Experimental Evaluation of Energy Efficiency Tactics in Industry: Results and Lessons Learned

Markus Funke*, Patricia Lago*, Esther Adenekan*, Ivano Malavolta*, Ilja Heitlager†
*Vrije Universiteit Amsterdam, The Netherlands
†Schuberg Philis, The Netherlands
m.t.funke@vu.nl, p.lago@vu.nl, adenekanesther13@gmail.com, i.malavolta@vu.nl, iheitlager@schubergphilis.com

***Abstract*—Integrating (and evaluating) energy efficiency tactics into daily industrial practice is challenging. This paper addresses the experimental evaluation of energy efficiency tactics in industrial contexts. Based on different real-world scenarios, we assess five energy efficiency tactics for cloud-based software through individual experiments conducted across two companies. The results of the experiments show significant improvements in energy efficiency for three tactics, with two others showing enhanced efficiency albeit without statistical significance.**

In addition to the experiments, we draw lessons learned and practical insights into utilizing tactics in industrial contexts. Our results could guide practitioners in selecting and applying the most suitable tactic for their individual context. By linking tactics that emerged in the literature with evidence-based measures, we help including sustainability in software architecture design decision making.

***Index Terms*—tactics, energy efficiency, cloud, experiment, sustainability, industrial practice**

I. Introduction

From a cloud *consumer* perspective, migrating software to the cloud promises to improve quality attributes (QAs) like availability [1], scalability [1], flexibility [2], and carbon efficiency [3]. However, this migration might be misinterpreted as "delegating" the fulfillment of these QAs to the cloud *provider*. What on the one hand might give the impression of having unlimited computing power has on the other hand led to an increase in the size of data centers and a need for more computing power, more storage space, and higher energy demands [4]. Even with the promise to be more carbon efficient, with an estimated 3% share of greenhouse gas emissions (GHG), data centers already surpass the 2.5% of the aviation industry [5].

With this increased demand for resources, the attention for data centers in society has also grown [6]. Cloud providers are therefore investing in strategies to improve the sustainability of their cloud services and data centers [7, 8, 9]. Strategies on the provider side include making data centers more energy efficient [10] or using more environmentally friendly energy sources [11]. However, the more obvious strategy is on the consumer side: designing the actual cloud-based software in order to use less energy. Recent research (*e.g.,* [12, 13, 14]) started addressing this problem by defining, or rethinking, architectural tactics [15] for energy efficiency. However, most of these energy efficiency tactics remain a theoretical concept, few of them having been evaluated and applied in practice.

Due to the lack of evaluation and industry involvement [16, 17, 14], Balanza-Martinez et al. [16] recognize a "call for action" where "academic researchers and industrial practitioners [need] to join forces for creating real impact". Industrial evaluation would help creating the necessary guidelines that software architects need [18, 19] to decide whether a particular tactic should be used in their context. Our research aims to respond to this "call for action" by working together with industrial practitioners and evaluating the impact of energy efficiency tactics for cloud-based software.

To achieve this goal, and following the suggestions of Kitchenham et al. [20], we are working closely with our industry partners to define and conduct five independent experiments. Based on these experiments, our **main contributions** are (i) empirical measurements for five different energy efficiency tactics in an industrial context; and (ii) lessons learned for helping software practitioners in selecting and applying tactics. The results show significant improvements in energy efficiency for three tactics, while two show enhanced efficiency albeit without statistical significance. We further identify two workflows choosing the right tactics: *tactic-driven* and *hotspot-driven*. Our contributions are relevant for both (i) researchers interested in the actual impact of tactics for cloud-based software and (ii) practitioners considering the utilization of tactics to improve the energy efficiency of their software applications.

II. Related Work

A few studies [21, 22, 23] consider architectural patterns [15] in the cloud context and measure their impact on energy consumption. Notwithstanding, all of them show that cloud patterns can effectively reduce energy consumption in cloud-based applications. Sahin et al. [23] are even able to increase energy efficiency up to 700% by using the *Decorator* design pattern. However, none of these experiments refer to an industrial context where the impact in a production environment is evaluated and discussed.

While "patterns package tactics" [15], tactics are tailored to focus on individual concerns [15, 24, 14]. The majority of research regarding tactics concentrates on the QA *security*. According to the systematic mapping study of Márquez et al. [17], 18 out of 79 papers discuss tactics related to security, while none of the primary studies are mapped to energy

This paper is based on thesis results from the students Esther Adenekan, Mochamad Rizal Hidayat, and Himanshu Pandey of the Erasmus Mundus SE4GD program. We thank all architects and practitioners participating in this study and the companies for providing the individual cases.

efficiency as dedicated QA. Despite these findings, Kazman et al. [25] argue that energy efficiency should indeed be treated equally to other QAs.

Vos et al. [12] follow up on this line and treat energy efficiency as a QA in the context of cloud-based software. Therefore, we consider this work as most relevant for our present research. In their study, Vos et al. identify a catalog of architectural tactics to optimize cloud-based software for energy efficiency through a literature review and interviews with industry practitioners. They identify 18 architectural tactics which are classified into three categories (*i.e.,* Resource Monitoring, Resource Allocation, and Resource Adaptation). The categorization is based on the taxonomic literature review by Paradis et al. [14].

We utilize the work of Vos et al. as a basis for selecting energy efficiency tactics that are relevant to the industry. All our tactics are either selected from this catalog or mapped to one of the presented tactics. Based on this catalog we evaluate five tactics in industrial contexts.

III. Study Design

Our study design follows the well-defined research guidelines by Wohlin et al. [26]. First, we outline the study goal and define our research questions. Then, we present an overview of our methodology by discussing each research and experiment phase. We provide an online replication package [27] to promote reproducibility and strengthen the reliability of our experiment findings. The package includes more documentation to explain the environments of each experiment (*i.e.,* scripts for testing, automation, and data analysis) and tactic implementation (*i.e.,* architecture diagrams and documentation). To comply with privacy agreements, we have excluded confidential information.

A. Goal and Research Questions

The goal of this study is twofold. First, we empirically evaluate the impact of a set of energy efficiency tactics in an industrial context. By bridging the gap between tactics from the literature and having evidence-based measures, we can draw conclusions about the effectiveness of these tactics. Second, to better guide practitioners in selecting and applying tactics in their own context, we draw lessons learned while designing and executing the experiments together with our industrial partners. The goal mentioned above is achieved by answering the following two research questions:

- $RQ1$ *What is the impact of energy efficiency tactics on the energy consumption of cloud-based software in industry?*
  This research question is answered by (i) applying five tactics across four different real-world scenarios from two companies and (ii) evaluating their impact through experiments.
- $RQ2$ *How can energy efficiency tactics be effectively identified and utilized in an industrial context?*
  This research question is answered by monitoring the workflow of the involved practitioners on how a tactic is selected and utilized for each of the identified scenarios. Based on these insights, we draw conclusions and lessons learned.

B. Study Design

Each individual experiment is organized according to two main phases and five consecutive steps. Together, the experiments constitute our overall study design (see Figure 1).

1) Application and Tactic Selection: To identify the experiment subjects and use cases, we conduct an exploratory review involving the two companies and their practitioners. In total, three researchers are involved in guiding the exploratory study and leading the experiment, of which one researcher and five practitioners are dedicated to company A, and two researchers and four practitioners are involved with company B.

Company A is an ICT consultancy company providing support for mission-critical operations for diverse businesses in the Netherlands. As this study is related to both cloud-based software and cloud energy efficiency tactics, we chose to collaborate with experts from this company, having technology-agnostic expertise in improving cloud infrastructure. This allowed us access to a variety of different business cases and access to the knowledge of experts working in and consulting for different business domains.

Company B is a large financial institution committed to improving the sustainability of its daily operations and its cloud-based software portfolio. Company B has initiated a strategic shift towards consolidating its cloud services under a single provider—Microsoft Azure. Therefore, we classify it as technology-specific, adhering to several standards and regulations associated with this particular platform.

With the consulting company on one side and the large financial enterprise on the other, we contribute to a diverse environment of companies for our research with the ambition to derive results applicable beyond our study context. As we are also interested in the workflow of identifying a relevant case and tactic (RQ2), we refrain from proposing a "one-size-fits-all" approach for this identification process. Instead, we collaborate closely with practitioners in each company to formulate their specific workflow to select the right application and best-fitting tactic. Each researcher conducted several rounds of informal interviews with their practitioners to first determine the application and tactic, and then interpret and analyze the results. Questions are based on the "tactics-based questionnaires" identified by Paradis et al. [14], such as *"Which of the tactics do you consider applicable to the use case?"*. We aim to derive potential differences in the way how a certain application and energy efficiency tactic is selected. The results of this exploratory study are therefore presented individually for each case in Section IV.

2) Experiment Design: Based on the previous selection, an individual experiment is designed. Each experiment consists of at least the subjects' application and tactic. Together with the energy consumption in Watt-Hour, these subjects represent the experiment variables. If applicable for a certain experiment, we define a load size as a variable, having various

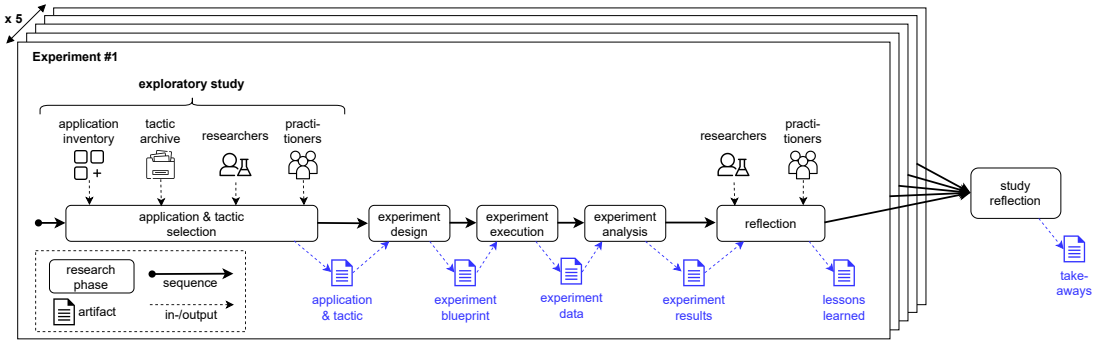

Fig. 1. Study design composed by five individual experiments

treatments to observe the consumption under varying loads. For each experiment, we defined a pair of null-alternative hypotheses. Without losing generality, all pairs of hypotheses have the following form.

$$H_0^t: \mu_o = \mu_t$$

$$H_1^t: \mu_o \neq \mu_t$$

where t represents a specific architectural tactic, μ_o represents the average energy consumption of the original version of the application, and μ_t represents the average energy consumption of the version of the application where the tactic is applied. The null hypothesis states that there is no significant difference between the average energy consumption of the original applications and their counterparts where the tactic t is applied, whereas the alternative hypothesis states that there is a significant difference between the two populations. When applicable, each experiment defines additional variables, factors, or hypotheses (*e.g.,* CPU usage, response time). In this paper, we focus primarily on energy consumption. Due to space limitations, the results related to additional factors considered in the experiments are not presented in this paper, but can be found in the replication package [27].

3) Experiment Execution: For each experiment, two instances of the use cases are created: (i) the original version of the application, where the tactic is not applied, and (ii) the enhanced version where the energy efficiency tactic is applied. This allows a direct comparison and provides necessary data for statistically analysing the differences between those two versions. As described before, we employed a load test to vary the number of concurrent users in the experiment. For this load testing purposes, we used the open-source framework locust.io [28]. Due to the independent nature of our experiments, the load size varies between experiments.

At the time of writing, the cloud dashboards provided by the major public cloud providers are still lacking transparent information on the energy consumption of their customers' cloud workloads [12]. Therefore, we adopted the Cloud Carbon Footprint (CCF) tool, an open-source energy estimation tool developed by Thoughtworks [29], to estimate the energy consumption during experiment execution. The tool estimates the consumption based on resource usage reports provided by the cloud providers. If not differently specified in the actual experiment description (see results in Section IV), the final data set produced in each experiment contains (i) the load testing results, (ii) the resource usage log from the individual cloud monitoring tool, and (iii) the raw energy consumption estimations.

4) Experiment Analysis: The dataset produced in each experiment is analyzed by following the classic data analysis phases of measurement-based experiments in software engineering, namely: data exploration, normality check, hypothesis testing, and effect-size estimation (when applicable). Initially, we employ descriptive statistics to explore the energy consumption values, followed by a visual representation of their distributions via histograms, boxplots, and violin plots. After assessing normality using histograms and Q-Q plots, a relevant hypothesis test (*e.g.,* two-way ANOVA) is conducted, with a follow-up estimation of effect size in case the results of the statistical test are significant. Given our focus on the main results and lessons learned in this work, not all statistics are presented in detail in this paper; the comprehensive statistical analyses are accessible in the replication package [].

5) Reflection: We reflect on the experiment results for both (i) the energy efficiency tactic itself, and (ii) the workflow related to selecting and applying a tactic. By providing evidence-based measures for each tactic, we can better understand their impact in real-world environments and scenarios, which could help practitioners at an early decision process. By drawing conclusions from monitoring the workflow, we can better understand how tactics are selected and utilized in practice, providing certain guidelines for architects.

IV. Results

In this section, we present the results derived from the five experiments. Table I summarizes all experiments, use cases, and tactics applied. For each experiment, we (i) describe the application under study, the tactic itself, and its selection process; (ii) outline the experiment execution; and (iii) present its results and reflect on their implications. We identified three applications in total: company A provided one application, having two use cases, for three different tactics in total. Company B provided two different applications for two different tactics.

TABLE I
APPLIED TACTICS MAPPED TO THE DIFFERENT USE CASES. TACTIC NUMBER REFERS TO THE REUSABLE TACTICS OF VOS ET AL. [12].

Experiment	Company	Application	Use case	Tactic No.	Tactic
#1	A	Validation-App	Insufficient utilization of computing resources.	T13	Choose a fitting deployment paradigm by deploying as Serverless with scaling features.
#2	A	*ibid.*	*ibid.*	T15	Apply Granular Scaling.
#3	A	*ibid.*	Container images are stored indefinitely.	T05	De-allocate unused resources by applying retention policies to container repository.
#4	B	Mortgage-App	Non-native proxies consume more energy than cloud-native API gateways.	T10	Re-use software services.
#5	B	Audit-App	Data compression is not considered for data-intensive applications.	T14	Compress infrequently accessed data.

When describing the tactics, we map them to the model of reusable tactics from Vos et al. [12] and the related open-source archive of these tactics [30]. This allows for better comparison with other tactics and sets a common ground for future discussions. Contributing to RQ2, we also describe the workflow on how this particular tactic was selected in relation to the industrial application.

The experiment execution identifies the experiment variables to perform a comprehensive analysis of the causal relationships within this particular experiment. Following up, we present the results of the experiment. The summaries of our statistical analyses help the reader in understanding the main characteristics of our different data sets. We focus on the essential parts (*e.g.,* hypothesis test) to interpret our data.

For each experiment and tactic, we discuss the implications of applying the tactic in an industrial context. We outline the insights related to the tactic itself (answering **RQ1**), followed by the lessons learned from measuring an energy efficiency tactic in a real-world industrial context (answering **RQ2**).

Experiment #1: "Choosing a Fitting Deployment Paradigm"

1) Application and Tactic Selection:

Application. This experiment was applied in the context of company A, the consultancy company. As case, a major provider of logistic solutions in the Benelux was chosen based on the current necessity and also interest in making their cloud applications more resource efficient. The logistic platform has diverse workload deployed in the public cloud, making it a suitable case for assessing the impact of tactics on the energy consumption of cloud-based software.

The logistics platform comprises several applications such as a data warehouse application, an event processing platform, and business application services. A suitable application was selected by considering three different sources: (i) the architecture documents; (ii) resource consumption and "energy hotspots" [31] uncovered by the AWS Well-Architected Framework [32]; and (iii) informal interviews with solution architects. The inputs were used to inspect the application's workload characteristics, resource utilization, and change feasibility. This assessment aimed to identify applications that have the potential for energy improvements and are also realistic for implementing architectural changes.

Considering all trade-offs, a business application service (in the remaining called Validation-App) offering validation services to the users of the logistics platform was selected. The Validation-App follows a traditional three-tiered Web application architecture comprising front-end, back-end, and API service. All components are implemented as micro-services using a gateway service [33] for routing the traffic.

Tactic. Based on the energy hotspots identified while selecting the Validation-App (see Application Selection above), we found the back-end component performing database operations with very low resource utilization. The component is currently deployed on containers running continuously without scaling, even during low peak periods. Hence, we decided to apply the *Chose a fitting deployment paradigm (T13)*[1] from the model of reusable tactics [12].

Upon assessment of the tactic, the involved industrial partners pointed out that deploying the component as serverless (*i.e.,* AWS Lambda function [34]) can indeed improve the energy consumption, since a serverless architecture can automatically scale based on demand. However, there might be some performance issues due to the time it takes for a Lambda function to restart after being idle (*i.e.,* Lambda cold start issue). To empirically assess the impact of this tactic T13, the following experiment is executed.

2) Experiment Execution: Our first experiment aims to assess the impact of choosing a suitable deployment paradigm in the context of the back-end component of the Validation-App. The two deployment paradigms considered are: traditional Container deployment (original version) versus Serverless - Lambda deployment (version with applied tactic). To have complete control over the experiment and overcome the limitations in measuring the energy consumption of a fully managed service, we mimic the real-life scenario in a controlled environment. The preparation of the experiment and its architecture was done in close collaboration with industry practitioners to ensure that the application and deployment to AWS are representative.

The component consists of four endpoints performing database Create-Read-Update-Delete (CRUD) operations. For both deployment scenarios, the application logic, database

[1]The number in parentheses links to the tactic identified in [12].

storage and data remains the same. The only variation is in their deployment architecture. For the serverless deployment scenario, each endpoint is written as a lambda function. For this experiment, we varied the workload between 0 and 100 concurrent users. One test run lasted for one hour. This allows sufficient time for the peak number of users to be reached, followed by a sufficient amount of traffic needed to capture the energy consumption.

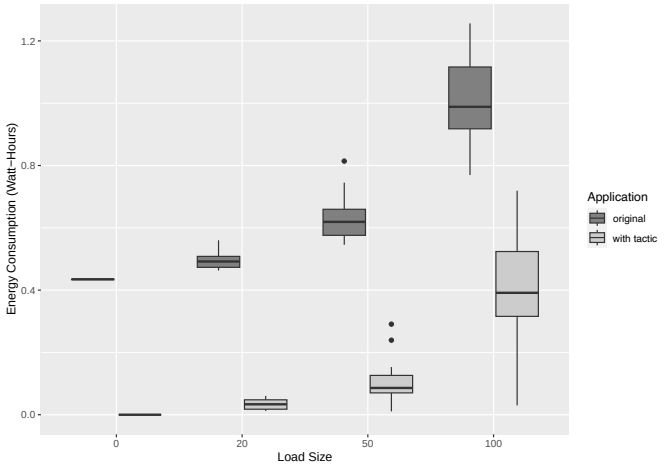

Fig. 2. Energy consumption (Watt-Hour) Original and With tactic (T13) *"choosing a fitting deployment paradigm"* across load sizes (users)

As shown in Figure 2, the energy consumption for the container-based deployment (original version) is higher than the serverless deployment (with applied tactic) across all load sizes. We further observe that for both types of deployment, the consumption increases as the load size increases. Compared by load size, the most significant impact is seen during Idle time (0 Load), as serverless yields more than **99%** energy improvement over container deployment. Since serverless deployment does not allocate computing resources when idle, energy is only consumed by the code storage. We record up to **80%** energy reduction for the other load sizes.

The two-way ANOVA test indicates a statistically significant difference between both deployment types ($p < 2\text{e}{-16}$) and load sizes ($p < 2\text{e}{-16}$) in terms of energy consumption. The interaction between deployment type and load size is not statistically significant ($p = 0.0815$). This indicates that in our experiment, the energy consumption is consistently affected by the deployment type, irrespective of the load size being used in this experiment.

3) Reflection: The consultation of the AWS Well-Architected framework uncovered the under-utilization of container resources and low workload of the Validation-App. Our experiment showed that selecting a fitting deployment paradigm can indeed enhance the energy efficiency of a cloud-based application. The most significant impact is observed during idle time (0 load), as serverless yields more than 99% energy improvement over container deployment. Although cloud monitoring dashboards identify areas for improvement and might imply an application of a certain tactic, such as the change in the deployment paradigm, the effort and cost of applying such a comprehensive architectural change to existing implementations may be relatively high. Instead of post-migrations, we therefore advise cloud developers and architects to carefully consider the different deployment options and the anticipated workload *before* committing to a certain deployment paradigm.

Tactic learning (T13): Applications with infrequent data traffic and an extended idle period can benefit significantly from serverless deployment and thus increase their energy efficiency.

Workflow learning: While dashboards such as a "Well-Architected framework" help identify areas for improvement, the trade-offs for applying tactics should be considered before implementation. Therefore, tactics should also be considered at design time.

Experiment #2: "Apply Granular Scaling"

1) Application and Tactic Selection:

Application. In this follow-up experiment, the same Validation-App was examined in the context of the logistics platform. Due to the remarkably low resource utilization of this app, additional opportunities for improvement were identified, prompting practitioners to suggest further investigations.

Tactic. Instead of applying the heavy architectural change, *i.e.,* moving from container to serverless deployment, *Applying granular scaling (T15)* could reduce energy consumption while preserving the application's structure. Applying granular scaling involves allocating a more suitable resource capacity to the workload instead of using the default capacity. In our case, this optimization can be done by replacing the single container, with multiple smaller containers having auto-scaling enabled. When selecting this tactic, practitioners pointed out that this tactic might indeed reduce energy consumption but might also deteriorate the performance of the application due to low-capacity resource provisioning. We again observe a trade-off discussion between energy efficiency and performance. We chose to empirically assess the impact of applying this tactic on the Validation-App from an energy perspective.

2) Experiment Execution: The two deployment paradigms considered for this experiment are containers without granular scaling (original version) versus containers with granular scaling (applied tactic). The same experiment environment and CRUD operations were considered as for the experiment before. To implement the tactic, the container resource for the back-end was replaced with multiple smaller containers to match the workload. Additionally, an auto-scaling policy was defined to increase or decrease the number of containers in response to the CPU resource utilization. A new container task is added when the resource utilization of the container is greater than or equal to 80%. The scaling policy monitors the utilization of application resources in AWS CloudWatch [35] to ensure the defined target is satisfied. The same workload (0 and 100 concurrent users) and test duration (one hour) as for the first experiment was chosen.

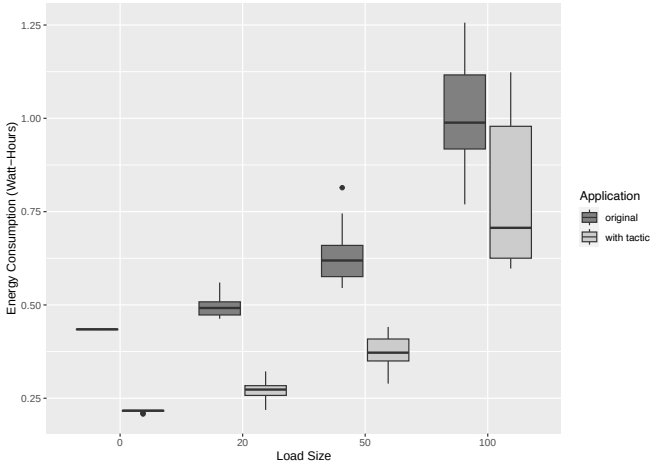

Fig. 3. Energy consumption (Watt-Hour) Original and With tactic (T15) *"apply granular scaling"* across load sizes (users)

As shown in Figure 3, the energy consumption without granular scaling (original version) is higher than the energy consumption with granular scaling (applied tactic) across all load sizes. We also observe that for both treatments of the tactic, the average energy consumption increases as the load size increases. For load size 100, a lower difference in energy consumption can also be noticed.

When assessed per load size level, we found that there is an average energy consumption reduction of up to **50%** when granular scaling is applied at load size 0, up to **45%** for load size 20, **40%** for load size 50 and up to **21%** reduction for load size 100. This shows that the tactic is more effective for smaller load sizes and less effective for large load sizes.

The two-way ANOVA test indicates a statistically significant difference for both with/without tactic ($p < 2e{-}16$) and load size ($p < 2e{-}16$) in terms of energy consumption. We observe a significant interaction between these factors ($p = 7.76e{-}07$, which is less than the significance threshold of 0.05). This suggests the existence of a significant interaction between the tactic application factor and the load size factor. This means that the impact of the tactic application factor depends on the load size assessed.

3) Reflection: Interestingly, for this particular area for improvement, two different tactics were suitable. Although serverless architecture (T13) would require major architectural changes and is therefore not suitable for all use cases, granular scaling (T15) appears to be a valid and convenient alternative. Therefore, even though a tactic might appear to be effective at first sight, architects should always consider alternative tactics and evaluate their trade-offs carefully.

We found that granular scaling could improve energy consumption between 21% and 50% on average, depending on the load size. Our experiment also showed that the tactic is more effective for smaller load sizes and less effective for large load sizes. Nevertheless, the concern about the potential impact on performance remains. Provisioning smaller resources means lower computing power, which could affect the performance of the application. This may not always be the case. Especially in our specific use case where the workload is rather light and resource utilization is low. Cloud developers should therefore continuously monitor capacity provisioning and performance to identify the right balance.

Tactic learning (T15): Applications with a smaller load size can benefit more effectively from granular scaling to reduce their energy consumption compared to applications with a high load size. However, trade-offs with performance should be considered.

Workflow learning: A given problem may benefit from multiple tactics, which bring multiple trade-offs. As such, problem solving requires informed decision making.

Experiment #3: "De-allocating Unused Resources"

1) Application and Tactic Selection:

Application. For the third use case in the context of the Validation-App, we studied the historical storage pattern of container images within the application using the cloud dashboard. Since the app is a container-based application, we found that many unused container images are being kept indefinitely, even though they are no longer needed.

Tactic. The only tactic considered suitable for this use case from the reusable tactics catalog model is the *De-allocation of unused resources (T5)*. One possible way to apply this tactic is to utilize a retention policy removing unused container resources after a certain period of time. This tactic was considered suitable by the industrial partners. They pointed out that retention policies should actually always be applied to remove unused images. However, one practitioner stressed the fact that the storage costs for the container images are insignificant, and therefore less attention should be paid to the removal of these container images. This tactic needs to be empirically assessed to determine whether setting retention policy to remove unused images will yield significant energy reduction for this particular use case.

2) Experiment Execution: For this third experiment, we collected historical data from the Validation-App container repository, including details of the container size and the date the container was created. We then analyzed the data to find the cumulative sum of containers stored per month over a period of one year. Using the estimation logic from the CCF energy conversion factor for storage, we estimated the energy consumption for each month under two scenarios: without setting a retention policy (original version) and with setting a retention policy (applied tactic) to de-allocate container storage. It should be noted that retention policies were not actually implemented in the Validation-App. We only made an estimate based on existing data and projections for the months in the last year, assuming that the retention policy is applied.

Taking the months of the year as the unit of analysis results in a sample size of 12. Figure 4 shows the dispersion of the data. We observe a difference in the energy consumption in the original version and with the tactic applied. Specifically, de-allocation unused containers results in less energy

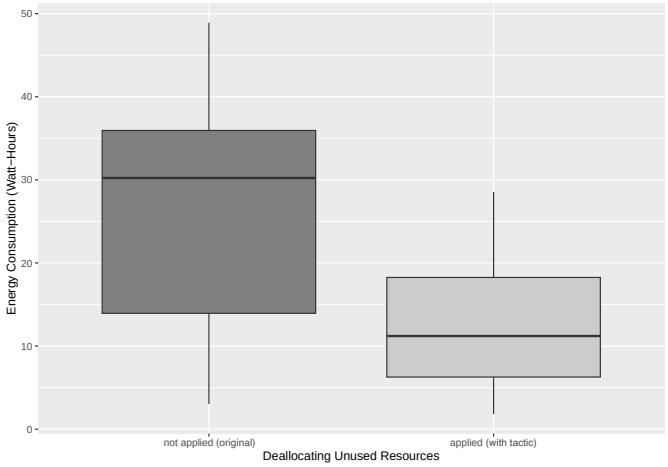

Fig. 4. Energy consumption (Watt-Hour) Original and With tactic (T05) *"de-allocating unused resources"*

consumption on average. The results obtained also indicate the positive skewness for the 'applied' treatment and negative skewness for the 'not applied' treatment. From the hypothesis testing with the non-parametric Mann-Whitney U test, we found that de-allocating unused container has statistically significant impact on the energy consumption of the cloud-based application ($W = 111, p = 0.02418$ which is less than the significance threshold of 0.05) and leads to energy reduction up to **53%** for the 1 year period assessed.

3) Reflection: This tactic acts as an example to achieve a more energy efficient solution without actual architectural change. Only by adjusting the cloud settings—impact can be made. Also, other tactics related to the category *resource allocation* [12], involving the management of resources for tasks or workloads [14], offer the possibility of reducing energy consumption without structural changes.

Given the significant improvement, we asked our practitioners if there are specific reasons to keep images indefinitely by standard. One practitioner mentioned that the images are stored in case of rollbacks—but only a few images are actually needed to be kept. They further mentioned that a good example of a retention policy for container images could be to delete unofficial container images (images pushed on every pipeline build) after one month, then official images (images that get deployed to production) can be deleted after one year.

This policy is often overlooked in practice, since the monetary cost of storing containers indefinitely is insignificant[2]. We suggest that cost should not be seen as the only motivation for resource optimization. In our case, applying retention policies to remove unused and outdated images can also help maintain a lean and efficient container image registry.

[2]$0.10 per GB/month for data stored in private or public repositories; Amazon Elastic Container Registry pricing, https://aws.amazon.com/ecr/pricing.

Tactic learning (T05): Establishing a company-wide retention policy, such as de-allocating unused resources and removing unused containers, can significantly reduce energy consumption.

Workflow learning: Cost should not be a motivator for pursuing energy efficiency and sustainability—it could obscure effective tactics.

Experiment #4: "Reuse software services"

1) Application and Tactic Selection:

Application. This experiment was carried out in Company B, the financial institution. In contrast to the use-cases from Company A, which involved identifying application hotspots *before* selecting potential tactics, the architects at Company B opted for providing a set of potential tactics. The tactics already exist in the company's environment and have resulted from internal investigations. The company is actively working towards a more sustainable cloud portfolio, resulting in various projects implementing energy efficiency tactics. Based on the tactic selection (outlined below), a mortgage platform that provides extensive back-end functionalities[3] was chosen as suitable application (in the remaining called Mortgage-App).

Tactic. The initial set is composed of tactics which are either (i) in production, (ii) under development, or (iii) potential future tactics, and are identified by both informal interviews with practitioners and reviewing architecture documents. In total, five tactics have been pre-selected:

(A) Use a cloud-specific API management gateway.
(B) Location shifting (parts) of your application.
(C) Time shifting (parts) of your application.
(D) De-allocate and schedule auto shutdown of Azure VMs.
(E) Auto scale down Azure SQL databases.

Tactic B and **C** have been excluded as they have not yet been employed within Azure environments. The tactics also challenge requirements to data center locations outside the Netherlands and adjustments to the operational hours, which might conflict with business policies. Similarly, we do not consider **tactic D** due to reliance on fluctuating workloads within the financial institution. Its implementation in a production environment could potentially create a bottleneck, affecting the availability of the software application. **Tactic E** is left out due to its heavy dependence on the financial institution's data storage needs; implementing could incur significant costs, thus placing it outside the scope of our empirical study. **Tactic A** is considered feasible as it does not require extensive data storage and is therefore selected.

Tactic A can be mapped to tactic *Reuse software services (T10)* according to Vos et al. [12]. When moving software applications to the cloud, it is generally advised to use cloud-native solutions compared to the pre-existing non-cloud-native ones, which results in reusing existing cloud-software services. In this concrete context of Company B, the cloud-native API gateway should be reused. The premise of this tactic is that the

[3]Due to a non-disclosure agreement, further insights about this application can not be provided.

proxy being non-native can consume more energy than an API gateway that the cloud provider offers. This premise is tested by our empirical assessment in the following experiment.

2) Experiment Execution: Also in this experiment, a controlled environment was created to mimic the functions of the Mortgage-App, exposing a Restful API endpoint. First, Flask [36] was used to perform testing and examine the behavior of the application. The application was then converted to the Java Spring Boot [37] framework to align with the real-world scenario. For this experiment, two different implementations are provided: (i) the API provider and consumer are mediated by an API proxy in the form of proxy VMs (original version); and (ii) the API provider and consumer are mediated by an cloud-native API gateway (applied tactic).

Azure was chosen due to the financial institution's active transition to the Azure environment. For the test of the original version, the experiment utilizes the Azure Web App, Azure VMs, and Azure Load Balancer. On the contrary, for the implementation with applied tactic, the experiment runs on Azure Web App and Azure API Management Gateway. The experiment encompasses ten trials for four distinct user workloads between 0 and 300 users.

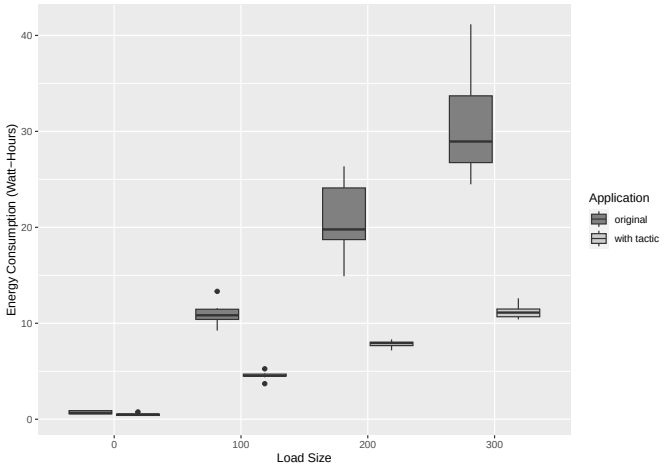

Fig. 5. Energy consumption (Watt-Hour) Original and With tactic (T10) *"reuse software services"* across load sizes (users)

The box-plot in Figure 5 shows that the energy consumption is higher for the implementation based on VMs and load balancer (original version) compared to the cloud-native API gateway (with applied tactic). It can also be observed that the energy consumption increases as the load size increases. Compared by load size, we record a decrease in energy consumption between **36%** (0 user) and **91%** (300 users).

When collecting results from the CCF tool, we discovered that although CCF reports energy consumption in kilowatt-hours for both implementations, the underlying units of measurement used for calculating energy consumption are different. For the implementation with VMs and load balancer, the unit of measurement includes CPU usage, Virtual CPU hours, and the number of API calls. For the implementation with applied tactic, the unit of measurement is solely CPU usage. As a result, the energy calculations are based on distinct units of measurement, rendering the energy consumption values incomparable between the two scenarios. Due to this discrepancy, we are unable to perform any hypothesis tests on the energy consumption values, and thus cannot reject our null hypothesis based on the energy consumption data from the CCF tool. We conclude that there is no significant difference between the energy consumption of the original version or with applied tactic in our experiment.

3) Reflection: Company B chose a different approach for application and tactic selection compared to Company A. This involved starting with potential tactics followed by identifying current areas for improvement. We define this as a tactic-driven workflow. This approach leverages the company's existing knowledge regarding a set of tactics, which are subsequently applied to specific problems. In contrast, the other company first identified hotspots followed by exploring solutions, *i.e.,* tactics to overcome the identified problem. We define this as a hotspot-driven approach.

Although we were unable to statistically prove the impact of this tactic, we observed increased energy efficiency in the data analysis process. This finding suggests potential areas for further exploration, especially considering Microsoft's commitment to improve the transparency of its cloud environment in terms of service metrics. Furthermore, as the CCF tool continues to develop, it could provide more comprehensive data for future investigations in the field of energy efficiency in cloud-based software applications.

Tactic learning (T10): Cloud-native API gateway may have an impact on the energy consumption. For empirical proof, however, transparent metrics from the cloud providers are necessary.

Workflow learning: Practitioners and researchers can adopt two alternative workflows for utilizing energy efficiency tactics: *tactic-driven* - originating from tactics to problems; and *hotspot-driven* - originating from problems to tactics.

Experiment #5: "Compress infrequently accessed data"

1) Application and Tactic Selection:

Application. This last experiment was also conducted in Company B. Compared to Company A, where the AWS Well-Architected framework was considered as source for identifying energy hotspots, this company offers an in-house developed power-BI-based dashboard for monitoring the footprint of their software applications. The dashboard collects the energy consumption in kWh and carbon emissions in MtCO2e for individual components, such as VMs or SQL databases.

As for the other experiments, architecture documentation and informal interviews complement the assessment of potential applications leading to an audit-log-application (in the remaining called Audit-App). The Audit-App is integrated into multiple systems to keep track of various information such as login attempts, file accesses, configuration changes, and other

relevant events, thus relying heavily on database transactions[4].

Tactic. When analyzing the application and considering the catalog of tactics, it was found that the Audit-App has already implemented some software tactics, but has not yet considered data compression given the high data load. The most fitting tactic would be to *Compress infrequently accessed data (T14)*. However, during the exploratory interview, it turned out that the type of application does not contain infrequently accessed data. Furthermore, manual compression using custom algorithms in the code did not seem feasible as it is not scalable to implement in all different applications. For these reasons, we investigated the data compression techniques proposed in the Azure SQL environment and ended up with the tactic "Consider row and page compression" [38].

The compression of information within a database system is an attractive option due to (i) reduction in storage requirements, and (ii) enhancement of system performance. The advantage of storage-saving can be considered as direct impact, while the performance enhancement results from the reduced space needed to relocate data during database operations physically [39]. Our experiment focuses on three compression techniques:

- *None* – no compression (original)
- *Row* – utilizing Row compression [40] (applied tactic)
- *Page* – utilizing Page compression [41] (applied tactic)

It should be noted, that Page compression builds upon the implementation of Row compression [42].

2) Experiment Execution: Following the same approach as for the other experiments, a controlled environment was created to mimic the Audit-App. The application is a Restful API application written in Java using Spring Boot framework providing two "GET" and one "POST" API endpoints. The Azure SQL database is prepared in three different scenarios according to the three compression types (none, row, page).

Initially, various workloads were examined to identify the right workload to represent the real-world scenario. It was discovered that the 100 user workload is best suited for experimentation since it produces 25,000 requests in one hour of testing. This workload represents requests received by the actual application in the real-world scenario. Additionally, 200 users is determined to investigate the difference that it can influence the dependent variables if we double the user workload.

As can be observed in Figure 6, the energy consumption for the baseline displays roughly the same size across all three compression techniques. This implies that in the workload of 0 users, the impact of the compression technique does not substantially influence the average energy consumption. All three compression techniques show a rise in energy consumption as the workload increases. Among the three compression techniques, none compression demonstrates the highest average energy consumption in the workload of 200 users. This implies that energy consumption is lower when a compression technique is applied.

[4]Due to a non-disclosure agreement, further insights about this application can not be provided.

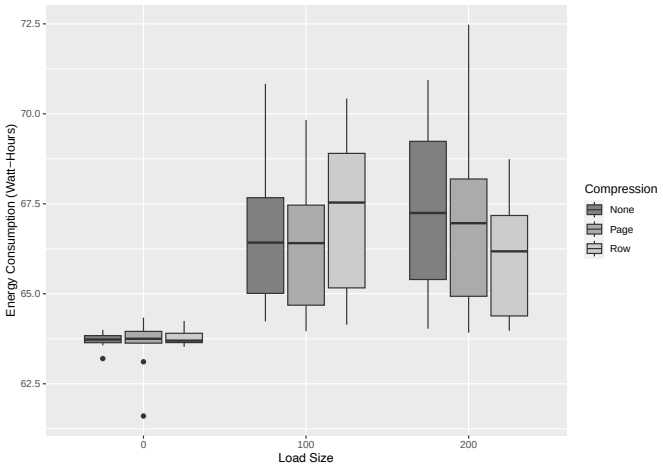

Fig. 6. Energy consumption (Watt-Hour) Original (*None*) and With (*Page* and *Row*) compression tactic (T14) across load sizes (users)

The hypothesis tests for both workloads with the parametric one-way ANOVA test imply that we fail to reject our null hypothesis as both p-values are greater than the significant level of 0.05 (workload 100: $p = 0.547$; workload 200: $p = 0.239$). This implies that we do not have strong enough evidence to suggest that the different compression types lead to a statistically significant difference in energy consumption.

3) Reflection: Interestingly, in this use case the practitioners initially identified a tactic from the catalog based on the needs of the identified hotspot (hotspot-driven). However, to seamlessly utilize the tactic into the specific context at hand, substantial modifications were necessary, resulting in the formulation of a novel tactic tailored to the application's unique requirements.

Row and page compression are the data compression techniques that are available in Azure and are easy to implement. However, in our experiment we do not have enough evidence that the implementation of row and page compression techniques significantly impacts a cloud application's energy consumption. It should be noted that another part of this experiment (not discussed in this present work) showed that the implementation of data compression techniques has a trend toward increasing the CPU usage of the cloud application and is also statistically significant based on hypothesis testing. This might lead to the conclusion that the nature of database-level compression focuses primarily on reducing storage needs rather than reducing data transmission.

Tactic learning (T14): Implementation of data compression techniques does not appear to affect energy consumption or response time significantly.
Workflow learning: Tactics may require adaptation to suit the specific context and meet its requirements effectively. To this aim, the characteristics of the context should be made explicit [43].

V. Discussion

For readability, each of the five experiments in Section IV reflects on and discusses its lessons learned individually, resulting in five takeaways for both the tactic itself and the individual workflow. The variety in how the tactics were selected and have been applied underlines that the workflow is highly dependent on the industrial context. Only by collaborating with experts from the specific company, one can derive meaningful results. In addition to the individual reflections, we provide overarching takeaway messages from this research:

A focus on a single QA always requires trade-offs. While our experiments mainly focus on applying tactics pertaining one single sustainability dimension (*i.e.,* energy efficiency), they often brought reflection on the trade-offs with other QAs (*e.g.,* performance) or extra-functional requirements (*e.g.,* cost). This points to the natural embedding of energy-efficiency in the broader multi-dimensional sustainability context (*e.g.,* environmental vs. economic or technical sustainability, in the examples above). This confirms the findings from Funke and Lago [19], namely, that architecture design often primary focuses on a single dimension, while it implicitly requires trade-offs with multiple ones.

Simulating the utilization of tactics could facilitate early adoption in practice. Our experiments show how challenging it is to apply tactics in software applications that are in a production environment. Our partners suggested that a test or simulation environment to estimate the potential impact of energy efficiency tactics would greatly help their adoption. Our ongoing research is investigating these options.

Based on the results and conclusions derived, we can now answer our RQs:

RQ1: Impact of tactics on energy efficiency. The experiments showed improvements in energy consumption of the cloud-based software ranging from 21% up to 99% across the different tactics and load sizes. While three tactics (T13, T15, T05) showed significant improvements, two (T10, T14) showed improvements without statistical significance. Implementing tactics like T13 may require significant architectural changes and should thus be addressed during system design. Conversely, tactics such as T15, T05, T14 can be implemented by adjusting cloud configurations. Despite some tactics showing minimal statistical significance, particularly those that can be implemented with ease should be standard practice in cloud contexts. Even small impacts have the potential to accumulate exponentially [16].

RQ2: Tactic identification and utilization in industry. By analyzing practitioners' workflows, *i.e.,* how they selected the application and tactics, we gained multiple lessons learned and general takeaways. These may support practitioners in applying and/or evaluating other tactics in the future and could help raise awareness of considering tactics in the first place. We uncovered two unique workflows for selecting tactics in industry. First, a *tactic-driven approach* can potentially accelerate finding and applying tactics, but it may also be limited due to the options offered by the available catalogue. In contrast, a *hotspot-driven approach* can address high-impact energy hotspots, but it may overlook others. To start with, practitioners could consider the open archive of *Awesome and Dark Tactics* [30] for the tactic-driven, and the energy-monitor *Scaphandre* [44] for the hotspot-driven strategy. Future work should empirically evaluate possible trade-offs and operational principles between the two workflows.

VI. Threats to Validity

We discuss the threats to validity according to Cook and Campbell's categorization [45]. For potential threats to validity related to each individual experiment (*e.g., Conclusion Validity* concerning statistical correctness and significance), we refer to specific reports in the online replication package [27].

External Validity. To ensure that our experiments are representative for the complexity and dynamics of the real-world cloud-based environments, industrial experts have been involved at any time of the experiment design. The behavior and architecture of the real-world cloud-based software was replicated based on the architecture documentation and by conducting informal interviews with the experts to ensure accurate representation.

Although the characteristics, architecture, and utilization patterns of our case studies may not be representative of all cloud-based applications, challenging generalizability, we mimicked real-world applications from three independent applications in two different companies to increase diversity in the subject selection. In addition, our findings help (re)evaluate the same or new energy efficiency tactics, hence contributing to the larger goal of evaluating a broad range of tactics in different contexts.

Internal Validity. We relied on practitioners which were not randomly selected, but based on availability and the researcher's own network. This selection of practitioners could have led to different applications and tactics. Nevertheless, each case and experiment was accompanied by multiple practitioners leading to mutual knowledge exchange and data triangulation [46].

VII. Conclusion

This paper presented five experiments evaluating energy efficiency tactics for cloud-based software in industrial settings. The industrial context helped us to derive relevant results for practitioners and gain insights into the workflow on how tactics can be selected and applied. Each experiment defined its own experiment variables, computed their statistical analyses, and derived conclusions based on the experiment results and observed workflow. Three tactics showed a significant decrease in energy consumption of cloud-based software, while two showed improved efficiency, although without statistical significance. The lessons learned underscore the importance of research collaborations with industrial professionals. We hope for practitioners to use our results improving the sustainability of their cloud-based software, and for researchers to continue joining forces with practitioners to empirically assess further energy efficiency tactics.

References

[1] H. Saini, A. Upadhyaya, and M. K. Khandelwal, "Benefits of Cloud Computing for Business Enterprises: A Review," *SSRN Electronic Journal*, 2019.

[2] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *2009 International Conference on High Performance Computing & Simulation*. Leipzig, Germany: IEEE, Jun. 2009, pp. 1–11.

[3] L. Peter, D. Paul, P. Pavel, and D. Kishore, "The green behind the cloud," Accenture, Tech. Rep., 2020.

[4] M. P. Mills, "The cloud begins with coal," Digital Power Group, Tech. Rep., 2013.

[5] B. Knowles, "ACM TechBrief: Computing and Climate Change," ACM, Tech. Rep., Nov. 2021.

[6] A. Bessin, F. de Jong, P. Lago, and O. Widerberg, "News from Europe's Digital Gateway: A Proof of Concept for Mapping Data Centre News Coverage," in *EnviroInfo*, Munich, Germany, 2023.

[7] Amazon Web Services, "Sustainability in the cloud," 2023, [Online]. https://sustainability.aboutamazon.com/environment/the-cloud (accessed: 2023-12-01).

[8] Microsoft Azure, "Azure sustainability," 2023, [Online]. https://azure.microsoft.com/en-us/explore/global-infrastructure/sustainability/carbon-benefits (accessed: 2023-12-01).

[9] Google Cloud Platform, "Cloud sustainability," 2023, [Online]. https://cloud.google.com/sustainability (accessed: 2023-12-01).

[10] 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, Feb. 2020.

[11] J. Gao, H. Wang, and H. Shen, "Smartly Handling Renewable Energy Instability in Supporting A Cloud Datacenter," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. New Orleans, LA, USA: IEEE, May 2020, pp. 769–778.

[12] S. Vos, P. Lago, R. Verdecchia, and I. Heitlager, "Architectural Tactics to Optimize Software for Energy Efficiency in the Public Cloud," in *ICT4S*. Plovdiv, Bulgaria: IEEE, Jun. 2022, pp. 77–87.

[13] G. Procaccianti, P. Lago, and G. A. Lewis, "Green Architectural Tactics for the Cloud," in *2014 IEEE/IFIP Conference on Software Architecture*. Sydney, Australia: IEEE, Apr. 2014, pp. 41–44.

[14] C. Paradis, R. Kazman, and D. A. Tamburri, "Architectural Tactics for Energy Efficiency: Review of the Literature and Research Roadmap," in *Hawaii International Conference on System Sciences*, 2021.

[15] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, fourth edition ed., ser. Sei Series in Software Engineering. Boston: Addison-Wesley, 2021.

[16] J. Balanza-Martinez, P. Lago, and R. Verdecchia, "Tactics for Software Energy Efficiency: A Review," in *Advances and New Trends in Environmental Informatics*, Munich, Germany, 2023.

[17] G. Márquez, H. Astudillo, and R. Kazman, "Architectural tactics in software architecture: A systematic mapping study," *Journal of Systems and Software*, vol. 197, p. 111558, Mar. 2023.

[18] P. Lago, D. Greefhorst, and E. Woods, "Architecting for Sustainability," in *EnviroInfo*, 2022, pp. 200–210.

[19] M. Funke and P. Lago, "Carving Sustainability into Architecture Knowledge Practice," in *Software Architecture*, ser. Lecture Notes in Computer Science, vol. 14212. Cham: Springer Nature Switzerland, 2023, pp. 54–69.

[20] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings. 26th International Conference on Software Engineering*. IEEE, 2004, pp. 273–281.

[21] F. Khomh and S. A. Abtahizadeh, "Understanding the impact of cloud patterns on performance and energy consumption," *Journal of Systems and Software*, vol. 141, pp. 151–170, Jul. 2018.

[22] B. Bani, F. Khomh, and Y.-G. Guéhéneuc, "A Study of the Energy Consumption of Databases and Cloud Patterns," in *Service-Oriented Computing*, Q. Z. Sheng, E. Stroulia, S. Tata, and S. Bhiri, Eds. Cham: Springer International Publishing, 2016, vol. 9936, pp. 606–614.

[23] C. Sahin, F. Cayci, I. L. M. Gutierrez, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Initial explorations on design pattern energy usage," in *2012 First International Workshop on Green and Sustainable Software (GREENS)*. Zurich, Switzerland: IEEE, Jun. 2012, pp. 55–61.

[24] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? A model and annotation," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1735–1758, Oct. 2010.

[25] R. Kazman, S. Haziyev, A. Yakuba, and D. A. Tamburri, "Managing Energy Consumption as an Architectural Quality Attribute," *IEEE Software*, vol. 35, no. 5, pp. 102–107, Sep. 2018.

[26] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[27] M. Funke, P. Lago, E. Adenekan, I. Malavolta, and I. Heitlager, "Replication Package of this Study - 'Experimental Evaluation of Energy Efficiency Tactics in Industry: Results and Lessons Learned'," Feb. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10649481

[28] "Locust.io," 2023, [Online]. https://locust.io (accessed: 2023-12-01).

[29] Thoughtworks Inc., "Cloud Carbon Footprint," 2023, [Online]. https://www.cloudcarbonfootprint.org/ (accessed: 2023-12-01).

[30] P. Lago, "An Archive of Awesome and Dark Tactics," 2024, [Online]. https://s2group.cs.vu.nl/AwesomeAndDarkTactics (accessed: 2023-12-01).

[31] G. Procaccianti, P. Lago, A. Vetro, D. M. Fernandez, and R. Wieringa, "The Green Lab: Experimentation in Software Energy Efficiency," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Florence, Italy: IEEE, May 2015, pp. 941–942.

[32] Amazon Web Services, "AWS Well-Architected Framework," 2023, [Online]. https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html (accessed: 2023-12-01).

[33] "Nginx Proxy Manager," 2023, [Online]. https://nginxproxymanager.com (accessed: 2023-12-01).

[34] Amazon Web Services, "AWS Lambda - Serverless Function," 2023, [Online]. https://aws.amazon.com/lambda (accessed: 2023-12-01).

[35] ——, "APM Tool - Amazon CloudWatch," 2023, [Online]. https://aws.amazon.com/cloudwatch (accessed: 2023-12-01).

[36] Pallets, "Flask Documentation," 2023, [Online]. https://flask.palletsprojects.com (accessed: 2023-12-01).

[37] Broadcom Inc., "Spring Boot," https://spring.io/projects/spring-boot, 2023.

[38] Microsoft, "Data compression," 2023, [Online]. https://learn.microsoft.com/en-us/sql/relational-databases/data-compression/data-compression (accessed: 2023-12-01).

[39] G. V. Cormack, "Data compression on a database system," *Communications of the ACM*, vol. 28, no. 12, pp. 1336–1342, Dec. 1985.

[40] Microsoft, "Row compression implementation," 2023, [Online]. https://learn.microsoft.com/en-us/sql/relational-databases/data-compression/row-compression-implementation (accessed: 2023-12-01).

[41] ——, "Page compression implementation," 2023, [Online]. https://learn.microsoft.com/en-us/sql/relational-databases/data-compression/page-compression-implementation (accessed: 2023-12-01).

[42] P. A. Carter, "Table optimizations," in *Pro SQL Server 2022 Administration: A Guide for the Modern DBA*. Berkeley, CA: Apress, 2023, pp. 219–261.

[43] A. Bedjeti, P. Lago, G. A. Lewis, R. D. De Boer, and R. Hilliard, "Modeling Context with an Architecture Viewpoint," in *2017 IEEE International Conference on Software Architecture (ICSA)*. Gothenburg: IEEE, Apr. 2017, pp. 117–120.

[44] Hubblo Org., "Scaphandre - energy consumption metrology agent," 2024, [Online]. https://github.com/hubblo-org/scaphandre (accessed: 2024-02-16).

[45] T. D. Cook, D. T. Campbell, and A. Day, *Quasi-Experimentation: Design & Analysis Issues for Field Settings*. Houghton Mifflin Boston, 1979, vol. 351.

[46] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering - An International Journal*, vol. 14, no. 2, pp. 131–164, Apr. 2009.