S.C.A.L.E
A CO2-Aware Scheduler for OpenShift at ING

Den Toonder, Jurriaan; Braakman, Paul; Durieux, Thomas

DOI
10.1145/3663529.3663862

Publication date
2024

Document Version
Final published version

Published in
FSE Companion - Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering

S.C.A.L.E: A CO2-Aware Scheduler for OpenShift at ING

Jurriaan Den Toonder
TU Delft
Delft, Netherlands
ING
Amsterdam, Netherlands
jurriaan@toonder.me

Paul Braakman
ING
Amsterdam, Netherlands
Paul.Braakman@ing.com

Thomas Durieux
TU Delft
Delft, Netherlands
thomas@durieux.me

ABSTRACT

This paper investigates the potential of reducing greenhouse gas emissions in data centers by intelligently scheduling batch processing jobs. A carbon-aware scheduler, `S.C.A.L.E` (Scheduler for Carbon-Aware Load Execution), was developed and applied to a resource-intensive data processing pipeline at ING. The scheduler optimizes the use of green energy hours, times with higher renewable energy availability, and lower carbon emissions. The `S.C.A.L.E` comprises three modules for predicting task running times, forecasting renewable energy generation and electricity grid demand, and interacting with the processing pipeline. Our evaluation shows an expected reduction in greenhouse gas emissions of around 20% when using the carbon-aware scheduler. The scheduler's effectiveness varies depending on the season and the expected arrival time of the batched input data. Despite its limitations, the scheduler demonstrates the feasibility and benefits of implementing a carbon-aware scheduler in resource-intensive processing pipeline.

CCS CONCEPTS

• **Software and its engineering** → *Grid computing*.

KEYWORDS

Climate change,Scheduling,OpenShift,Greenhouse gas,Data center

ACM Reference Format:
Jurriaan Den Toonder, Paul Braakman, and Thomas Durieux. 2024. S.C.A.L.E: A CO2-Aware Scheduler for OpenShift at ING. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24), July 15–19, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3663529.3663862

1 INTRODUCTION

The global climate change crisis and the associated phenomenon of global warming have taken center stage in recent years as an existential challenge facing humanity. The consequences of global warming are already noticeable. For example, recent studies show that it harms crop production, threatening food security worldwide [11, 15]. Global warming has been significantly influenced by increased greenhouse gas emissions caused by human activities [20]. This includes the carbon emissions generated by Internet services, which account for a substantial portion of the overall emissions and are forecasted to only increase over time [5, 10]. These services operate through large data centers, making data centers a considerable contributor to carbon emissions and global warming.

A study conducted in 2010 revealed that data centers already emitted as much CO_2 as the entire country of Argentina due to their high energy consumption [6]. In 2016, the world's data centers used more than 416.2 tWh [2], which was higher than the total consumption of Britain at that time of around 300 tWh. The energy consumption and associated CO_2 emissions by data centers have only grown in recent years [2, 18, 19] and are expected to rise to 974 tWh in 2030 [1]. As such, a growing interest has been in reducing carbon emissions, including from data centers [10].

In addition to their high carbon emissions, data centers are generally inefficient and underutilized [6]. Server utilization rates within data centers barely exceed 6%, while facility utilization hovers around 50%. This inefficiency increases the environmental impact of data centers and highlights the need for improvements.

Society relies on these servers, and it is unrealistic to remove them from our daily lives. In this work, we explore the possibility of mitigating the environmental impact of these servers. We leverage the fact that most servers do not need to operate continuously, allowing tasks to be scheduled throughout the day. Additionally, we consider the variable environmental impact of energy production, which changes throughout the day. For instance, energy is greener at noon when the sun is shining compared to the night when solar panels are not producing any energy.

This contribution focuses on a specific type of load in data centers: batch processing compute jobs. These compute jobs are characterized by their lack of strict requirements regarding the timing of their execution. In other words, they have a temporally flexible element, meaning their processing can be delayed to a later time of the day without causing issues for the system and without interrupting business as usual. A compute job consists of any processing step that does resource-intensive data processing. In this context, the resources refer to CPU and memory usage, which in turn have a direct correlation to electricity consumption [4, 13].

There is an opportunity to intelligently schedule computing jobs to reduce the amount of greenhouse gasses emitted, provided these jobs have temporal flexibility. It becomes possible to optimize their energy consumption and reduce carbon emissions by leveraging the flexibility of compute jobs, the variability of greenhouse gasses emitted depending on the time, and the general under-utilization of data centers. The optimization can be done by scheduling these compute jobs to times at which the carbon load of the energy grid is low. Intelligently scheduling and redistributing compute jobs can align the demand for computing resources with periods

FSE Companion '24, July 15–19, 2024, Porto de Galinhas, Brazil
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0658-5/24/07
https://doi.org/10.1145/3663529.3663862

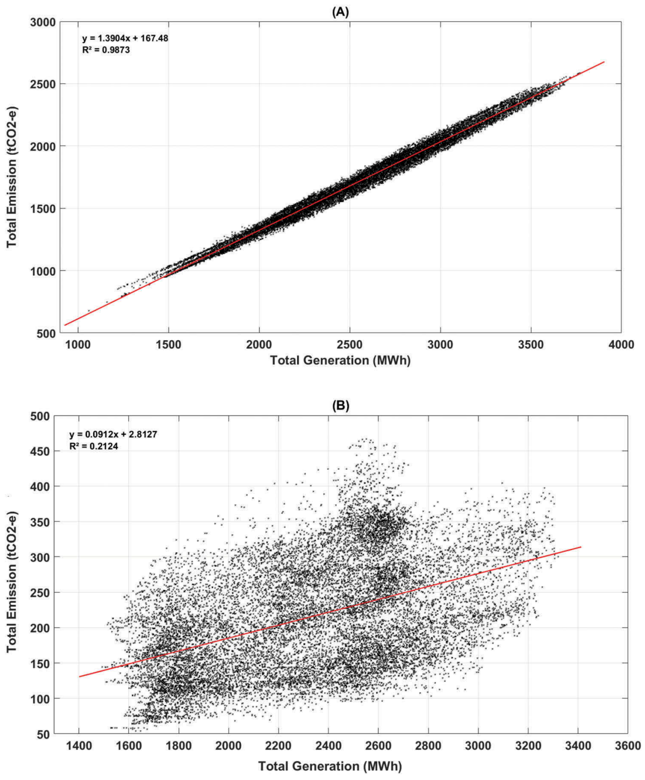


Figure 1: Correlations between generations and emissions for the electricity systems of (A) Bangladesh and (B) New Zealand in 2015 [8].

of low-carbon energy generation, thereby minimizing the overall environmental impact of data centers.

The optimization will work best in areas in which there is an abundance of green energy generation available. The variability of greenhouse gas emissions per mWh is much lower when there is little green energy generation in an area. This is shown in Figure 1 from [8], which shows two plots of greenhouse gas emissions per generated MWh from two different countries. Figure 1 B shows emissions per MWh in New Zealand, which has much more green energy generation available compared to Bangladesh shown in Figure 1 A. The optimization aims to utilize the variability of emissions per MWh as shown in Figure 1 B.

This contribution explores the possibility of creating and implementing a carbon-aware scheduler to minimize the carbon load of a real-world resource-intensive data processing pipeline at ING in the trade and communications surveillance squad. ING is a large multinational banking corporation headquartered in Amsterdam. It offers various financial services, including retail and commercial banking, and has a significant presence in many countries. The trade and communications surveillance squad manages a data processing pipeline within ING that runs on the ING data centers.

In summary, the contributions of this paper are:

- An open source implementation of `S.C.A.L.E`: a CO_2-aware Scheduler for OpenShift and Docker. [1]

[1]https://github.com/Fastjur/S.C.A.L.E./

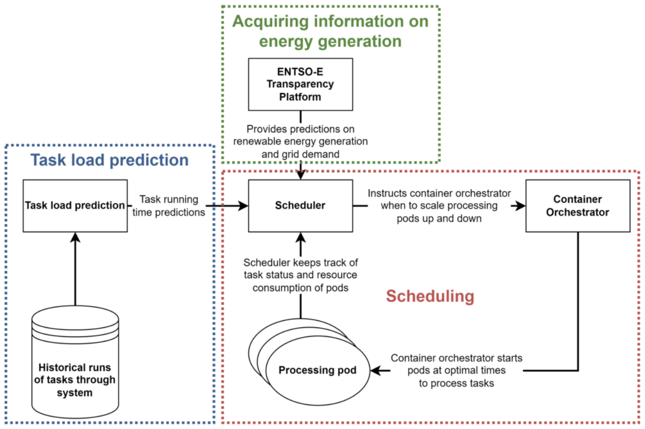

Acquiring information on energy generation
ENTSO-E Transparency Platform
Provides predictions on renewable energy generation and grid demand
Task load prediction
Task load prediction
Task running time predictions
Scheduler
Instructs container orchestrator when to scale processing pods up and down
Container Orchestrator
Scheduler keeps track of task status and resource consumption of pods
Scheduling
Historical runs of tasks through system
Processing pod
Container orchestrator starts pods at optimal times to process tasks

Figure 2: The overview of the implementation of the carbon-aware scheduler at ING.

- An empirical evaluation of `S.C.A.L.E` effectiveness on open source data.

2 `S.C.A.L.E`: CARBON-AWARE SCHEDULER

This section presents, `S.C.A.L.E`, the carbon-aware scheduler that we designed. Figure 2 shows an overview of the scheduler. In the middle, the scheduler is shown. It gathers data from various sources, such as the ENTSO-E Transparency Platform and historical runs of the system. It combines these data points to create a schedule aiming to reduce the system's carbon emissions. Once it has determined a schedule, it communicates this to the container orchestration platform, which in turn starts up pods to process at the right times. Each part of this system is described in this section

`S.C.A.L.E.` is designed to work on container application platforms such as OpenShift. OpenShift is built on top of Kubernetes and provides features and functionality for building, deploying, and managing containerized applications. It provides `S.C.A.L.E.` the ability to create a schedule and process compute jobs at optimal times, by scaling deployments for those compute jobs.

2.1 Task Load Prediction

The initial step in determining a schedule involves predicting the execution time of incoming tasks. This step is required, first, to ensure that deadline constraints are not violated when scheduling tasks, and second, to facilitate shifting of task processing to periods of low carbon intensity.

Execution time prediction is achieved by analyzing the historical data of previous task execution. We collect two types: foreknown properties of tasks that can be determined prior to scheduling or processing such as input file size and task type, and metrics related to task processing such as start and end times of tasks and used system resources. We also collect data related to the system, such as allocated (virtual-)CPUs, allocated memory, and the number of concurrent jobs allowed.

In the case of the ING system, the task type and the file size of the accompanying file are considered. From this, the scheduler calculates the processing speed in B/s for every previous task. The metric can be adapted depending on the type of task.

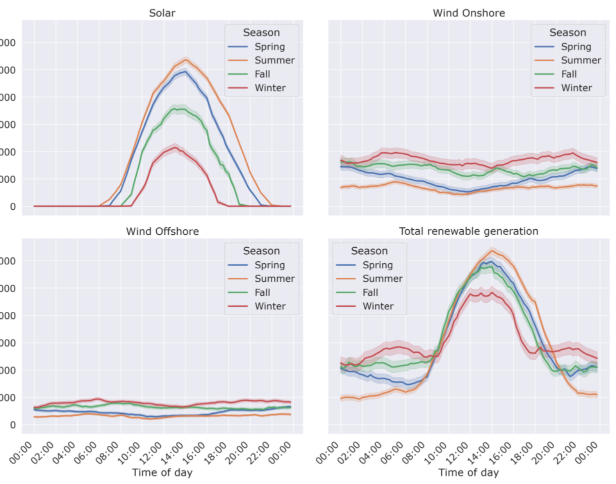


Figure 3: Renewable generation predictions per season in 2022 per time of day in The Netherlands

Based on this data, the scheduler can compute the median execution time of the previous execution based on the file size. The expected processing time of an incoming file is then calculated by dividing the task file size by the median processing speed:

$$\texttt{expected\_duration} = \frac{\texttt{task.file.file\_size}}{\texttt{median\_processing\_speed}} \quad (1)$$

A time-based sliding window approach is also used to adapt the prediction based on recent executions and discard historical data that could have been collected from an outdated system. In the ING system, a 30-day window is considered a tradeoff between the accuracy of the prediction and adaptation to system changes.

2.2 Gathering Data on Energy Generation

The second component of the CO_2-aware scheduler is to acquire information regarding energy generation and more specifically the proportion of green energy production in the next hours. The variability in renewable energy generation offers the opportunity to optimize the execution pipeline and therefore reduce the CO_2 emission. Figure 3 shows the average renewable energy generation in the Netherlands by the time of day for every season in 2022. The figure shows that, on average, the amount of renewable energy generation is highest between 12:00 and 14:00, depending on the season. Although renewable energy peaks around noon on average due to solar generation, other sources should be considered. Figure 4 displays the energy generation statistics in the Netherlands on November 13th, 2023. These graphs show that wind energy should also be taken into consideration when scheduling tasks on a day-to-day basis, as the peak production of renewable energy occurred around midnight on November 13th, 2023.

Therefore, the naive expectation of executing the tasks around noon is not sufficient. The scheduler has to dynamically adapt depending on the forecast of the day. Moreover, it is important to take into consideration the usage of the grid, i.e., scheduling tasks at 7 pm when all household are running their appliances is not optimal even if a lot of wind energy is produced. Therefore, the scheduler should take into account the peak of green energy as well as the

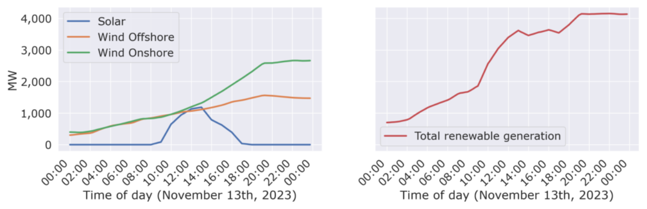


Figure 4: Renewable energy generation predictions in the Netherlands for November 13th, 2023, with an increase of wind energy. The total amount of renewable energy available does not peak during noon but at night.

expected load on the electricity grid. This combination may provide better options throughout the day, at which the total renewable generation is slightly lower, but the predicted demand is significantly lower. To that end, the scheduler calculates a "renewable energy percentage", using both the renewable energy generation predictions and the predicted demand on the electricity grid. The renewable energy percentage is the main source of information the scheduler uses for determining periods of low-carbon-intensive energy generation.

The renewable energy percentage is calculated by summing the renewable energy predictions for solar and wind and dividing them by the forecasted demand.

$$\texttt{renewable energy percentage} = \frac{\texttt{solar} + \texttt{wind}}{\texttt{grid demand}} \quad (2)$$

Figure 5 shows the renewable energy percentages over the time of day and per season. At the top left, the forecasted renewable energy generation is plotted. The top right shows the forecasted load in The Netherlands. The combination of these values is then plotted in the bottom graph. As can be seen, on average, renewable energy generation is highest around noon every day, and the forecasted demand is lowest at these times. Therefore, it is expected that the energy generation will be, on average, the cleanest around noon in terms of gCO_2-eq per generated kWh. Those figures confirm that there is an opportunity to schedule processes for optimizing carbon emissions.

`S.C.A.L.E` relies on a 'transparency platform' provided by ENTSO-E. This platform provides API access to electricity generation, transportation, and consumption information for the European market. ENTSO-E provides a forecast of energy production for the next 24 hours. `S.C.A.L.E` utilizes this information for the scheduling and adapts its schedule for the conditions of the day.

2.3 Scheduling

The tactic of the scheduler aims to predict the highest renewable energy percentage in the next 24 hours. Once it has determined the optimal time, the scheduling methodology is simple; it aims to process all tasks symmetrically around this point. In other words, tasks are batched together as one processing job to be processed symmetrically around this peak in low-carbon-intensive energy generation. To do so, it predicts a processing time for all tasks and sums these times up. Once this has been calculated, the scheduler knows the expected total running time of the tasks to be processed. Using this information, it can determine when the set of tasks should start to

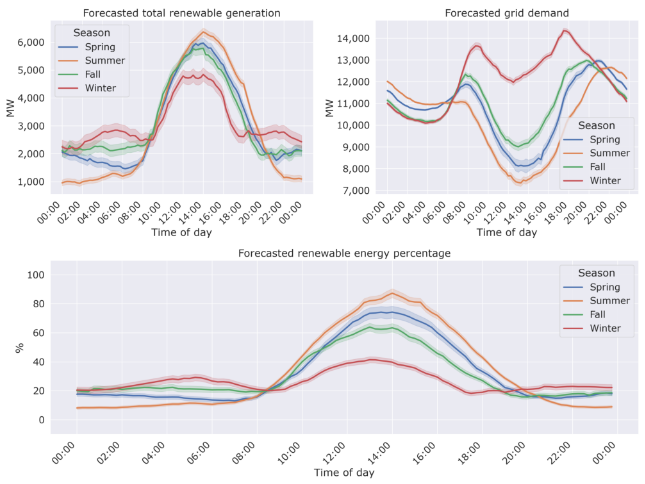


Figure 5: Average renewable energy percentage forecasts for the four seasons in 2022, based on the predicted renewable energy generation and grid demand in The Netherlands.

process to maximize the usage of low-carbon-intensive energy. The following paragraphs describe the contributions from the red box, shown in the middle and bottom right parts of Figure 2.

Additionally, the scheduler takes concurrent processing into account. A maximum concurrency setting is configurable in the system; the scheduler considers this value when predicting the total running time of a batch of tasks. The purpose of adding concurrency is to maximize the utility of low-carbon-intensive energy.

Figure 6 shows this concept visually. It plots the average renewable percentage over the time of day in summer. As can be seen, the peak in renewable energy percentage is at 14:00; the scheduler aims to process tasks symmetrically around this time. Assume for this scenario that the scheduler has predicted the total task running time to take 4 hours without any concurrency. If the scheduler would get this as input, it would start processing at 12:00, as it expects to finish around 16:00. However, if the maximum concurrency is set to 4, the scheduler expects the running time to be approximately 1 hour. Therefore, it would delay processing until 13:30, with an expected end time of around 14:30. From this figure, it should be clear that adding concurrency maximizes the utility of renewable energy generation, on average.

Concurrent processing is implemented by the scheduler using horizontal scaling. Horizontal scaling refers to a method of increasing the capacity of a system by adding more instances of the same type of resource. In this case, by adding more processing pods. Conversely, vertical scaling increases the capacity of existing resources by allocating more resources to them. Vertical scaling could be implemented in this context by allocating more CPU time or memory to one pod.

To prevent deadline incursions when scheduling files, the concept of a 'latest feasible starting time' is introduced. The scheduler calculates the 'latest feasible starting time' based on the task's predicted running time and its deadline, ensuring that every task is processed within the given deadline. When scheduling a set of new tasks, the scheduler collects all the corresponding files and puts them into a single queue. The tasks within the queue are then prioritized based on their latest feasible starting times.

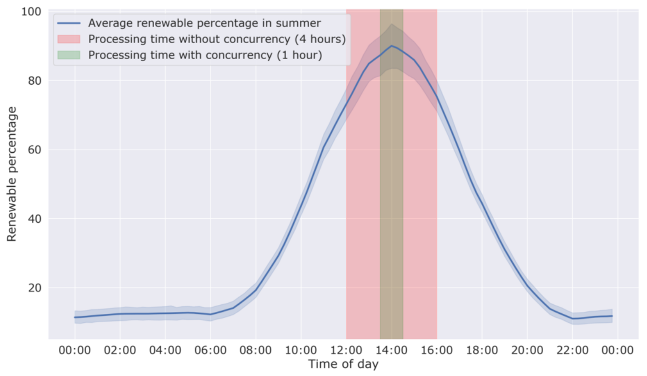


Figure 6: Example of scheduling decisions without and with concurrency, based on the average renewable energy percentage in the summer of 2022.

2.4 Scaling and Resource Metrics Collection in OpenShift

To integrate the system at ING, the scheduler has been integrated into the OpenShift system. OpenShift is used to scale up and down the cluster as well as collect the used system resources. This is handled by directly interfacing with the OpenShift API. This API provides direct access to the OpenShift environment, given that the correct rights are assigned to the pod. Interfacing with the OpenShift API is handled in the `scheduler/scheduling_app/openshift` package.

The scheduling pod needs to be assigned certain rights, named roles in OpenShift, to be able to interface with the API. These rights are assigned by means of a `ServiceAccount`[2]. A `ServiceAccount` can be granted certain `Roles` by means of a `RoleBinding` and is defined in a `yaml` configuration file.

Similarly, gathering metrics on CPU and memory consumption by pods can be implemented by calling the OpenShift metrics API[3]. Again, the scheduling pod needs to be assigned certain rights by means of a `ServiceAccount`. Managing metrics measurements and querying them is fully managed by OpenShift and therefore requires little work to implement for the carbon-aware scheduler. This API exposes endpoints that allow the scheduler to retrieve CPU and memory percentages for every pod it manages.

With the `ServiceAccount` definition, the scheduler is able to perform its primary duties. It can scale up and down deployments in the OpenShift environment to start and stop processing pods, and it can call the OpenShift metrics endpoint to query all metrics for all the pods in the same namespace.

2.5 Energy Consumption Calculation for Pods

Finally, to be able to calculate the greenhouse gas emissions of the pipeline, data on kWh consumption is required for every processing

[2] https://docs.openshift.com/container-platform/4.14/authentication/understanding-and-creating-service-accounts.html
[3] https://docs.openshift.com/container-platform/4.14/monitoring/managing-metrics.html

pod. Once the total kWh consumption for a pod is known, it can be converted into carbon emissions. CPU and memory usage directly influence the electricity consumption of pods and are therefore used for the calculation of a pod's kWh consumption.

The process of acquiring kWh measurements requires two main parts. First of all, for every pod that is started, metrics are collected on its CPU and memory consumption. Secondly, an ING tool converts these metrics into the expected kWh consumption. This is encapsulated in the package found in the repository at `scheduler/energy_calculator`. The process is explained in the next few paragraphs and describes the parts shown in the purple box at the bottom of Figure 2.

OpenShift keeps track of CPU and memory consumption statistics for every pod in the system and provides this data to external tools using an API. The scheduler must interface with this metrics API to gather the CPU and memory consumption statistics of pods. It requires certain rights to be able to do this, as explained in the previous section. A conversion to kWh consumption can be performed based on these metrics.

The conversion to kWh uses an ING-provided tool. This tool converts CPU and memory percentage measurements to kWh consumption. It determines the expected energy consumption based on measurements within the ING private cloud data centers. In other words, based on real-world measurements in the ING data centers, it is able to calculate kWh consumption based on CPU and memory statistics. The tool is configurable based on platform and hardware configuration. It takes into account the average minimum and maximum watts consumed by a platform, along with the specific hardware configuration on which the pods run.

3 EVALUATION

We presented the implementation of our CO_2-aware task scheduler. This section presents the methodology, results, and evaluation of our contribution. During this evaluation, we will answer the following research questions:

RQ1 To what extent can task load be predicted regarding running times? In this first research question, we evaluate the effectiveness of our approach to predict the required execution time for a task.

RQ2 How can predictions on solar and wind energy generation, as well as energy consumption, be utilized for implementing a carbon-aware scheduler? In this second research question, we evaluate the effectiveness of our approach to schedule tasks while aiming to reduce the amount of CO_2 produced during the execution. To do so, we first evaluate the identification of the low carbon intensity period, then we measure the amount of reduction in carbon emissions achieved by using a carbon-aware scheduler.

RQ3 Is there any overhead introduced by implementing a carbon-aware scheduler? In the final research question, we explore the impact of the overhead introduced by `S.C.A.L.E`.

3.1 Methodology

To evaluate our approach, we replicate the ING system in a standalone distributed system. The goal of the replicated system is to provide a sandboxed environment where the experiments can be executed and to provide a replication package for the contribution.

To do so, we implemented `S.C.A.L.E` as an open-source project that is compatible with OpenShift as well as Docker.[4] We also rely on open-access data for the evaluation instead of using ING proprietary data. The implemented platform is presented Section 3.1.1 and the data used for this evaluation is presented in Section 3.1.2. This effort has been made to improve the generalization of the approach and the replication of this contribution.

3.1.1 Processing Pipeline. For the sake of the evaluation, we recreated the ING processing Pipeline that ING uses to analyze internal Bloomberg Chat data. Figure 7 presents the ING pipeline and our evaluation pipeline. It shows that the data flows similarly through both pipelines. The pipeline follows the following steps:

(1) The first step consists of moving files from the `pending` bucket to the `processing` bucket. The original pipeline performs a data integrity check that is not required for the evaluation.
(2) The second step decompresses the bucket of files into the processing bucket. The input file is an encrypted `tar.gz` file and the resulting output is multiple decompressed files uploaded to the `processing` bucket. The original pipeline also decrypts the input bucket before decompressing it.
(3) The last step differs slightly from the original pipeline which transforms the Bloomberg Chat from the XML files to a proprietary format used by the ING conversation analyzer. For this evaluation, this step performs sentiment analysis on the chat to simulate the ING conversation analyzer

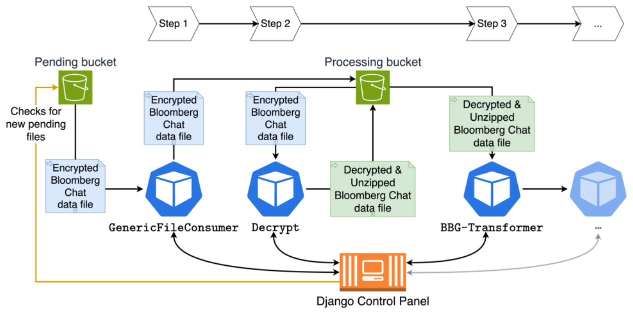
3.1.2 Dataset. In this section, we present the input that we used for the evaluation. We did not use the data-sensitive input file used by ING, but we replaced it with similar data. We chose a Twitch chat dataset [9]. This dataset is a collection of chat logs of 2,162 Twitch streams by 52 streamers, ranging from April 24th, 2018, to June 24th, 2018. It was chosen because it contains textual chat logs in which multiple people are conversing with each other. Therefore, it accurately represents one of the significant types of input data used by the ING system. Additionally, it contains chat logs over a long period of time, thereby providing a large variance in the number of chat logs per day.

This data is then transformed to match the ING input file. More specifically, we group the conversations per day and split the big Twitch chat into multiple files to simulate daily conversation. Finally, we compressed the resulting files. The resulting dataset contains 468 conversations spread over 3 months.

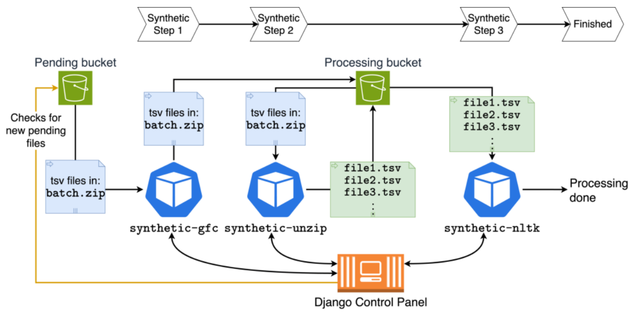
3.1.3 Data collection. To evaluate our scheduler, we executed the evaluation pipeline on the dataset that we previously presented. During the execution, we collect all metrics required for our scheduler to work such as the starting and end time of each task as well as the CPU and memory usage.

Additionally, We also converted the kWh used to perform our evaluation to CO_2. We used Electricity Maps [12], which provides data on the environmental impact of energy generation in a specific region at a given time. The degree of cleanliness in electricity generation indicates the amount of greenhouse gases emitted per kWh

[4] https://github.com/Fastjur/S.C.A.L.E./



(a) trade and communications surveillance pipeline.



(b) Synthetic pipeline.

Figure 7: The ING processing pipeline to analyze Bloomberg Chat compared to the evaluation pipeline.

of electricity generated and consumed. This metric is quantified in gCO_2-eq/kWh. The platform factors in various greenhouse gases, converting their emissions to an equivalent CO_2 value based on their global warming impact over a 100-year period.

The calculation of gCO_2-eq/kWh on Electricity Maps includes a Life-Cycle Analysis that assesses the environmental impact of a product or process throughout its entire life cycle, from raw material extraction to production, distribution, use, and disposal. In the context of electricity generation, this means considering emissions not only from the direct combustion of fuels during electricity production but also from activities such as fuel extraction, transportation, and plant construction. For example, it considers the extraction of metals like iron for manufacturing steel used in windmills, as well as the anticipated greenhouse gas emissions associated with the construction and dismantling of the windmill.

The data provided by Electricity Maps contains gCO_2-eq/kWh emissions per hour for 2021 and 2022 in The Netherlands. For this evaluation, we use season averages for electricity generation statistics and carbon intensity instead of real-time data. There are two main reasons for this. Firstly, it is the purpose of the tests to estimate the expected reduction of CO_2 emissions throughout the four seasons of the year. The weather at the time of conducting the tests should not impact the results. Secondly, it allows for a predictable input to the scheduler. This allows for testing the system with and without the scheduler, using the same input into the scheduler regarding electricity generation statistics. The averages of renewable energy generation statistics, such as predicted wind and solar generation, are based on data by ENTSO-E. Section 2.2

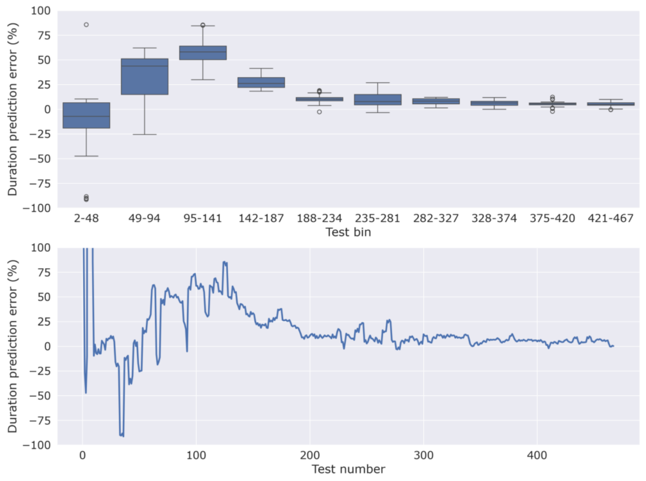


Figure 8: Boxplot shows the percentage of test errors over time, it is divided into 10 equally sized bins. The bottom plot shows the error percentage of all tests over time. Tests are ordered by starting time from.

provides an explanation of how this data is gathered and shows the averages per season in 2022.

To determine the average emissions per kWh generation in The Netherlands, the data from Electricity Maps [12] is used. This is a platform that provides predictions on gCO_2-eq/kWh emissions, as well as historical emissions per kWh for a certain bidding zone or country. The latter data was used to calculate the average emissions per season per time of day in The Netherlands.

3.2 RQ1. Execution Time Prediction

In this first research question, we evaluate the accuracy of our approach to predict the execution time. The accuracy of the predictions is calculated by determining the difference between the expected duration prediction and the actual duration.

A total of 468 tasks were processed by the system, where every task contained one synthetic test file as described in Section 3.1.2. Of those tasks, 467 were completed successfully. The maximum concurrency during this test was set to 1; i.e., at all times, only one task containing one synthetic test file was processed at a time.

Figure 8 shows the evaluation of the processing speed predictions over time. The top graph displays the tests grouped into 10 bins. Every bin, therefore, contains 46-47 tests grouped together based on their starting time. From left to right, the graph shows the performance of the predictions over time, with the first bin containing the oldest tests and the last bin containing the most recent tests. The y-axis shows the percentage error of the predictions. The bottom graph shows a plot of the percentage errors of the tests as a line, also ordered by their starting times from left to right. The most recent tests have more historical data available to them and, therefore, take more tasks into account when calculating the expected processing speed.

The graphs in Figure 8 show that the scheduler's predictions improve over time as more tasks have been processed by the system. The first few tests heavily overestimate and underestimate the running times of new tasks, explaining the very jittery lines for the first few tests. After roughly 100 tests, the system starts to converge to a lower error rate. Eventually, the system accurately predicts tasks with an error percentage between 5 and 10%, a 5 to 10% error results in an error of 12 to 24 minutes if the expected running time of a day's tasks is 4 hours.

Table 1 shows the data from the first 12 tests that were performed. Tests 5 to 8 overestimated the running times up to 2540% but quickly subdued back to a much smaller percentage error from row 9 onwards. The source of these overestimations can be found in the bottom table when looking at the `File size (KiB)` column. When starting test number 5, the scheduler only had information from tests 1 through 4 available to determine the median processing speed. As can be seen, tests 1-4 contained much smaller files as input. Tests 1-4 had a processing speed between 939 and 2.064 B/s, whereas tests 5-9 had a higher processing speed of 35.000 B/s.

This can be explained by the overhead of starting up a container for every step in the processing pipeline. For example, the third step performs some natural language processing on the chat data. For this step, the 'Natural Language Toolkit' Python package needs to download some datasets into the container. For every step, this happens once when the pod starts up before processing all files of that test. This adds some overhead to the test, as it needs to download this tooling into the docker container before being able to process all the chats of that task. This overhead reduces the processing speed of tests with very small input files, as the preparation steps of the task require more time proportionally than the actual processing compared to tests with larger input files. However, as can be seen in the bottom table, over time, the mean and median processing speeds actually recover to normal values. Therefore, the median processing speed will take into consideration the processing speeds of large as well as small input files.

Answer to RQ1. the prediction errors settle down to between 5 and 10%. Therefore, the scheduler will have a prediction error of 12 to 24 minutes if a day's batch of tasks requires 4 hours of processing time. This is an acceptable margin of error. Therefore, it can be concluded that a task's running time can be predicted accurately, given enough historical data has been processed by the system.

3.3 RQ2. Scheduler Effectivneness

In this research question, we evaluate the amount of reduction in CO_2 emissions that our scheduler can achieve. An extensive full-system test was set up to evaluate the scheduler's performance, which will be explained in this section.

All the gathered synthetic data was processed by the system whilst measuring container kWh consumption to determine the expected reduction in greenhouse gas emissions. The differences in greenhouse gas emissions can be calculated using the season averages of gCO_2-eq/kWh emissions per consumed kWh. Section 3.1.2 explains the creation of the synthetic data used in this test.

The system also calculated the consumed kWh for every pod. To do so, for every pod that is started, a separate thread is started by the scheduler, polling the docker API for CPU and memory consumption statistics. The total kWh consumption can be calculated for every pod from the CPU and memory consumption statistics using the same methodology as explained in Section 2.5. These values are calculated roughly every second and are saved separately to the file system after the test has been performed. The kWh measurements are also aggregated in the `Metric` class for every synthetic input file. In other words, the kWh measurements from every container that performed processing on the arriving synthetic source file, as well as any containers that processed any derived files from that source file are aggregated into a single metric.

It is essential to save the individual kWh measurements for each container, not just aggregates per metric. The main reason is that long-running tasks may take hours to process. Therefore, the expected gCO_2-eq/kWh emissions may vary throughout the processing time of that task. Two scenarios are simulated to evaluate the expected reduction in greenhouse gas emissions by implementing the carbon-aware scheduler. The first scenario represents a system before the scheduler optimizes it, and the second scenario represents a system that uses the scheduler.

The first scenario is one in which the files arrive randomly throughout the day, with a uniform probability for every time of the day. In this scenario, a task is immediately processed upon arrival, as is the case in the real ING system. A uniform probability for the arrival times is chosen in this scenario to reduce the influence of arrival times on the expected reduction in emissions. The arrival time of input data has a significant impact when comparing situations with and without the scheduler.

The second scenario chooses the optimal time of the day to "schedule" the processing of a task, and it simulates as if the scheduler delayed the execution of the task to this time. The optimal time of the day is the point at which the gCO_2-eq/kWh emissions are lowest. Utilizing this optimal value in this test provides an upper bound on the expected reductions by the carbon-aware scheduler and simulates a best-case scenario of implementing it. Both simulations are performed for every season of the year, using the season averages of gCO_2-eq/kWh emissions in 2021. For both scenarios, cumulative emissions are calculated and ordered by the test date.

Figure 9 summarizes the results for this research question. It shows a side-by-side comparison of the cumulative gCO_2-eq emissions by the system in both scenarios. The left graph shows the scenario without the carbon-aware scheduler, and the right graph shows the scenario using the scheduler. The x-axis displays the synthetic tasks, ordered by their date. The y-axis displays the cumulative gCO_2-eq emissions for both scenarios. The graphs show no discernable differences between seasons. It also shows that the system using the carbon-aware scheduler performs better than the system without it in terms of greenhouse gas emissions.

Table 2 presents the cumulative gCO_2-eq emissions for both scenarios, for all seasons. The CO_2-aware scheduler is able to save around 20% emissions compared to the scenario in which files arrive randomly throughout the day.

Answer to RQ2. The results of this research question show that `S.C.A.L.E` is able to reduce the reduction production of greenhouse gasses by around 20% by intelligently scheduling the tasks. The reduction varies depending on the season, the reduction is the highest in summer, when there is a larger difference in renewable energy generation throughout the day.

Table 1: Data from first 12 tests of the test to determine if the processing speed converges.

			File size	Processing speeds		Durations		
#	Proc. speed	Error	(KiB)	Mean	Median	Prediction	Actual	Error
1	2.064	106%	27,2	2.064	2.064	00:00:27.862000	00:00:13.498787	106%
2	1.511	27%	19,6	1.787	1.787	00:00:09.704282	00:00:13.258699	-27%
3	939	47%	12,3	1.505	1.511	00:00:07.049457	00:00:13.420734	-47%
4	1.360	10%	17,7	1.468	1.435	00:00:12.020205	00:00:13.354103	-10%
5	37.892	2540%	1305,9	8.753	1.511	00:15:31.698940	00:00:35.290904	2540%
6	36.890	2342%	1272,3	13.443	1.787	00:14:22.433027	00:00:35.317646	2342%
7	35.219	1870%	1214,7	16.554	2.064	00:11:35.915645	00:00:35.317486	1870%
8	34.833	1588%	1218,8	18.838	18.449	00:10:04.647016	00:00:35.828439	1588%
9	34.224	86%	995,7	20.548	34.224	00:00:55.264734	00:00:29.790370	86%
10	30.855	-10%	943,1	21.579	32.540	00:00:28.216634	00:00:31.297615	-10%
11	33.069	2%	991,0	22.623	33.069	00:00:31.185316	00:00:30.686416	2%
12	30.692	-7%	909,2	23.296	31.962	00:00:28.154662	00:00:30.334998	-7%

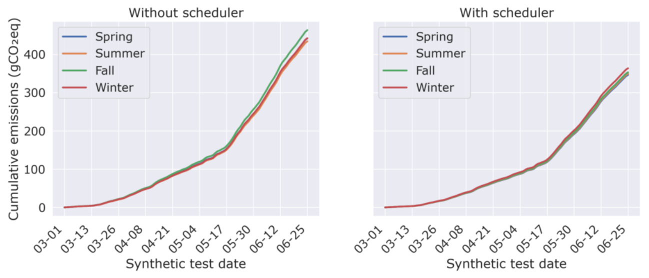


Figure 9: Comparison of cumulative gCO$_2$-eq emissions with and without the carbon-aware scheduler. The x-axis displays the synthetic tasks, ordered by their date. The y-axis displays the cumulative emissions.

Table 2: Comparison of cumulative gCO$_2$-eq emissions for the system with and without the carbon-aware scheduler.

Season	Scheduler (gCO$_2$-eq) Without	With	Difference	%
Spring	442.76	346.91	-95.85	-21.65
Summer	435.06	350.15	-84.91	-19.52
Fall	464.43	353.92	-110.51	-23.79
Winter	443.25	364.41	-78.84	-17.79

3.4 RQ3. Scheduler Overhead

In this final research question, we discuss the potential overhead that this approach could bring to the system. We identify two potential overheads. The first overhead to consider is that of the scheduling algorithm itself. However, the scheduling decisions are very simple. Therefore, the added overhead by the scheduler is negligent. The scheduler takes less than a second to determine the optimal schedule for a full day's worth of tasks to process; it is, therefore, inconsequential to the overall processing time and resource consumption.

Secondly, the overhead introduced by parallel processing should be considered. Whenever any system introduces parallel processing or multi-threading, there is an overhead associated with it. The source of this overhead depends on the system and the processing done. For example, there needs to be communication between different parts of the system to ensure synchronization of the processing and prevent deadlocks. At some point, the overhead added by parallel processing will outweigh the benefits of the parallel processing itself.

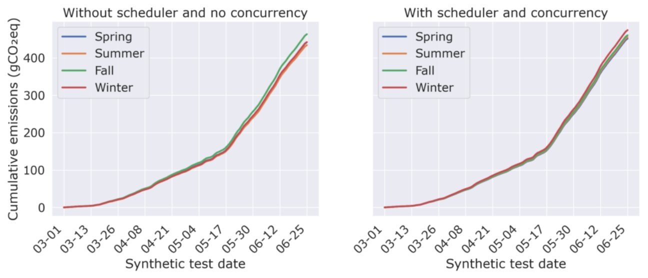


Figure 10: Comparison of cumulative gCO$_2$-eq emissions. The left graph shows the scenario without the carbon-aware scheduler and without concurrency. The right graph shows the scenario using the scheduler and using a maximum concurrency of 4. The x-axis contains the tasks, ordered by their date. The y-axis displays the cumulative emissions.

We performed the evaluation of RQ2 using the concurrency of 4 tasks executed in parallel. Figure 10 shows the results of this experiment. The left graph shows the scenario without the carbon-aware scheduler and without concurrency. The right graph shows the scenario using the scheduler and using a maximum concurrency of 4. The x-axis displays the synthetic tasks, ordered by their date. The y-axis displays the cumulative gCO$_2$-eq emissions for both scenarios. This is performed for every season, using the season averages of gCO$_2$-eq/kWh emissions in 2021. For both scenarios, cumulative emissions are calculated and ordered by the test date.

Table 3 presenters the cumulative gCO$_2$-eq for both scenarios, for all seasons. The table shows that the added kWh introduced by the overhead of processing tasks concurrently outweighs the benefit of scheduling tasks concurrently in this scenario. The system emitted more greenhouse gasses for all seasons except during fall. In fall, there is a very light reduction but only about half a percent.

Table 3: Comparison of total emissions without and with the carbon-aware scheduler and without and with concurrency.

Season	Scheduler (gCO$_2$-eq) Without	With	Difference	%
Spring	442.76	452.38	+9.62	+2.17
Summer	435.06	456.61	+21.55	+4.95
Fall	464.43	461.52	-2.91	-0.63
Winter	443.25	475.19	+31.95	+7.21

However, it should be noted that the tipping point of this is influenced by task size and system specifications. In the synthetic scenario, processing tasks concurrently does not add any significant gain, as the tasks only run for a maximum of 10 minutes. Therefore, the added gain by processing in roughly a quarter of the time is very small. However, as task sizes start to increase, the expected gain will increase, as the scheduler is then able to run more tasks at peak times of renewable energy availability. The tipping point should be determined experimentally for every system that introduces a carbon-aware scheduler.

Answer to RQ3. The results of this research question show that the scheduling algorithm itself is very lightweight and, therefore, does not add any meaningful resource consumption. However, an overhead is expected when adding concurrent processing of tasks. It is shown that, at some point, the overhead outweighed the benefits of implementing concurrency and a carbon-aware scheduler but this observation is highly dependent on the task and the system. Therefore, each system and task set should experimentally determine this tipping point.

4 RELATED WORK

In this section, we review prior research and works related to the development of a carbon-aware scheduler, grouping them by topic.

4.1 Global Warming, Carbon Emissions and Impact of Internet Services

This group delves into the broader environmental context surrounding global warming, carbon emissions, and the consequential impact of internet services. Studies in this category, such as those examining the environmental consequences of electricity generation and estimating global energy use of ICT networks, provide essential insights into the environmental footprint of information and communication technologies and data centers.

4.1.1 Climate Change and Greenhouse Gas Emission Sources. The issue of global warming and carbon emissions has gained significant attention in recent years. Yoro and Daramola [20] discuss CO_2 emission sources, global warming, and climate change, focusing on environmental impacts. They provide insights into the impact of various industries and sectors, including coal-fired power plants and fossil fuels. Raza et al. [15] provide a review of the impact of climate change on agriculture, showing the adverse effects on crop adaptation and global food security. These works offer an understanding of the environmental impact of CO_2 emissions and are crucial for showing the relevance of the work in this paper and other carbon-aware computing approaches.

4.1.2 Data Center Power Consumption. Brown et al. [3] present a report to Congress on server and data center energy efficiency. The report discusses the energy consumption of data centers and provides recommendations for improving energy efficiency in these facilities. Their work highlights the importance of reducing carbon emissions from internet services.

Kaplan et al. [6] state the significant impact of data centers on carbon emissions, and thus, to that extent, the impact of services ran on them. The work shows that data centers are largely inefficient and underutilized. Their work provides insights into the strategies and technologies that can be employed to enhance the energy efficiency of data centers. It also shows that there is a large margin to work with regarding the maximum utilization of data centers. This margin can be used to postpone batch compute jobs to later times when the carbon load is lower, as apparently the capacity to do that exists in data centers.

Jones [5], Andrae [1] and Dayarathna et al. [4] addresses the energy consumption of data centers and the modeling thereof. They emphasize their significant environmental and financial costs. The articles highlight the need for increased efficiency and accountability in data centers to address the growing concerns related to energy consumption and environmental impact.

Katal et al. [7] provide a survey on software technologies for enhancing energy efficiency in cloud computing data centers. The study discusses software-based approaches at the virtualization, operating system, and application levels, aiming to reduce the energy consumption of data centers and address environmental concerns.

Radovanovic et al. [13] delve into accurately mapping data center resource usage to power consumption for Google data centers. They prove that there is a strong link between CPU and memory consumption in data centers and expected power consumption.

4.2 Green Energy and Carbon Emission Variability

Focused on the variability of green energy and carbon emissions, this group explores strategies for managing data center power efficiently and enhancing energy efficiency in cloud computing. The studies within this category contribute to the understanding of how green computing and renewable energy usage can be optimized to reduce the carbon footprint associated with information and communication technologies. The variability of green energy sources and their impact on carbon emissions is an important consideration in the transition towards a more sustainable energy system.

Khan [8] conducts a temporal carbon intensity analysis comparing renewable and fossil fuel-dominated electricity systems. The study reveals the temporal variability of carbon intensity, explores the interaction between electricity generation and emissions, and identifies peak carbon-intensive hours. In other words, the study investigates the carbon intensity of different electricity generation sources over time. It highlights the significant differences in carbon intensity at different hours and shows the importance of renewable energy integration in reducing carbon emissions.

Thind et al. [17] explore the environmental consequences of electricity generation. The study, focused on the Midcontinent Independent System Operator (MISO) region, reveals temporal trends in emission factors. They analyze the carbon intensity of different electricity generation methods by hour, day, month, and year. This provides valuable information for understanding the environmental impact of electricity consumption at different times throughout the day, generated by different sources. This work is relevant because it provides a basis for understanding varying greenhouse gas emissions at different times throughout the day, which is the basis of the carbon-aware scheduler in this paper.

4.3 Other Carbon-Aware Scheduler

This group contains diverse works displaying carbon-aware solutions beyond scheduling, including approaches to data center energy efficiency and the potential of smart grids. The studies highlight frameworks and strategies for optimizing energy consumption and carbon efficiency in various contexts, offering insights for developing a carbon-aware scheduler.

Radovanović et al. [14] present a system focussed on minimizing the electricity-based carbon footprint of Google data centers. By leveraging analytical pipelines, day-ahead demand prediction models, and risk-aware optimization, the system generates carbon-aware Virtual Capacity Curves. The study demonstrates effective limitations on hourly capacity during carbon-intensive periods, contributing to a more sustainable operation of data centers. It serves as a basis for the ideas implemented in this paper.

Smale et al. [16] explore the shifts in goals concerning domestic energy uses in the context of smart grid transitions. The study analyzes how smart grids can be implemented to ensure that electricity grids can handle increased demands. Additionally, they argue that renewable energy consumption can be balanced more effectively by implementing smart grids. A carbon-aware scheduling system, which is the subject of this paper, can be a part of such a smart grid.

In the context of internet services, Le et al. [10] propose an approach to capping the brown energy consumption of internet services, emphasizing the importance of renewable energy usage. They address the issue of reducing carbon emissions from data centers and internet infrastructure. Their approach focuses on reducing the carbon footprint of internet services by leveraging renewable energy sources and optimizing the allocation of workloads across data centers in different geographical regions.

5 THREATS TO VALIDITY

The use of synthetic data as a stand-in for real ING system data represents a potential threat to the validity of this work. Despite efforts to mimic the real data, differences in data structure, type, and processing steps between the synthetic and real systems could impact the results. The synthetic data, extracted from Twitch chat logs, differs in content and size from the real system data, which comes from various sources such as email and direct messaging apps. Additionally, the processing steps in the synthetic system, though representative of text processing steps in the ING system, are not an exact match. Despite these differences, the synthetic system is considered a good representation of the ING system, and the results are assumed to indicate results in the real system.

The conversion from CPU and memory statistics to kWh consumption is based on an ING-provided tool designed for the ING private cloud. However, the tests were not performed on the ING private cloud but on containers in a docker environment. This discrepancy will result in a difference between actual and expected kWh consumption. However, it is not considered a significant threat, as the study aims to show a relative difference in emissions between scenarios with and without the carbon-aware scheduler. Both scenarios use the same methodology for calculating kWh consumption, so any error should affect both scenarios similarly.

The potential for unnoticed mistakes or bugs in the code of the carbon-aware scheduler or in the simulation of the ING pipeline is another concern. These could impact the scheduler's performance and influence the evaluation outcomes. Although efforts were made to test and validate the code, it is important to acknowledge that potential issues in the code may affect the presented results.

6 CONCLUSION

This paper aims to explore the potential of reducing greenhouse gas emissions by intelligently scheduling batch processing jobs, thus decreasing the environmental impact of data centers. A carbon-aware scheduler, `S.C.A.L.E` (Scheduler for Carbon-Aware Load Execution), was developed to optimize a resource-intensive data processing pipeline at ING, one of the larger consumers of the ING private cloud.

`S.C.A.L.E` reduces emissions by scheduling resource-intensive processing jobs during green energy hours, leveraging the variability of renewable energy production. It uses 24-hour forecasts on solar and wind generation and grid load to determine optimal scheduling times, aiming to maximize the use of renewable energy.

The scheduler comprises three modules, one for predicting task running times based on previous runs, another for providing predictions regarding renewable energy generation and electricity grid demand, and the third for interacting with the processing pipeline. The accuracy of the scheduler's predictions was validated, showing that it can predict task running times with an acceptable error margin of 5 to 10%, and accurately determine periods of low-carbon-intensive energy generation.

The effectiveness of the scheduler varies depending on the season and the expected arrival time of the batched input data. It was found that the scheduler reduces carbon emissions in all seasons, but the expected reduction varies depending on the input data's arrival and processing times. During our experiment, we show that `S.C.A.L.E` reduces the carbon footprint by around 20%.

In conclusion, `S.C.A.L.E` has demonstrated the feasibility and benefits of implementing a carbon-aware scheduler to reduce greenhouse gas emissions in resource-intensive data processing pipelines. ING will continue developing and generalizing the scheduler to provide it as a service in its cluster.

DATA AVAILABILITY

`S.C.A.L.E` and the results of the experiment are available at https://github.com/Fastjur/S.C.A.L.E./.

REFERENCES

[1] Anders Andrae. 2019. Comparison of Several Simplistic High-Level Approaches for Estimating the Global Energy and Electricity Use of ICT Networks and Data Centers. *International Journal of Green Technology* 5, 1 (12 2019), 50–63. https://doi.org/10.30634/2414-2077.2019.05.06

[2] Tom Bawden. [n. d.]. Global warming: Data centres to consume three times as much energy in next decade, experts warn. *Independent* ([n. d.]). https://www.independent.co.uk/climate-change/news/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html

[3] Richard Brown, Eric Masanet, Bruce Nordman, Bill Tschudi, Arman Shehabi, John Stanley, Jonathan Koomey, Dale Sartor, Peter Chan, Joe Loper, Steve Capana, Bruce Hedman, Rebecca Duff, Evan Haines, Danielle Sass, and Andrew Fanara. 2007. *Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431*. Technical Report.

[4] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794. https://doi.org/10.1109/COMST.2015.2481183

[5] Nicola Jones. 2018. The Information Factories. *Nature* 561 (9 2018), 163–166.

[6] James M Kaplan, William Forrest, and Noah Kindler. 2008. *Revolutionizing Data Center Energy Efficiency*. Technical Report. McKinsey&Company.

[7] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury. 2023. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing* 26, 3 (6 2023), 1845–1875. https://doi.org/10.1007/s10586-022-03713-0

[8] Imran Khan. 2018. Temporal carbon intensity analysis: renewable versus fossil fuel dominated electricity systems. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects* 41, 3 (9 2018), 1–15. https://doi.org/10.1080/15567036.2018.1516013

[9] Jeongmin Kim. 2019. *Twitch.tv Chat Log Data*. https://doi.org/10.7910/DVN/VE0IVQ

[10] Kien Le, Ricardo Bianchini, Thu D. Nguyen, Ozlem Bilgir, and Margaret Martonosi. 2010. Capping the brown energy consumption of Internet services at low cost. In *International Conference on Green Computing*. IEEE, 3–14. https://doi.org/10.1109/GREENCOMP.2010.5598305

[11] Gurdeep Singh Malhi, Manpreet Kaur, and Prashant Kaushik. 2021. Impact of climate change on agriculture and its mitigation strategies: A review. , 21 pages. https://doi.org/10.3390/su13031318

[12] Electricity Maps. 2023. *Electricity Maps*. https://www.electricitymaps.com/methodology

[13] Ana Radovanovic, Bokan Chen, Saurav Talukdar, Binz Roy, Alexandre Duarte, and Mahya Shahbazi. 2022. Power Modeling for Effective Datacenter Planning and Compute Management. *IEEE Transactions on Smart Grid* 13, 2 (3 2022), 1611–1621. https://doi.org/10.1109/TSG.2021.3125275

[14] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne. 2023. Carbon-Aware Computing for Datacenters. *IEEE Transactions on Power Systems* 38, 2 (3 2023), 1270–1280. https://doi.org/10.1109/TPWRS.2022.3173250

[15] Ali Raza, Ali Razzaq, Sundas Saher Mehmood, Xiling Zou, Xuekun Zhang, Yan Lv, and Jinsong Xu. 2019. Impact of climate change on crops adaptation and strategies to tackle its outcome: A review. https://doi.org/10.3390/plants8020034

[16] Robin Smale, Bas van Vliet, and Gert Spaargaren. 2017. When social practices meet smart grids: Flexibility, grid management, and domestic consumption in The Netherlands. *Energy Research and Social Science* 34 (12 2017), 132–140. https://doi.org/10.1016/j.erss.2017.06.037

[17] Maninder P. S. Thind, Elizabeth J. Wilson, Inês L. Azevedo, and Julian D. Marshall. 2017. Marginal Emissions Factors for Electricity Generation in the Midcontinent ISO. *Environmental Science & Technology* 51, 24 (12 2017), 14445–14452. https://doi.org/10.1021/acs.est.7b03047

[18] Charlotte Trueman. [n. d.]. What impact are data centres having on climate change? *Computerworld* ([n. d.]). https://www.computerworld.com/article/3431148/why-data-centres-are-the-new-frontier-in-the-fight-against-climate-change.html

[19] John Vidal. [n. d.]. 'Tsunami of data' could consume one fifth of global electricity by 2025. *Climate Home News* ([n. d.]). https://www.climatechangenews.com/2017/12/11/tsunami-data-consume-one-fifth-global-electricity-2025/

[20] Kelvin O. Yoro and Michael O. Daramola. 2020. CO2 emission sources, greenhouse gases, and the global warming effect. In *Advances in Carbon Capture*. Elsevier, 3–28. https://doi.org/10.1016/B978-0-12-819657-1.00001-3

Received 2024-02-08; accepted 2024-04-18