 152 microgrids having six nodes each.

It is possible to change the speed of the experiment by changing the speed of both the microgrid simulator and the worktrace runner, in addition to proportionally changing the running time of the jobs. This enables simulating multiple days in a single day, while still staying true to the work trace, with the

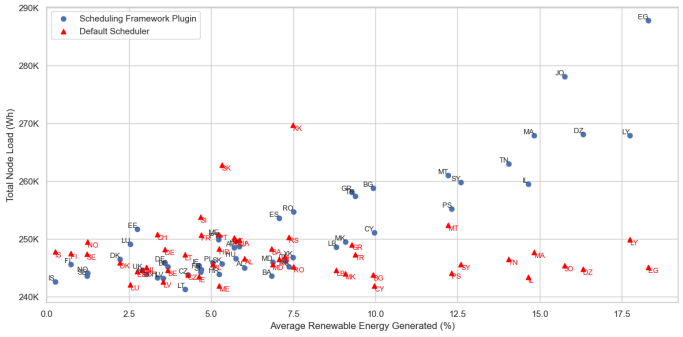


Figure 5. Total Node Load and Average Renewable Generation for Country Codes.

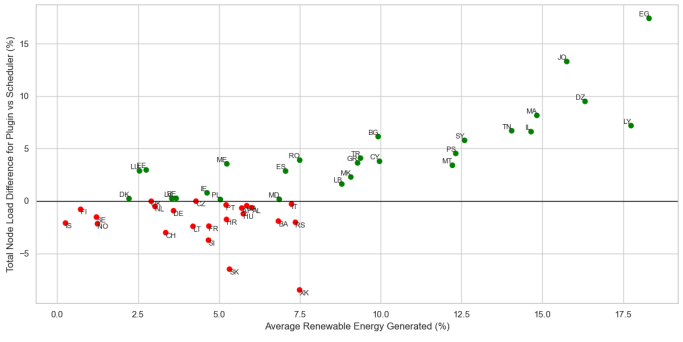


Figure 6. Total Node Load Difference between Plugin and Scheduler.

resources of the host PC and the speed of Kubernetes control plane components being the limiting factor. The experiment was run on a MacBook with an M1 Max CPU and 48 GB of RAM available to Docker with a speed of 30x real time. The simulation ran for 11.2 hours, which results in 336 simulated hours or two weeks.

C. Experiment Results

In our experiments we have evaluated our proposed Kubernetes plugin using the scheduling framework against the default Kubernetes scheduler running the same worktrace.

1) State of a Microgrid: Figure 8 shows the state of a microgrid during the simulated two weeks. Each bump in renewable energy production matches the beginning of a new day. Everything is denoted in Watts except for the battery, which is a percentage between $0 - 100$ indicating its SOC.

2) Renewable Generation Utilization: To understand the relationship between our nodes load and the renewable generation of our microgrids, we have plotted some of the results of our experiments. Figure 5 shows that our plugin prioritizes the renewable energy production of a microgrid in a region when scheduling a job, as we covered in Section III-F1. In Figure 6 we can see the percent difference of jobs scheduled for each country code, which for countries with greater renewable generation is between 0.14% to 17.42% and for the countries with a lower renewable generation is between -8.49% to -0.02%

Likewise in Figure 7, it is shown that using our plugin the total electricity export to the grid during the experiment was $1.081.619Wh$ compared to the default scheduler which was $1.357.732Wh$ with a difference of $276.113Wh$. As a result, we are utilizing 20.34% more renewable energy in comparison to the default Kubernetes scheduler.

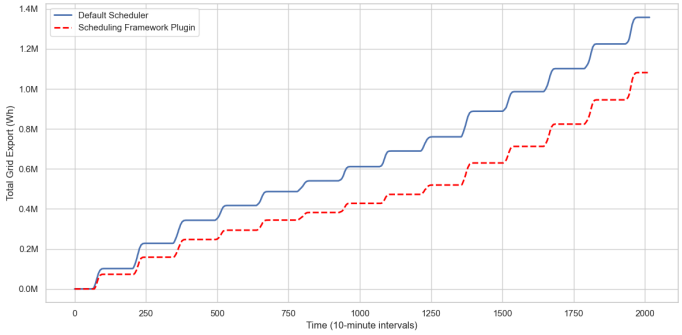


Figure 7. Grid Export for Plugin and Default Scheduler

V. Discussion

When scheduling a job, the scheduler has currently no information about the job length or what the resource utilization average might be. For the scheduler there is no difference between a quick 10 second job, and a job that might run for multiple days with a high CPU utilization. This could be one of the reasons why some countries stick out in Figure 5 If a job is scheduled at night, only the battery has influence on the score, which allows big multi-day jobs to end up on microgrids with low amounts of renewable energy.

As Figure 7 highlights, we see the improvements appear in periodic bumps, followed by a flat line. This is because the only renewable energy source in our experiment is PV, which can only produce power during the day. This gives the scheduler a limited time frame in which the microgrids are significantly different, which could be extended by adding alternative sources of renewable energy. This can also be observed in Figure 8, where the node load spikes when available renewable energy is high. This increase in load is caused by the additional jobs being scheduled to nodes in that grid. The opposite can be seen on days when significantly less solar energy is available. On those days, other microgrids with better PV conditions are being prioritized.

It is worth mentioning that these improvements to renewable energy usage are only possible because of an abundance of nodes, which allows some of them to be idle in days with low renewable energy production. However, these idle nodes still use power, so finding a balance between two should be explored in the future.

VI. Related Works

Rao et al. [31], propose an energy-aware scheduling algorithm to reduce the energy consumption in a Kubernetes cluster. The goal of their study is to create an algorithm that is able to reduce the energy consumed by considering energy consumptions of pods, CPU, communication latency and PUE of nodes while adhering to the Service Level Agreement (SLA). Their experimental results using said algorithm are the

Figure 8. Detailed State of TR10 Microgrid

reduction of energy consumption by at least 5% compared to the default scheduling algorithm. The approach they utilize is different from ours, due to their approach containing only two geographically different data centers with a total of 6 nodes. We focus on the utilization of renewable energy and the carbon reductions thereof, while having a scalable and modular microgrid and data center simulation, with 152 data centers located in microgrids and disregard communication latency and the PUE of data centers.

In a similar article *Kaur et al.* [32], propose an energy efficient Kubernetes scheduler called KEIDS. The goal of their study is to define the problem of energy efficiency as a Multi-Objective Optimization Problem (MOOP), which ensures maximum usage of green energy and overall energy minimization for a minimal carbon footprint. By evaluating KEIDS against existing Kubernetes schedulers with a real-time Google worktrace, they achieve 14.42% improved energy usage and a carbon footprint rate reduction of 47%. They rewrite the default Kubernetes scheduler to solve said problem, which differs from our contribution, though it shares a similar goal of maximizing the usage of renewable energy and reducing carbon emissions.They utilize four geographically scattered clusters, each with a renewable energy source. However, they do not consider energy storage such as batteries, to save renewable energy when it cannot be immediately consumed. Even with a similar goals, our approach is more resilient to changes in the default Kubernetes scheduler, since the scheduling framework is *out-of-tree*, whereas KEIDS would need to introduce every future change into their own scheduler.

A carbon emission-aware job scheduler for Kubernetes was proposed by *Piontek et al.* [33]. Both the default scheduler and their proposed solution was evaluated with real-world workloads with different CPU utilization and CO_2 emissions. By shifting non-critical jobs in time in combination with an algorithm which predicts future CO_2 emissions, they were able to decrease emissions by an average of 1.3% in comparison to the default scheduler while still adhering to the SLA for all jobs. Their approach to decrease carbon emissions are fundamentally different from ours. We utilize the scheduler framework, which is different from using the scheduler extender solution as it relies solely on REST. With a similar goal of reducing the carbon emission from the grid, we focus on scheduling jobs to get the best utilization of renewable energy, rather than time shifting jobs.

Hanafy et al. [34] propose a greedy algorithm implemented in Kubernetes in which resource allocation for scheduled jobs are dynamically changed based on the carbon cost of the grid. By doing so they take advantage of the temporal elasticity of batch jobs which in turn results in carbon savings. The algorithm is evaluated using a real-world machine learning training and shows carbon savings of 51% by temporal shifting compared to a carbon-agnostic execution which will run the job instantly and 37% over a suspend-resume policy. They utilize the elasticity of jobs like the previous study, and temporally shifts jobs and allocate more/less resources based on the state of the grid. This is a different approach from us, due to the fact that our approach does not spatially or temporally shift jobs after they have been scheduled, but rather schedule a job based on the best fitting node with our prioritized requirements. Another large difference in our studies, is the fact that we focus on utilizing local renewable energy production and storage to reduce the carbon intensity of the computation while they are solely relying on

the grid and the carbon intensity thereof.

Researchers have explored ways to reduce both the carbon emission and electricity cost in data centers. In an article by *Dou et al.* [35], they introduce an optimization problem which is solved by a workload scheduling algorithm called CECM with no future knowledge of the system state. CECM compromises between electricity cost and the performance of delay tolerant workloads. By evaluating their algorithm with real-life workload traces, they were able to reduce electricity costs of 9.26% and increase the usage of local renewable energy sources in the data center. Compared to our study, their goal is to reduce electricity costs in the data center by utilizing local renewable energy sources and temporal shifting delay tolerant workloads with the side-effect of reducing carbon emissions. This is similar to our study, where we also utilize local renewable energy but our primary focus is reducing carbon emissions where reducing electricity cost is a side-effect.

VII. Conclusion

We have managed to extend our preliminary research [5] by utilizing and extending the python-microgrid framework to up scale the amount of microgrid locations, implemented a node module for scheduling and employ realistic weather data. Our proposed Kubernetes plugin using the scheduling framework has successfully used the renewable production and battery state in its scheduling of workloads to appropriately select the most fitting microgrid. Our proposed contribution has been evaluated against the default Kubernetes scheduler with a real worktrace from Azure and have managed to achieve an increase in renewable energy utilization of 20.34%. At last, we list some of the future research directions which have not been studied in this paper and could be interesting to explore.

VIII. Future Works

In the following section we will cover some of the fields of research which was not covered in this paper and thus can be further explored.

A. Introducing Carbon Intensity

At the moment, the scheduler aims to maximize the amount of renewable energy consumed by nodes by prioritizing nodes in microgrids with high current renewable energy production. Because the only renewable source in our experiment is PVs, some countries with known low carbon intensity are being down prioritized, as illustrated by the case of Norway (NO) in Figure 6, as solar power may not be optimally suited for their geographical location. By exposing the real time carbon intensity of the main grid connected to each microgrid through the API, the scheduling algorithm could use it in its decision making. This could especially make a difference during the night where PVs is unavailable, unless the geographical space is expanded to include vastly different timezones.

B. Mixed Renewable Generation

A noteworthy topic to research further, is the effect of introducing different renewable energy sources such as wind or nuclear. Due to PVs being more volatile, as shown in the data in Figure 2, we see that there are longer periods with no renewable generation available. In our case, this meant that we could employ the use of electrical battery storage, which could be utilized whenever the production was low. As such, a less volatile renewable source in terms of production such as a small nuclear reactor, could provide the reduction in carbon emissions and reduce the energy drawn from the main grid for more environmentally friendly data centers. Another alternative which would be interesting to explore, could be combining one or more renewable energy sources in a single data center, for instance both utilizing PVs and wind turbines if the location supports it.

C. Different Workload Types

In our proposed solution, we assume a homogeneous workload type and server configuration, which is utilizing the CPU of the server. By introducing servers in the data centers with different hardware specifications, to handle workloads such as memory or GPU intensive, the scheduler could take these into consideration when scheduling different workloads. GPU intensive workload types such as AI training, which can utilize time shifting or suspend-resume techniques could exploit the periods in renewable energy production when the servers would otherwise need to consume power from the battery or the main grid. Thus the impact of not having homogeneous servers, and the potential difference in power consumption could be valuable when planning renewable energy sources which the data center could consume energy from.

D. Different Energy Storage Sizing

As we just discussed with our server configuration in Subsection VIII-C, our contribution also assumes a homogeneous microgrid configuration. This meant that for the microgrid locations where the utilization value of the PVs was low, the battery was over sized and thus is not able to be fully charged given a lack of surplus energy. To combat this, it could be interesting to look at the regions in Figure 1 with a utilization value under 0.14 and downscale the capacity of the configured batteries. Likewise, it could be interesting to research whether or not it is possible to achieve even greater reductions in carbon emissions and energy drawn from the main grid, by scaling up the batteries capacity in the regions with a utilization value over 0.14.

E. Optimizing for Price

As a result of our scheduler scoring algorithm, its primary goal is to minimize the amount of energy exported to the grid. At the moment, all energy exported is considered equally ”bad” in our experiment setup, but that might not reflect a real life scenario. In practice, the current energy market price may be varying at different microgrids, and could be taken into account by extending the scoring algorithm, by for example down-prioritizing nodes where it makes financial sense to sell back to the grid at that moment. By adding this aspect to the simulation, one could explore if, and how much money could be saved.

F. Simulation Performance

As mentioned in Subsection IV-A, the Azure worktrace contains data for an entire month. We did however run into performance issues around the 20 day mark, with the KWOK cluster struggling to keep up with the rate that jobs were being scheduled. By observing the Docker resource monitor, it is clear to see that our setup reaches the threshold of both CPU and memory after prolonged running time. Running the simulation on better hardware might allow for bigger and longer simulations, which could potentially give more realistic results. it might also be possible to optimize parts of the scheduler to run more efficiently, saving the resources available.

G. Exploring Permutations

We have simulated and evaluated the result of a single experiment setup based upon the Azure worktrace and accompanying paper [25]. In the future, it would be interesting to explore the effects of changes in our setup. What would be the result of having double the amounts of available nodes, or only half? Same goes for the size of the batteries, or the renewable sources at the microgrids. Additionally, the value of the weight in the scoring as seen on line 11-12 in Algorithm 1 could be tweaked, to explore if a better default value exists.

Acknowledgment

We would like to thank **Michele Albano** who helped us as our shepherd and supervisor during the project. Additionally, a big thanks to **Hessam Golmohamadi**, for proof-reading and giving feedback on the specification and understanding of microgrids and other energy-related questions we encountered.

References

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aba3758

[2] A. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges (Basel)*, vol. 6, no. 1, pp. 117–157, 2015.

[3] W. A. Hanafy, Q. Liang, N. Bashir, A. Souza, D. Irwin, and P. Shenoy, "Going green for less green: Optimizing the cost of reducing cloud carbon emissions," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 479–496. [Online]. Available: https://doi.org/10.1145/3620666.3651374

[4] S. Mizani and A. Yazdani, "Optimal design and operation of a grid-connected microgrid," in *2009 IEEE Electrical Power '& Energy Conference (EPEC)*, 2009, pp. 1–6.

[5] L. C. B. Mumberg and S. M. P. Andersen, "Towards renewable kubernetes scheduling for microgrids using custom plugin," 2025.

[6] G. Henri, T. Levent, A. Halev, R. Alami, and P. Cordier, "pymgrid: An open-source python microgrid simulator for applied artificial intelligence research," 2020. [Online]. Available: https://arxiv.org/abs/2011.08004

[7] "ahalev/python-microgrid: python-microgrid is a python library to generate and simulate a large number of microgrids." [Online; accessed 2025-04-11]. [Online]. Available: https://github.com/ahalev/python-microgrid

[8] D. Ton and M. Smith, "The u.s. department of energy's microgrid initiative," *The Electricity Journal*, vol. 25, p. 84–94, 10 2012.

[9] A. G. Amanda McGrath, "What is a microgrid? — ibm," [Online; accessed 2024-12-12]. [Online]. Available: https://www.ibm.com/topics/microgrid

[10] D. o. E. United States, "Microgrid overview — grid deployment office," January 2024, [Online; accessed 2024-12-12]. [Online]. Available: https://www.energy.gov/sites/default/files/2024-02/46060_DOE_GDO_Microgrid_Overview_Fact_Sheet_RELEASE_508.pdf

[11] "Announcing the stargate project — openai," [Online; accessed 2025-05-05]. [Online]. Available: https://openai.com/index/announcing-the-stargate-project/

[12] "Kepler — cncf," 5 2025, [Online; accessed 2025-05-26]. [Online]. Available: https://www.cncf.io/projects/kepler/

[13] D. T. US, "Dell technologies — enterprise infrastructure planning tool," [Online; accessed 2025-05-05]. [Online]. Available: https://www.dell.com/calc

[14] "Kubernetes," [Online; accessed 2025-04-07]. [Online]. Available: https://kubernetes.io/

[15] "Who we are — cncf," [Online; accessed 2025-05-05]. [Online]. Available: https://www.cncf.io/about/who-we-are/

[16] "Why you need kubernetes and what it can do — kubernetes," [Online; accessed 2024-12-12]. [Online]. Available: https://kubernetes.io/docs/concepts/overview/#why-you-need-kubernetes-and-what-can-it-do

[17] "Kubernetes scheduler — kubernetes," [Online; accessed 2024-12-23]. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

[18] "Kepler — cncf," 12 2023, [Online; accessed 2025-05-29]. [Online]. Available: https://kubernetes.io/docs/concepts/extend-kubernetes/#scheduling-extensions

[19] "Scheduling framework — kubernetes," [Online; accessed 2024-12-23]. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

[20] "Homer - hybrid renewable and distributed generation system design software," [Online; accessed 2025-04-11]. [Online]. Available: https://homerenergy.com/products/pro/index.html

[21] J. P. Murcia, M. J. Koivisto, G. Luzia, B. T. Olsen, A. N. Hahmann, P. E. Sørensen, and M. Als, "Validation of european-scale simulated wind speed and wind generation time series," *Applied Energy*, vol. 305, p. 117794, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261921011296

[22] M. J. Koivisto and J. P. M. Leon, "Pan-european wind and solar generation time series (pecd 2021 update)," [Online; accessed 2025-04-22]. [Online]. Available: https://data.dtu.dk/collections/Pan-European_wind_and_solar_generation_time_series_PECD_2021_update_/5939581

[23] ——, "Solar pv generation time series (pecd 2021 update)," 5 2022, [Online; accessed 2025-04-22]. [Online]. Available: https://data.dtu.dk/articles/dataset/Solar_PV_generation_time_series_PECD_2021_update_/19727239

[24] A. Ogunjuyigbe, T. Ayodele, and O. Akinola, "Optimal allocation and sizing of pv/wind/split-diesel/battery hybrid energy system for minimizing life cycle cost, carbon emission and dump energy of remote residential building," *Applied Energy*, vol. 171, pp. 153–171, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261916303713

[25] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 153–167. [Online]. Available: https://doi.org/10.1145/3132747.3132772

[26] K. SIG, "Kubernetes with out kubelet," 6 2024, [Online; accessed 2025-05-29]. [Online]. Available: https://kwok.sigs.k8s.io/

[27] "Python client library," [Online; accessed 2025-05-29]. [Online]. Available: https://github.com/kubernetes-client/python

[28] "Welcome to flask — flask documentation (3.1.x)," [Online; accessed 2025-05-22]. [Online]. Available: https://flask.palletsprojects.com/en/stable/

[29] "Sqlite home page," [Online; accessed 2025-05-22]. [Online]. Available: https://www.sqlite.org/

[30] C. to Wikimedia projects, "List of iso 3166 country codes - wikipedia," 9 2018, [Online; accessed 2025-04-25]. [Online]. Available: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes

[31] W. Rao and H. Li, "Energy-aware scheduling algorithm for microservices in kubernetes clouds," *Journal of Grid Computing*, vol. 23, 2024. [Online]. Available: https://link-springer-com.zorac.aub.aau.dk/article/10.1007/s10723-024-09788-w

[32] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and M. Atiquzzaman, "Keids: Kubernetes-based energy and interference driven scheduler for

industrial iot in edge-cloud ecosystem," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4228–4237, May 2020.
[33] T. Piontek, K. Haghshenas, and M. Aiello, "Carbon emission-aware job scheduling for kubernetes deployments," *The Journal of supercomputing*, vol. 80, no. 1, pp. 549–569, 2024.
[34] W. A. Hanafy, Q. Liang, N. Bashir, D. Irwin, and P. Shenoy, "Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency," *Proceedings of the ACM on measurement and analysis of computing systems*, vol. 7, no. 3, pp. 1–28, 2023.
[35] H. Dou, Y. Qi, W. Wei, and H. Song, "Carbon-aware electricity cost minimization for sustainable data centers," *IEEE transactions on sustainable computing*, vol. 2, no. 2, pp. 211–223, 2017.
[36] "Fully managed container solution – amazon elastic container service (amazon ecs) - amazon web services," [Online; accessed 2025-04-07]. [Online]. Available: https://aws.amazon.com/ecs/
[37] "Cloud run — google cloud," [Online; accessed 2025-04-07]. [Online]. Available: https://cloud.google.com/run
[38] S. H. Valerie Silverthorne, "Cloud native 2024: Approaching a decade of code, cloud, and change," The Linux Foundation, March 2025. [Online]. Available: https://www.cncf.io/reports/cncf-annual-survey-2024/
[39] M. S. Mahmoud, "Microgrid control problems and related issues," in *Microgrid*. United Kingdom: Elsevier Science & Technology, 2016, pp. 1–42.
[40] "Extending kubernetes — kubernetes," citehttps://kubernetes.io/docs/concepts/extend-kubernetes/, (Accessed on 10/01/2025).
[41] "Scheduling framework — kubernetes," [Online; accessed 2024-12-23]. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/#interfaces
[42] "Scheduling plugins — kubernetes," https://kubernetes.io/docs/reference/scheduling/config/#scheduling-plugins, (Accessed on 10/01/2025).
[43] "pymgrid.modules.batterymodule — pymgrid 1.4.1 documentation," [Online; accessed 2025-04-30]. [Online]. Available: https://python-microgrid.readthedocs.io/en/latest/reference/api/modules/pymgrid.modules.BatteryModule.html
[44] "pymgrid.modules.renewablemodule — pymgrid 1.4.1 documentation," [Online; accessed 2025-04-30]. [Online]. Available: https://python-microgrid.readthedocs.io/en/latest/reference/api/modules/pymgrid.modules.RenewableModule.html
[45] "pymgrid.modules.gridmodule — pymgrid 1.4.1 documentation," [Online; accessed 2025-04-30]. [Online]. Available: https://python-microgrid.readthedocs.io/en/latest/reference/api/modules/pymgrid.modules.GridModule.html
[46] T. Sukprasert, A. Souza, N. Bashir, D. Irwin, and P. Shenoy, "On the limitations of carbon-aware temporal and spatial workload shifting in the cloud," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 924–941. [Online]. Available: https://doi.org/10.1145/3627703.3650079

APPENDIX A
PYTHON MICROGRID CONFIGURATION

As can be seen in our Github repository, each component in the microgrid is configured with the following settings in the code. The battery module, renewable energy module, grid connection module are all default implementations of the library components. The node module is our custom configuration of the load module, such that it is able to act as a node and dynamically shift the consumption rate based on the scheduled jobs.

Table II
BATTERY MODULE CONFIGURATION

Battery Module	Description (From python-microgrid documentation [43].)	Configured Value
min_capacity	Minimum energy that must be contained in the battery.	0 Wh
max_capacity	Maximum energy that can be contained in the battery. If soc=1, capacity is at this maximum.	23296.7 Wh
max_charge	Maximum amount the battery can be charged in one step.	2329.67 Wh
max_discharge	Maximum amount the battery can be discharged in one step.	2329.67 Wh
efficiency	Efficiency of the battery.	90 %
init_soc	Initial state of charge of the battery.	50 %

Table III
NODE MODULE CONFIGURATION

Node Module	Description	Configured Value
Default Power Consumption	The default power consumption for each node when no workload is running.	120 watts

Table IV
RENEWABLE ENERGY CONFIGURATION

Renewable Energy Module	Description (From python-microgrid documentation [44].)	Configured Value
times_series	Time series of renewable production, which is turned from 1-hour intervals to 10-minute intervals.	solarPV_10min.csv
final_step	Length of time_series data.	solarPV_10min.csv

Table V
GRID CONNECTION CONFIGURATION

Grid Connection Module	Description (From python-microgrid documentation [45].)	Configured Value
max_import	Maximum import at any time step.	100000 Wh
max_export	Maximum export at any time step.	100000 Wh
time_series	Time series data of import price, export price and co2 per kWh, converted to Wh.	(import_price, export_price, co2-data.csv)

APPENDIX B
RENEWABLE ENERGY PERCENTAGE OF GRIDS ACROSS REGIONS

The following figure is based on the carbon emission data from Electricity Maps, which we discussed in Subsection II-B1 and Subsection III-B.

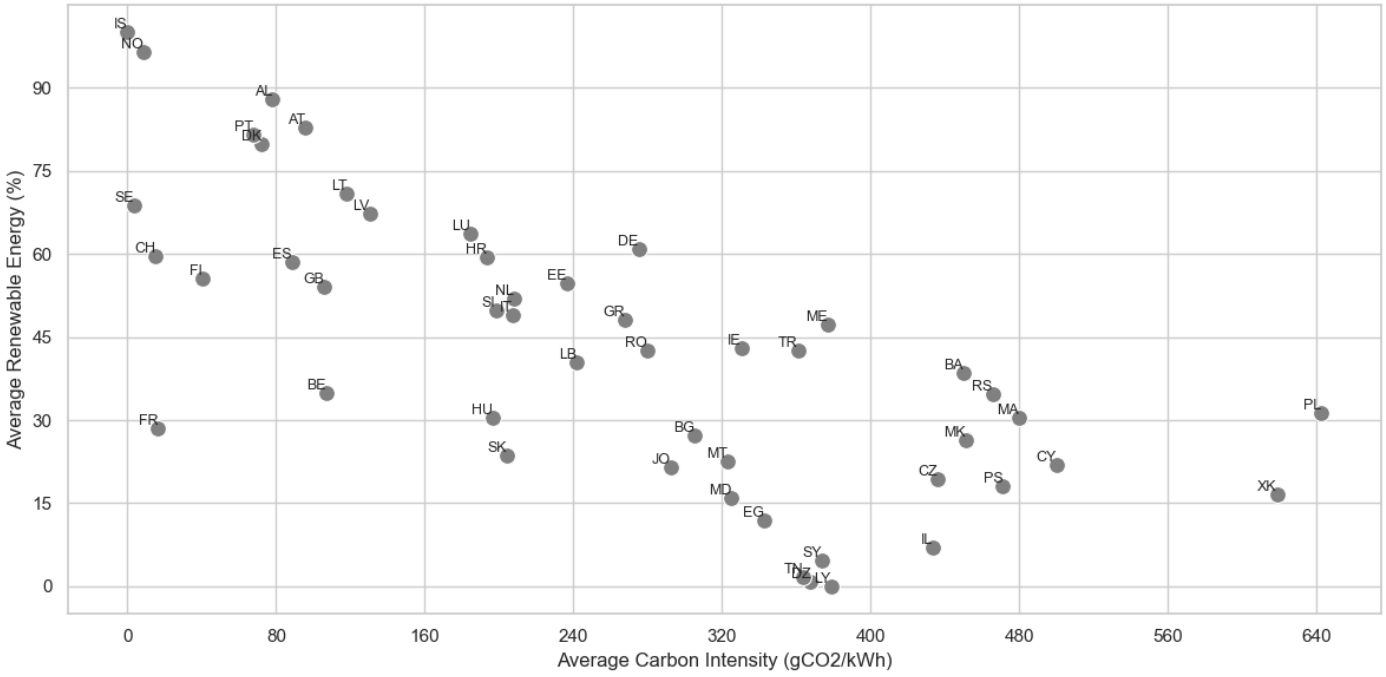


Figure 9. Renewable Energy Percentage / Carbon Intensity across all regions.

APPENDIX C
NODE TECHNICAL CONFIGURATION

In order to configure each node as realistically as possible, we utilized Dells Infrastructure Planning tool [13]. This choice was also based on the fact that the tool is able to estimate the power consumption both at idle and with $0 - 100\%$ CPU utilization. To match the specifications of the servers in the workload trace from Azure [25], we have decided to mimic their setup in our own configuration. As a result, we had a choice between 5 or 6 nodes per data center in each microgrid, which meant having either 760 or 912 servers. We went with 6 nodes per data center, where each node has its power consumption modeled after a PowerEdge R660 Server, where the only modification in the tool is the removal of the additional processor and the addition of more RAM meaning that there is a total of 128 GB RDIMM memory instead of the default 64 GB. The exported solution in JSON can be found in the Github repository[3] under the *python-microgrid-simulation/dell-eipt* folder files and imported into the Dell EIPT.

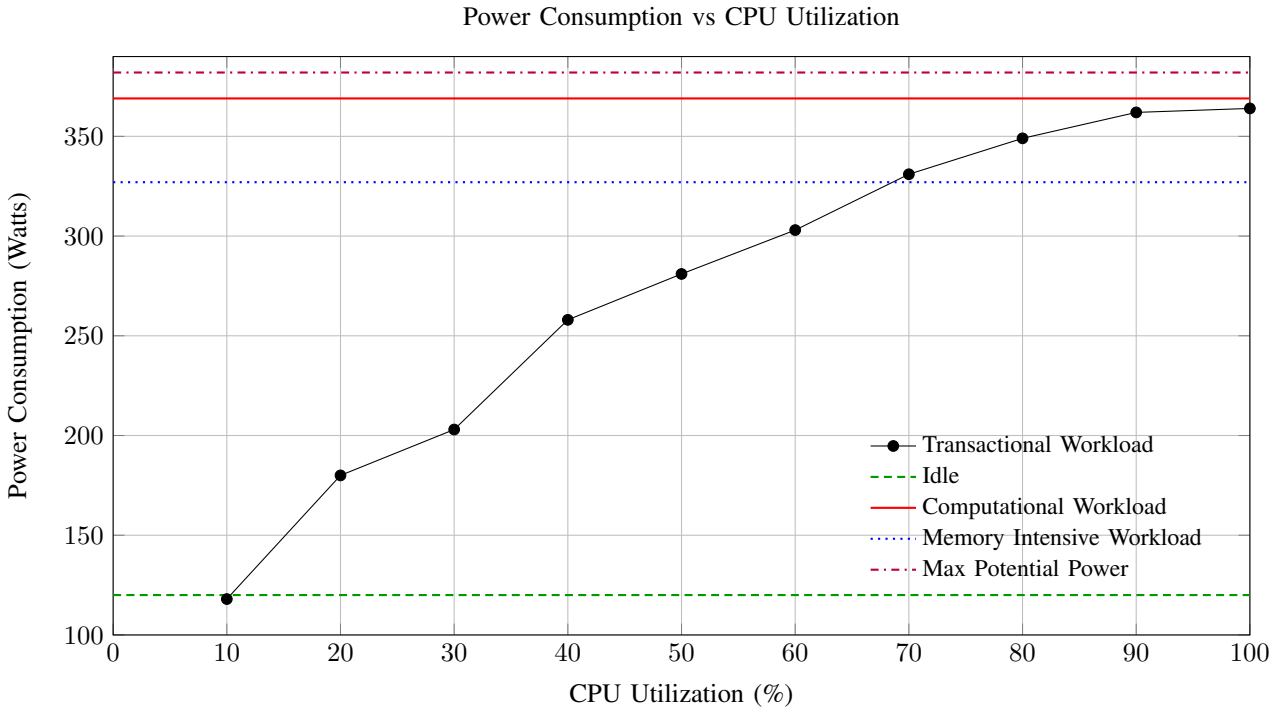


Figure 10. Power consumption based on CPU utilization for PowerEdge R660 [13].

[3]https://github.com/simonmpa/kubernetes-microgrid-research/