Architectural Tactics to Optimize Software for Energy Efficiency in the Public Cloud

Sophie Vos
Vrije Universiteit Amsterdam
The Netherlands
s.o.vos@outlook.com

Patricia Lago
Vrije Universiteit Amsterdam
The Netherlands
p.lago@vu.nl

Roberto Verdecchia
Vrije Universiteit Amsterdam
The Netherlands
r.verdecchia@vu.nl

Ilja Heitlager
Schuberg Philis
The Netherlands
iheitlager@schubergphilis.com

***Abstract*—A promise of cloud computing is the reduction of energy footprint enabled by economies of scale. Unfortunately, little research is available on how cloud consumers can reduce their energy footprint when running software in the public cloud. Moreover, cloud consumers do not have full access to information regarding their cloud infrastructure usage, which is required to understand the impact of design decisions on energy usage. The purpose of our study is to support cloud consumers in developing energy-efficient workloads in the public cloud. To achieve our goal, we collaborated with a large cloud solution provider to discover an initial set of reusable architectural tactics for software energy efficiency. Starting from interviews with 17 practitioners, we reviewed and selected available tactics to improve the energy efficiency of individual workloads in the public cloud, and synthetized the identified tactics in a reusable model. In addition, we conducted a case study to assess the impact of utilizing a tactic, which was selected following a prioritization provided by the practitioners. Our results demonstrate the possibility to architect cloud workloads for energy efficiency through reasoning and estimation of resource optimization. However, the process is not (yet) straightforward due to the current lack of transparency of cloud providers.**

***Index Terms*—Software Architecture, Tactics, Energy Efficiency, Public Cloud, Edge Computing**

I. Introduction

An increasing number of organizations migrate their ICT infrastructures to the public cloud. Cloud computing involves the shift from performing computation and data storage on-premise to a geographically remote Data Center (DC). The global cloud computing market was worth over 300 billion dollars in 2020 and is predicted to reach 832 billion dollars by 2025 [1].

An important benefit of migrating software to the public cloud is shifting the responsibility of maintaining and operating hardware resources to cloud providers. Moreover, migrating ICT workloads to the public cloud is expected to reduce the energy footprint of cloud consumers. Research from Accenture shows that shifting from on-premise DCs to the public cloud can reduce an enterprise's energy usage by 65% and cut carbon emissions by more than 84% [2]. This is due to efficient hardware, economies of scale, and resource sharing.

Nevertheless, the gain in energy reduction that cloud providers offer can be overshadowed by the rapid growth of highly accessible and available ICT services. The energy footprint of the ICT sector grows exponentially and significantly contributes to the global Greenhouse Gas Emissions (GHGEs). Belkhir et al. [3] estimated that the global GHGEs of the ICT sector relative to the worldwide footprint doubled from 1-1.6% in 2007 to 3–3.6% in 2020. If this trend continues, the share of ICT will increase to 14% of the total global GHGEs by 2040. Currently, DCs have with 45% the largest energy footprint within the ICT sector [3].

To reach global climate goals, the energy consumption of hyperscale cloud DCs needs to be reduced, *e.g.,* by utilizing Green Software techniques [4]. By quoting Amazon Web Services (AWS), ICT energy reduction is a shared responsibility between cloud providers and consumers [5]: cloud providers are responsible for optimizing the sustainability **of** the cloud (*e.g.,* by delivering efficient infrastructure), whereas cloud consumers are responsible for sustainability **in** the cloud (*e.g.,* by optimizing workloads).

Numerous researches have been conducted to increase the energy efficiency from the perspective of cloud providers. Nevertheless, as of today, no peer-reviewed research has been conducted to increase the energy efficiency in the public cloud from the perspective of cloud consumers. In this research we address this gap by focusing on the research question (RQ): *"How can cloud consumers architect energy-efficient software in the public cloud?"*. Consumer-centric energy optimization in the public cloud is a challenging topic, as the cloud forms an opaque abstraction between cloud consumers and the actual energy-hungry DCs. To tackle our RQ, we collaborate with SBP, an ICT consultancy company focusing, among others, on cloud migration and digital transformation. We opted to collaborate with professionals from a ICT firm experienced in optimizing software infrastructure in the cloud. This collaboration allowed us to gain access to valuable industrial experience, which we used to identify tactics for resource optimization and energy efficiency in the cloud.

The main contributions of this study are:

- A set of reusable tactics to optimize energy efficiency in the public cloud (Section IV);
- A review of methods to monitor the energy consumption of individual workloads in the public cloud (Section V);
- A case study to assess the impact on energy efficiency of one of the tactics discovered, namely "Apply edge computing" (Section VI);

- A proposed definition of energy efficiency in the public cloud (Section VII).

II. Related Work

Numerous researches on improving the energy efficiency of software and hardware in hyperscale DCs have been published throughout the years [6], [7]. Nevertheless, these studies assume full control over the cloud infrastructure, and can hence only be applied by cloud providers. In contrast, we did not identify any peer-reviewed literature focusing on the perspective of cloud consumers, which is instead adopted in this research.

A set of studies closely related to our investigation reflect on the integration of energy efficiency aspects while architecting software-intensive systems. Kazman et al. [8] performed a case study exploring the management of energy efficiency as an architectural quality attribute, concluding that energy efficiency should be considered on par with other quality attributes, *e.g.,* availability. These findings serve as a foundation to our study.

Similarly, other related studies focus on identifying architectural tactics to improve energy efficiency. In contrast to such studies, which discovered tactics based on literature reviews or *in vitro* experimentation, in this investigation we base our findings directly on industrial knowledge. In addition, we explicitly focus on identifying practical and actionable tactics, while the tactics elicited in the related studies result to lie at a higher level of abstraction. As an example, Procaccianti et al. [9] discovered tactics for software energy efficiency in the cloud through a systematic literature review. Procaccianti et al. defined three categories to classify tactics: (1) *Energy Monitoring*, *i.e.,* tactics to gather and presenting energy consumption information; (2) *Self-Adaptation*, *i.e.,* tactics implementing mechanisms to modify runtime software configurations to improve energy efficiency; and (3) *Cloud Federation*, *i.e.,* tactics allowing cloud-based software systems to select cloud services based on energy consumption information.

The study of Procaccianti et al. [9] offers a useful categorization of tactics for energy efficiency which is later modified and extended by Paradis et al. [10]. Paradis et al. performed a taxonomic literature review to discover architectural tactics for energy efficiency. The authors classify 10 tactics providing a basis for architectural design and analysis focusing on energy efficiency. The tactics are organized into the categories *Resource Monitoring*, *Resource Allocation*, and *Resource Adaptation*. In this study, we extend this categorization of tactics. Therefore, in the remainder of this section, we focus on documenting the categories identified by Paradis et al., by additionally describing tactics constituting such categories.

Tactics falling into the category *Resource Monitoring* are *Metering*, *Static Classification*, and *Dynamic Classification* [10]. The tactic *Metering* concerns real-time data collection of the energy consumption *via* sensors. When cloud users do not have access to real-time data reflecting the energy consumption (*e.g.,* in a public cloud), a *Static Classification* of the resources is necessary to drive decision making. *Dynamic Classification* is positioned in between the previous two tactics, it can be applied when real-time data collection is infeasible, but the estimation can be based on transient conditions (*e.g.,* workload).

The *Resource Allocation* category contains the tactics *Vertical Scaling*, *Horizontal Scaling*, *Scheduling*, and *Brokering* [10]. *Vertical Scaling* concerns adding or activating resources (*e.g.,* CPU or RAM) to meet processing requirements. In the context of energy efficiency, Vertical Scaling is utilized to deactivate or remove computing resources to save energy. A related tactic is *Horizontal Scaling*, which involves adding resources (*e.g.,* servers or VMs) to the currently available set of resources. A frequently adopted example of horizontal scaling tactic is VM consolidation, *i.e.,* multiple VMs are consolidated onto the same physical server to save energy. The *Scheduling* tactic instead involves the allocation of tasks among available resources. This tactic aims at improving energy efficiency by scheduling tasks across machines to realize optimal vertical and horizontal scaling. Lastly, the tactic *Brokering* involves including energy-related information to services, enabling service requests to prioritize service execution based on energy efficiency information.

The final category presented presented by Paradis et al. [10] is *Resource Adaptation*. The tactics within this category are *Service Adaptation*, *Increase Efficiency*, and *Reduce Overhead*. The tactic *Service Adaptation* extends the Brokering tactic by emphasizing that services should be selected based on provided energy-related information. The tactic *Increase Efficiency* instead entails increasing the resource efficiency of software and algorithms, hence improving also their overall energy usage. Finally, the tactic *Reduce Overhead* involves reducing redundant processes. This tactic often introduces a trade-off between energy efficiency and other quality attributes.

Paradis et al. [10] stress that the current body of literature lacks cloud energy-efficiency research based on industrial insights. To fill this gap, in this study we extend the model of Paradis et al. [10], by grounding our findings in the knowledge of industrial practitioners.

III. Study design

With this study, we aim at providing concrete guidance to cloud consumers in architecting their software workload in the public cloud to optimize energy efficiency. More formally, by applying the template introduced by Basili et al. [11], the goal can be defined as follows:

Analyze *Cloud-Native Tactics*
For the purpose of *Optimizing*
With respect to *Energy Efficiency*
From point of view of *Cloud Consumers*
In the context of *Public Cloud*

In terms of research questions (RQs), the main RQ of this study is: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. In order to answer our RQ, we adopt a four-phased research process, with each phase associated with a specific sub-RQ. An overview of the research process is illustrated in Figure 1 and is further detailed below.

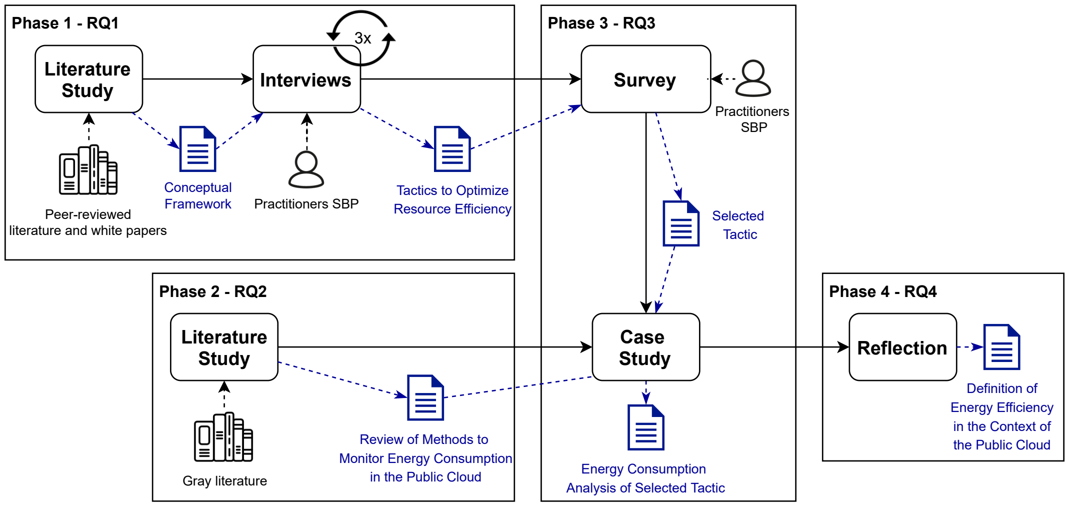
Phase 1 - RQ1
Literature Study
Interviews
3x
Peer-reviewed literature and white papers
Conceptual Framework
Practitioners SBP
Tactics to Optimize Resource Efficiency
Phase 3 - RQ3
Survey
Practitioners SBP
Selected Tactic
Phase 2 - RQ2
Literature Study
Gray literature
Review of Methods to Monitor Energy Consumption in the Public Cloud
Case Study
Energy Consumption Analysis of Selected Tactic
Phase 4 - RQ4
Reflection
Definition of Energy Efficiency in the Context of the Public Cloud

Fig. 1. Research process overview

Phase 1 aims to answer the first sub-RQ, namely *"Which cloud-native tactics are available to optimize the energy efficiency of cloud workloads?"*. We start by inspecting related literature presenting tactics for energy efficiency (see Section II). From this inspection, we extract the most representative single source, namely, the work from Paradis et al. [10]. Next, we conduct 3 iterations of interviews to extend the tactics from the literature. In the first iteration, we interview 17 practitioners (selected based on experience in the software industry with diverse roles such as developer, architect, operation officer, cloud vendor relation manager) at SBP to define an initial set of tactics for resource optimization. A replication package containing the interview questions and an overview of the participants is made available online[1]. Next, we map the tactics from the literature to the discovered tactics and created a model of reusable tactics. As many practitioners at SBP refers us to the Well-Architected Framework (WAF) introduced by AWS, we opt to incorporate this framework into our model as well. In the second iteration of interviews, we ask each practitioner to review the model and tactics. In the final iteration, we ask the AWS technical practice lead at SBP to verify the constructed model of tactics. This contribution is presented in Section IV.

Conducted in parallel to Phase 1, **Phase 2** aims to answer the sub-RQ: *"What methods are available to monitor the energy consumption of individual workloads in the public cloud?"*. Such method enables to measure the sustainability impact of the discovered tactics in the public cloud. Answering this sub-RQ is achieved *via* a gray literature study[12], as no peer-reviewed literature is available on the subject. We review blog posts from the computer science community and the documentation provided by the most popular cloud providers, namely, AWS, Azure, and GCP. The output of this research phase is an overview of existing methods to monitor the energy consumption of software running in the public cloud. The results of this review are presented in Section V.

Subsequently, with **Phase 3**, we aim to answer the sub-RQ: *"What is the impact of a selected tactic on energy consumption?"*. In order to answer this sub-RQ, we conduct a case study to assess the impact on energy consumption in the public cloud of a specific tactic, namely "Apply edge computing". The tactic is selected *via* a survey with practitioners at SBP, in order to identify the tactic which showcases higher potential w.r.t. energy efficiency improvement. The results of the case study are presented in Section VI.

Finally, **Phase 4** involves a reflection on the definition of energy efficiency in the context of the public cloud to answer the final sub-RQ, namely: *"How can energy efficiency be defined in the context of the public cloud?"*. This reflection is presented in Section VII.

IV. Tactics for Energy Optimization in the Public Cloud

The interviews show that energy efficiency is not well adopted as a quality attribute in the industry. Fortunately, the industry is well experienced in increasing the resource efficiency of cloud software. We therefore start from the assumption that resource efficiency leads to energy efficiency, as energy usage is correlated to resource usage, and utilizing resources more efficiently and less frequently leads hence to energy saving. Accordingly, we begin with discovering tactics for resource efficiency in the public cloud and from there derive potential tactics for energy efficiency. We mainly focus on AWS as SBP utilized this platform in numerous projects, and AWS is currently the market-leading cloud provider [13].

A. A Model of Reusable Tactics for Energy Efficiency in the Public Cloud

This section presents the constructed model of reusable tactics, by describing each of the discovered tactics. The diagram is presented in Figure 2. In the model, the identified

[1]https://github.com/S2-group/ICT4S-2022-cloud-tactics-rep-pkg

tactics are grouped following the categorization presented by Paradis et al. [10] (see Section II). Our contribution (tactics for energy efficiency discovered from SBP) are colored blue. The tactics that are related to AWS are labeled with its logo. The discovered tactics are potentially impactful to improve the energy efficiency of cloud consumers but this still needs to be (empirically) assessed. The discovered tactics are described per category, namely *resource monitoring* (Section IV-A1), *resource allocation* (Section IV-A2), and *resource adaptation* (Section IV-A3).

1) Resource monitoring

The category *resource monitoring* considers the metering and classification of cloud workloads to optimize the performance [10]. Resource monitoring does not directly affect energy reduction, however, it is a prerequisite for most tactics of the other two categories (*resource allocation* and *resource adaptation*).

T1: Experiment to discover optimal architecture. *Description.* The public cloud provides access to a wide variety of resources without the need for prior investments. Hence, it is relatively straightforward to compare cloud services of similar nature, as opposed to purchasing, installing, and operating similar components on-premise. This enables to measure and compare the performance of different resources for a specific workload, allowing a data-driven decision to select the most efficient architecture. *Relation to energy efficiency.* Based on energy-related information, developers can experiment with design decisions to discover the most energy-efficient architecture.

T2: Automatically monitor efficiency per workload. *Description.* Automatically monitoring the performance of software provides insights into its behavior and highlights the sections to focus on for optimization. *Relation to energy efficiency.* Automatic monitoring of energy consumption data is a key prerequisite for optimizing workloads for energy efficiency.

2) Resource allocation

The category *resource allocation* involves assigning tasks and workloads to the instances and resources [10]. The following 8 tactics belong to this category.

T3: Apply auto-scaling. *Description.* Auto-scaling involves horizontally and vertically scaling the resources to optimize performance and costs[14]. The application is automatically monitored and adjusted to ensure stable performance at the lowest possible cost. This enables on-demand resource usage, which is in contrast with the traditional approach of allocating extra resources, which are seldom used yet constantly available, to satisfy potential peak loads. *Relation to energy efficiency.* When applying auto-scaling, the number of resources used is adjusted based on the runtime requirements. Therefore, the energy required by resources is proportional to the actual workload, thus saving the energy required by superfluous resources.

T4: Continuously evaluate right sizing. *Description.* Right sizing is the process of matching instance types and sizes to your workload performance and capacity requirements at the lowest possible cost. Additionally, this tactic involves the identification of opportunities to downsize without compromising capacity or other requirements [15]. *Relation to energy efficiency.* From an energy perspective, it can be assessed which resources are most suitable to optimize for energy efficiency. For example, data can be stored using several different services (*e.g.,* S3 Reduced Redundancy Storage, Glacier, Tape). Where and how the data is stored, can have a considerable impact on energy consumption.

T5: Deallocate resources that are not used. *Description.* Resources that are no longer used should be deallocated. *Relation to energy efficiency.* If a certain resource is not used during a specific moment in time (*e.g.,* in the weekends), switching off the (idle) resource saves the energy required to keep the resource running.

T6: Select nearby regions with better renewable energy rates. *Description.* Regions with high renewable energy rates can be selected to reduce the energy footprint of cloud instances. *Relation to energy efficiency.* Replacing gray energy with green energy does not *per se* improve energy efficiency. However, it does decrease the carbon footprint. A trade-off between the distance of the region to the end-users and renewable energy rate should in any case be considered. Selecting a distant region that offers more renewable energy might not save carbon emissions, as more energy will be required to transport the data to the end-users.

T7: Use reserved instances. *Description.* Reserved instances are long-term subscriptions to software services at a discounted price [16]. *Relation to energy efficiency.* While reserved instances consume the same amount of energy as non-reserved instances, reserved instances increase the predictability of future energy consumption patterns. Therefore, cloud providers can plan resource provisioning, lowering the number of idle resources which may be needed with changing runtime requirements.

T8: Use spot instances. *Description.* AWS EC2 spot instances allow access to spare EC2 capacity [17]. These instances are offered for a discounted price. The catch is that these instances are only offered if there are available resources and can be retracted at a two-minute notice. Hence, they are suitable for fault-tolerant, stateless applications. *Relation to energy efficiency.* If additional resources are unexpectedly needed, use spot instances first, as they are more sustainable, and could work as a quick fix for temporary high loads which will soon go down.

T9: Perform specialized tasks that occur infrequently in the cloud. *Description.* Specialized tasks that occur infrequently might need specialized hardware (*e.g.,*, AI accelerators). To benefit from economies of scale, it is more efficient to share specialized hardware among multiple consumers. *Relation to energy efficiency.* A consumer who purchases hardware that is infrequently used and otherwise runs idle has a negative effect on energy efficiency. If multiple consumers share the hardware in the cloud, the hardware will be more efficiently used and, therefore, is expected to have a positive effect on the energy efficiency.

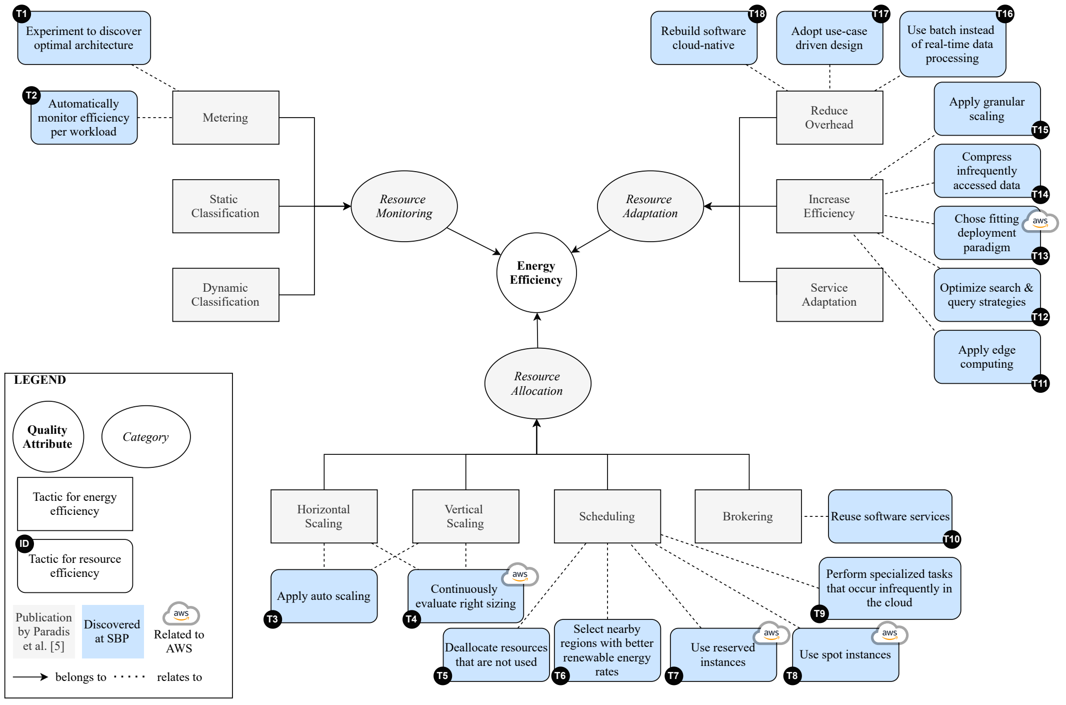
T1 Experiment to discover optimal architecture
T2 Automatically monitor efficiency per workload
Metering
Static Classification
Dynamic Classification
Resource Monitoring
Energy Efficiency
Resource Adaptation
Resource Allocation
T18 Rebuild software cloud-native
T17 Adopt use-case driven design
T16 Use batch instead of real-time data processing
Reduce Overhead
Increase Efficiency
Service Adaptation
T15 Apply granular scaling
T14 Compress infrequently accessed data
T13 Chose fitting deployment paradigm
T12 Optimize search & query strategies
T11 Apply edge computing
Horizontal Scaling
Vertical Scaling
Scheduling
Brokering
T10 Reuse software services
T3 Apply auto scaling
T4 Continuously evaluate right sizing
T5 Deallocate resources that are not used
T6 Select nearby regions with better renewable energy rates
T7 Use reserved instances
T8 Use spot instances
T9 Perform specialized tasks that occur infrequently in the cloud
aws
LEGEND
Quality Attribute
Category
Tactic for energy efficiency
ID Tactic for resource efficiency
Publication by Paradis et al. [5]
Discovered at SBP
Related to AWS
belongs to
relates to

Fig. 2. Model of reusable tactics

T10: Reuse software services. *Description.* Using services made available by cloud providers instead of implementing all functionalities from scratch is expected to increase the resource efficiency as the services are native to the infrastructure and developed by specialists. *Relation to energy efficiency.* Software services directly supplied by cloud providers are expected to be engineered and fine-tuned to best fit the cloud-native environment they are deployed in. Hence, such services are expected to be more efficient than *ad-hoc* services implemented independently by consumers.

3) Resource adaptation

The category *resource adaptation* involves changing software and hardware resources to increase the efficiency [10].

T11: Apply edge computing. *Description.* Moving computing resources closer to users decreases network traffic. Furthermore, the system can be designed in a way that only processed/aggregated data need to be transported which reduces the amount of data traffic. For workloads that involve the generation of high volumes of data (*e.g.,* IoT sensors), it is shown that performing all computation in the cloud can be inefficient. This is due to the gap between the relatively high processing speed in the cloud and the limited bandwidth capacity to transfer the high amount of data throughput [18]. *Relation to energy efficiency.* Transporting less data over the network is expected to reduce the energy consumption.

T12: Optimize data search & query strategies. *Description.* Data search and query strategies can be optimized to increase their efficiency. For example, when using (a query service to analyze data in Amazon S3) consumers are charged for the amount of data being scanned to retrieve query results [19], [20]. Hence, to optimize the computing resources of a query, developers need to ensure that a minimum amount of data is scanned. *Relation to energy efficiency.* Processing less data results in less computing power and memory usage, hence reducing the associated energy consumption.

T13: Choose fitting deployment paradigm. *Description.* Among the most popularly adopted deployment paradigms in the cloud are VMs, containers, and serverless architectures. Choosing the fitting paradigm for the workload will optimize the performance. *Relation to energy efficiency.* A deployment paradigm strategy can optimize resource usage in the cloud. The choice of deployment paradigm is dependent on the cloud workload. For example, VMs work well with a stable, predictable workload whereas serverless architectures are suitable for bursty workloads.

T14: Compress infrequently accessed data. *Description.* Infrequently used data should be compressed to optimize the storage costs. In contrast, data that is more frequently accessed should not be compressed, as the energy required to compress and extract the data might outweigh the energy saved by storing a smaller volume of data. Understanding the exact threshold to compress the data depends on the underlying hardware and should be identified *via* appropriate data usage monitoring. *Relation to energy efficiency.* Whenever less data is stored, less energy is used for storage. A trade-off that needs to be considered is the amount of energy that is required to (de)compress the data.

T15: Apply granular scaling. *Description.* Granular scaling involves breaking down a workload into smaller components. Accordingly, the used resources can be scaled down into smaller chunks, hence allowing a better match between the physical resources and the workload. *Relation to energy efficiency.* Granular scaling allows a precise match between physical hardware and workloads. For example, if a workload consists of two components with the same specification and the resource utilization is 75%, both components are required to run the workload. In contrast, when the workload consists of four smaller components, one component can be switched off to facilitate the 75% resource utilization.

T16: Use batch instead of real-time data processing. *Description.* If software product objectives allow, transporting data in batches rather than in real-time can optimize resource usage, as there is less overhead. *Relation to energy efficiency.* Sending data in batches can reduce energy consumption, as less data needs to be transmitted over the network due to reduced network overhead. Appropriate experimentation is required to measure whether the reduction in overhead has a significant effect on energy efficiency.

T17: Adopt use-case driven design. *Description.* As emerging from the interviews, frequently not all deployed ICT services create direct business value. Use-case driven design is an approach enabling to detect redundant software and data storage, and hence allowing to turn off or eliminate unutilized software services. *Relation to energy efficiency.* By eliminating redundant software services and data storage, the costs and energy associated to keep them running can be saved.

T18: Rebuild software cloud-native. *Description.* Instead of migrating existing software, rewriting cloud-native software products from scratch can increase the efficiency, as the system becomes more efficient and optimally uses the underlying infrastructure [21]. *Relation to energy efficiency.* Cloud-native applications are expected to be designed to make best use of the cloud paradigm and underlying data center hardware. In addition, cloud provider optimizations, *e.g.,* VM consolidation, are expected to improve the overall energy consumption of cloud-native applications.

V. Methods to Monitor Energy Consumption in the Public Cloud

The energy consumption of individual cloud workloads needs to be monitored in order to assess the impact of the discovered tactics on the energy efficiency. As concepts related to software sustainability are gaining traction, in the last year (2021) a laudable "race" towards providing cloud sustainability monitoring services started among cloud providers. In this section, we discuss the available possibilities to monitor the energy footprint provided by the major cloud providers (Section V-A), while Section V-B reports a generic open-source monitoring solution that can be used across multiple cloud providers.

A. Monitor energy footprint in AWS, Azure, and GCP

Microsoft Azure offers the *Microsoft Emissions Impact Dashboard*. This dashboard allows cloud consumers to trace carbon emissions related to cloud service usage [22]. An advantage of the dashboard is that it allows users to categorize their carbon emissions per geographic location and cloud service. Another advantage is that the dashboard includes Scope 1, 2, and 3 emissions [23]. Scope 3 emissions entail indirect emissions from external vendors and suppliers. Hence, users have a complete overview of all hidden emissions. Moreover, the methodology to calculate the emissions is validated by Stanford University and adheres to ISO standards for measuring GHGEs. A disadvantage of the dashboard is that a Power BI Pro subscription is required [24]. Hence, costs and a setup process are associated with the tool. Another disadvantage is that the tool only tracks the overall emissions associated with the workload. This does not allow small-scale experiments to reduce the energy consumption of individual workloads.

Google Cloud Platform (GCP) allows users to trace their carbon footprints using the *Carbon Footprint Tool* [25]. This tool provides users with insights into gross carbon emissions that are associated with their GCP usage. The tool is free of charge and available directly in the Cloud Console. The carbon footprint is calculated based on the electricity usage of the used cloud services together with the carbon intensity of the region in which the resources are located. This methodology has as implication that the reported emissions do not include Scope 3 emissions. This entails that using these emissions for reporting results in incomplete statistics. On the other hand, an advantage of this approach is that the emissions are directly related to the electricity usage of cloud services. Accordingly, cloud consumers are tracing a metric that they can impact directly.

Among prominent cloud providers, when it comes to sustainability monitoring, AWS resulted to be slightly behind its competitors. At the 2021 re:Invent Conference AWS announced their effort in developing the *AWS Customer Carbon Footprint Tool*, a tool allowing cloud consumers to track their carbon footprint [26]. The tool is expected to be released in the second quarter of 2022.

B. Cloud Carbon Footprint Tool

An open-source tool to estimate energy consumption in the public cloud for AWS, Azure, and GCP is the Cloud Carbon Footprint (CCF) tool [27]. CCF allows the estimation of emitted CO_2 and, in contrast to the dashboards described in Section V-A, the consumed energy of cloud instances. The CCF takes as input usage data (computation, storage, and networking loads) reported by cloud providers and estimates the energy consumption [Wh] by building upon the conversion factors introduced by the company Etsy [28]. The estimated energy is then multiplied by the PUE of cloud providers, to account for the supporting equipment. Finally, the carbon intensity of the region where the DC is located is considered to calculate carbon emissions. An advantage of the tool is the ability to trace both energy usage [Wh] and carbon footprint. In addition, the tool enables tracking workloads of multiple cloud providers in a single unified dashboard. As main limitation, both energy consumption and carbon footprint are based on estimations, rather than direct measurements.

VI. Case Study Impact Of Tactic "Apply edge computing" on Energy Consumption

After discovering tactics for energy efficiency (Section IV) and establishing a method to estimate energy consumption in the cloud (Section V), we demonstrate the applicability of one of our tactics and gain further hands-on insights on energy efficiency optimization *via* architectural tactics in the public cloud. The tactic used for this exemplary case study, selected in collaboration with the practitioners at SBP, was "Apply edge computing" (see T11, Section IV-A3). The scope of the example entails a comparison between full cloud adoption versus edge computing for one software application. The tactic is selected by considering implementation ease and envisioned energy saving.

A. Case study description

As case study, we consider a real-life software project of SBP, which focuses on image-based fall detection. Based on a sensor (*e.g.,* a camera) an environment (*e.g.,* a bedroom in a nursing home) is recorded. Based on the images captured by the sensor, the software detects whether a person fell down using a ML classification algorithm.

We apply tactic T11 "Apply edge computing" (see Section IV-A3) in our context, in order to evaluate the extent to which such tactic can lead to energy optimization. In other words, we evaluate the energy efficiency achieved by utilizing edge computing (Scenario 1) instead of utilizing exclusively cloud computations (Scenario 2). An overview of the main components characterizing the two scenarios are depicted in Figure 3, and further described below.

In Scenario 1, the sensor transmits the raw video data to an edge device over a Local Area Network (LAN). The edge device stores the ML model and classifies the incoming video data to detect whether a person fell down or not. Only when a fall is detected is data sent to the cloud. The decision-making is performed in the cloud, as opposed to on-premise, to benefit from the advantages of deploying in the cloud, such as auto-scaling and centralization. The alternative Scenario 2 consists of directly transferring the raw video data to the cloud. In this case, all steps (from classification to decision-making) are performed in the cloud.

B. Estimation energy consumption edge computing scenario

This section covers the estimation of the energy consumed in the edge computing scenario (Scenario 1).

Cameras utilized in the case study project generate 4.1Mbit/s and send the data over a cable to the edge device. As commonly 8 cameras are connected to the same edge device, in total 32.8Mbit/s is generated and sent to the edge device. This data is transferred through a cable.

The edge device used is an "NVIDIA Jetson AGX Xavier" [29]. The device allows three power modes, corresponding respectively to 10W, 15W, and 30W. For the case study, the higher 30W power mode is utilized to make full use of the edge device resources. The energy consumption of the edge device for one day can be estimated as $E_{[kWh]} = P_{[W]} \times t_{[hours]}/1000 = 30 \times 24/1000 = 0.72\,\text{kWh}$, where E is the energy consumption in kWh, based on the power (P) in W and time (t) in hours.

Based on the above calculation, we conclude that in Scenario 1 the edge device consumes *0.72 kWh* in 24 hours.

In addition to the energy consumed by the edge device, we also have to consider the energy required to transfer the data to the cloud. In the edge-computing scenario, data is transferred to the cloud only when a fall event is detected. The data to be transferred to the cloud for each event is equal to 2.4 KB. On average, 20 events are recorded for each camera. As we utilize an 8 camera setup, each day a total of 384.0 KB data is transferred from the edge device to the cloud (8 cameras $\times$ 20 messages $\times$ 2.4 KB = 384 KB).

To calculate the energy necessary to transfer data to the cloud, we use the electricity intensity rate of 0.06 kWh/GB, as defined by Aslan et al. [30]. Therefore, in Scenario 1, the total energy consumed to transfer data from the edge device to the cloud in 24 hours equals *0.00002304 kWh.*

C. Estimate energy consumption cloud-only scenario

As for Scenario 1, we conduct a separate energy estimation for the computation and data transfer tasks entailed by the use case for Scenario 2 (see Figure 3).

Regarding the energy estimation of the computational tasks (*i.e.,* the ML classification) in the cloud, we leverage the CCF tool [27] presented in Section V-B. The total CPU usage is 84.38% of the edge device and the total memory usage is 5.12 GB. We use this data (together with the hardware specification from the edge device) to estimate the energy consumption of the same workload in AWS for 24 hours.

Running the program for one day results in 24 [hours] $\times$ 8 [cores] = 192 vCPU hours. Each threat is represented by a vCPU and the edge device contains 8 cores running one threat each [31]. The minimum and maximum watts for a vCPU in AWS are 0.71 and 3.46 respectively. We calculate the energy consumption of computing according to the formula introduced by the CCF tool, *i.e.,*
Average Watts $=$ Min Watts $+$ Avg vCPU Utilization $\times$ (Max Watts$-$Min Watts) $= 0.71 + 0.84 \times (3.46 - 0.71) = 3.03$.
Compute Watt-Hours = Average Watts $\times$ vCPU Hours = 3.03 $\times$ 192 = 581.85 Wh $\approx$ 0.58 kWh.

The energy consumption of the GPU needs to be considered as well. When connected to 8 cameras, the GPU utilization is 90% of the edge device. This includes the encoder/decoder that transforms the input stream to the right format through the hardware. The encoding/decoding processes is expected to be similar in AWS as most GPUs have this functionality built in. The CCF tool currently does not provide an energy coefficient for GPU usage. Hence, we estimate the energy consumption based on the power consumption of a similar GPU which is 18 W [32]. Accordingly, in 24 hours, the expected energy consumption of the GPU is: $E_{[kWh]} = P_{[W]} \times t_{[hours]}/1000 = 18 \times 24/1000 = 0.432\,\text{kWh}$

Next, we calculate the energy consumption of the memory usage. According to the formula used by the CCF tool, the en-

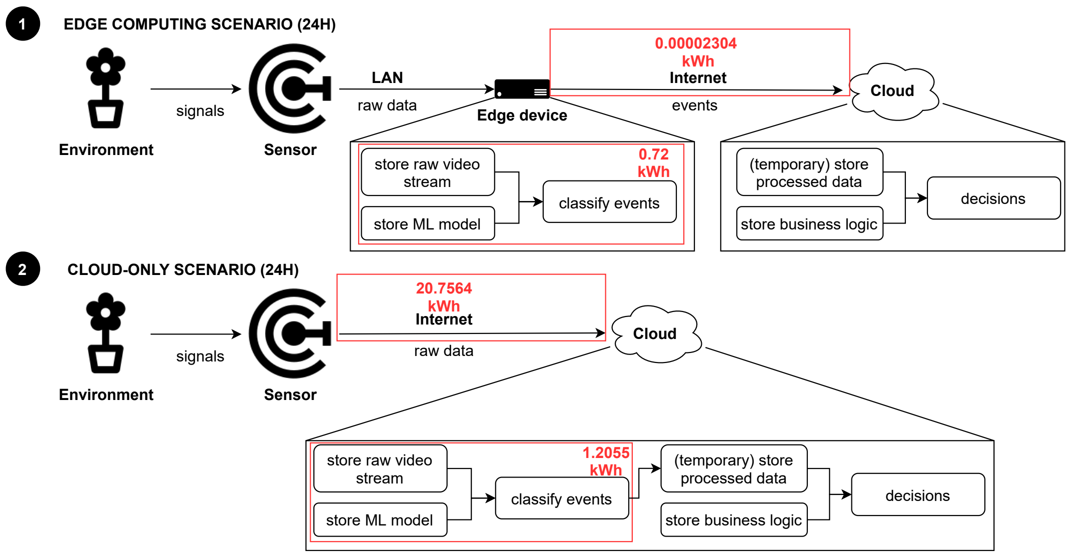

Fig. 3. Edge computing case study.

ergy consumption of the memory is equal to: Memory [GB] $\times$ Memory coefficient $\times$ Usage amount [hours] $= 5.12\,[\text{GB}] \times 0.000392\,[\text{kWh/GB}] \times 24\,[\text{hours}] \approx 0.0482$ kWh.

Adding the energy consumption of the CPU, GPU, and memory together results in 0.5819 [kWh] + 0.4320 [kWh] + 0.0482 [kWh] = 1.0621 kWh. Next, we need to multiply the estimated energy consumption by the PUE of AWS, resulting in an energy consumption of 1.0621 [kWh] $\times$ 1.135 [PUE] $\approx$ 1.2055 kWh. Hence, the energy consumption of the computation and memory in the cloud-only scenario for 24 hours is estimated to be *1.2055 kWh*.

Regarding the energy consumed to transfer data to the cloud, we use the same estimate utilized for Scenario 1, namely the electricity intensity rate of 0.06 kWh/GB, as defined by Aslan et al. [30]. As in total 32.8 Mbit/s raw data is generated, and in this scenario the raw data is transferred to the cloud, we estimate the energy consumption of **transferring the camera stream from the edge device to the cloud over the Internet** in 24 hours to be approximately $32.8\,[\text{Mbit}] \times 60\,[\text{minute}] \times 60\,[\text{hour}] \times 24\,[\text{day}] \times 0.06\,[\text{kWh/GB}] =$ *20.7564 kWh*.

VII. Discussion on Energy Efficiency in the Public Cloud

Using the previous findings, we consider our last research question, namely, *"How can energy efficiency be defined in the context of the public cloud?"*.

In general, energy efficiency can be defined as: *"the ratio between service output or results and the energy input required to provide it"* [33], *i.e.,* the amount of energy consumed to provide one "service output". Based on such definition, to assess the energy efficiency of a workload, energy consumption needs to be metered. Due to the abstraction the cloud forms between the cloud consumers and the DCs, energy consumption can commonly only be estimated, and not directly measured (*e.g.,* due to multiple software products running on the same hardware).

Once an energy consumption estimation is calculated, the estimation needs to be related to a quantifiable service output. Ideally, the service output would be represented as the concrete value software services deliver to end-users, as this is the ultimate goal of consuming computing resources. Unfortunately, end-user experiences are difficult to quantify and monitor in an automated fashion. To solve this representational issue, service output is often expressed in terms of resource consumption metrics (*e.g.,* GBs, vCPU minutes, or data transferred) [34]. However, using these resource usage metrics to calculate energy efficiency can result in the energy efficiency of the hardware. To illustrate, the metric $\frac{\text{energy consumption [kWh]}}{\text{storage [GB]}}$ reflects the energy efficiency of the cloud DC storage infrastructure. Such a metric is independent of the architecture of the cloud consumer, and can therefore not be optimized by the cloud consumer.

Accordingly, the architecture of the cloud consumer needs to be included when defining the service output to calculate the energy efficiency. The process of defining the service output is highly case-specific. For example, in the edge computing scenario, the service output can be quantified in terms of the amount of processed data at a certain time frame. Other examples to define the service output are: number of processed requests, number of transactions completed, number of simultaneous active users, API calls served [34]. Thus, an example definition for energy efficiency in the cloud is $\frac{\text{energy consumption [kWh]}}{\text{API calls served}}$. Consequently, cloud practitioners can optimize the energy consumed for an API call.

Last, we wish to address that resource efficiency can be used as an alternative metric to optimize energy efficiency as resource consumption is proportional to energy consumption. This alternative is recommended when energy estimations are not precise enough. Resource efficiency can be interpreted as $\frac{\text{resource consumption}}{\text{service output}}$ [34]. An example of a metric is: $\frac{\text{vCPU minutes}}{\text{completed transactions}}$. Optimizing this metric reduces the vCPU minutes required for a transaction. This implies that also less energy is consumed by the vCPU enabling the transaction.

VIII. Discussion

In this section, we discuss the results obtained for the four sub-RQs (Section VIII-A - SectionVIII-D), followed by a revisitation of our main research question (VIII-E).

A. Phase 1: Tactics for Energy Optimization

The first research phase considers the sub-RQ: *"Which cloud-native tactics are available to optimize the energy efficiency of cloud workloads?"*. We discovered 18 tactics by surveying both multivocal literature and practitioners. The tactics we discovered result to lay at a more applicable and concrete level than the tactics currently available in the academic literature. While the tactics we identified result to be straightforward and can be utilized to architect for energy efficiency in the cloud, a prominent downside has to be considered before implementing a tactic. Specifically, tactics may be utilized only for specific use cases, and can hence not be applied to every workload. For example, the tactic "Apply edge computing" used in the case study is not relevant for workloads that are not network-intensive. Therefore, the tactics cannot be perceived as *guidelines*, but rather as a catalog of tactics that may or may not be applicable to a specific context. To choose the relevant tactics from the catalog, cloud consumers should have a good understanding of their cloud infrastructure. Furthermore, some tactics (e.g., "Choose fitting computing paradigm") are fundamental architectural decisions that cannot be easily applied to an existing infrastructure. Cloud providers could support their consumers in deciding which tactics are most relevant and applicable.

B. Phase 2: Methods to Monitor Energy Consumption in the Public Cloud

This phase involves reviewing methods that estimate the energy consumption in the public cloud. The relevant sub-RQ is: *"What methods are available to monitor the energy consumption of individual workloads in the public cloud?"*. For cloud consumers, it is currently difficult to monitor their energy footprint in the public cloud. First of all, all major cloud providers do not provide cloud consumers with direct energy consumption measurements of individual workloads. Therefore, energy consumption estimates have to be used. From our review, the Cloud Carbon Footprint (CCF) tool [27] resulted to bridge this gap, by providing energy and carbon footprint estimates for individual workloads. The Cloud Carbon Footprint (CCF), which currently is the only open-source and cross-platform tool, estimates energy consumption by relying on billing data. Specifically, the energy estimation is derived from the billing data by considering the predicted hardware usage and corresponding wattage. As downside of the Cloud Carbon Footprint (CCF) tool, its results have to be regarded as coarse estimations, and hence would not work for fine-tuning energy-aware optimization. Nevertheless, cloud consumers with larger infrastructures can utilize the tool to perform experiments, and understand which tactics are more effective to optimize their energy consumption. As a final recommendation, we appeal to cloud providers in order to disclose more accurate energy-related metrics, in order to jointly move towards more environmentally-sustainable software products.

C. Phase 3: Case Study to Assess Impact of Tactic "Apply edge computing" on Energy Consumption

After completing research Phases 1 and 2, we addressed the sub-RQ: *"What is the impact of a selected identified tactic on the energy consumption?"*. The goal of this step is to gain further hands-on insights on the results of the previous phases, and consisted in applying the tactic T11 "Apply edge computing". The study investigated the extent to which applying edge computing positively affected the energy consumption of a software product. By considering the energy required for computation and data transfer, we concluded that applying the tactic leads to a drastic improvement in energy efficiency. From the case study results, applying T11 resulted in a 96% total energy consumption decrease. As discussed also in Section VIII-A, it is important to note that the degree of energy optimization implied by a tactic depends on the specific context. In some cases, applying a tactic may not be convenient (*e.g.,* due to the refactoring effort involved), or even possible. With our case study presented in Phase 3, we showcased how our tactic catalog (see Section IV) and energy estimation techniques (see Section V) can be applied to reduce energy consumption in the public cloud. Nevertheless, successfully applying a tactic ultimately lies in the hand of who is selecting and implementing it.

D. Phase 4: Energy Efficiency Definition in the Public Cloud

In this phase, we tackled the sub-question: *"How can energy efficiency be defined in the context of the public cloud?"*. As often in computer science, the no-free-lunch theorem applies, as the metric to define the energy efficiency of a cloud workload is highly case-dependent. In general, measuring the energy efficiency of a workload in the public cloud requires two metrics, energy consumption and service output. Only cloud providers can directly measure energy consumption. In contrast, only cloud consumers can define service output. We argue that an appropriate abstraction level to define the service output lies in-between resource usage metrics and the (difficult to quantify) value to the end-users. Examples of appropriate metrics to express the service output are the amount of processed data or the number of handled requests. These metrics are appropriate as they (i) include design decisions of the cloud consumer and not solely the efficiency of the underlying infrastructure, and (ii) capturing these metrics is automatable.

E. Architecting Energy-Efficient Software in the Public Cloud

Finally, we reflect on our main research question: *"How can cloud consumers architect energy-efficient software in* *the public cloud?"*. First, awareness must be raised on the importance of energy reduction strategies [35]. On a company level, relevant strategies and requirements need to be established and followed. In addition, as for any other quality attribute, energy efficiency should be measured and monitored over time. To optimize energy efficiency, two metrics are required, energy consumption and service output. On one hand, cloud consumers are currently in need of accurate measures of their their energy footprint. On the other hand, cloud consumers need to define an appropriate metric reflecting their service output. Once the two metrics are established, cloud consumers are enabled to investigate opportunities to optimize their energy efficiency. To accomplish this goal, the tactics discovered with this investigation, and the methods to monitor energy efficiency, can be used to identify and evaluate relevant strategies.

IX. Limitations

Despite our best efforts, our research may be affected by threats to validity. In this section, we discuss potential threats to validity and mitigation strategies by following the threats classification of Wohlin et al. [36].

External Validity. The identified tactics were derived from interviews with 17 practitioners belonging to the same company (SBP). Therefore the tactics may not be representative of the entire sector. Albeit not mitigated, we conjecture this threat did not significantly influence our results, as SBP collaborates with many companies from different sectors (*e.g.,* public, financial, transport) and therefore reflects the experiences of heterogeneous cloud consumers. Regarding our case study (see Section VI) we do not claim external validity of its results, also in light of the goal of the research phase, namely demonstrating the applicability of the results gained from our previous findings (see Section IV and Section V). Furthermore, the case study results are based on estimations, which may vary according to the specific resource considered as ground data.

Internal Validity. Prominent internal validity threats arise from the calculations performed in the case studies, which are of theoretical rather than empirical nature. We mitigated this threat by relying exclusively on well-known peer-reviewed energy estimation methods. As additional internal threat, the identified tactics for energy efficiency rely on the underlying assumption that resource usage optimization is linearly correlated to energy consumption optimization. While this assumption may hold in the vast majority of cases, it is important to note that the impact of the tactics may vary according to the specific context considered. Also, we have to be careful that the practitioners lack the scientific background when making claims on resource efficiency. This threat was mitigated by reasoning on the impact of applying the tactics on energy efficiency with the domain experts. Furthermore, the discovered tactics may not be complete due to the limited number of interviewees, *i.e.,* other tactics, such as the importance of caching, did not emerge from our interviews.

Construct Validity. A potential threat to conclusion validity arises due to only one researcher conducting the interviews and identifying tactics during Phase 1. Furthermore, the interview transcripts can not be made public, due to the confidential nature of the conversations. We mitigated this threat by requesting domain experts to review and verify the tactics. In addition, the interview and tactic identification process was monitored by two academic supervisors, and an industrial one.

Conclusion Validity. A threat to conclusion validity is due to the missed involvement of cloud providers and consumers in the study. We mitigated this threat by ensuring the interviewed practitioners frequently collaborated with cloud consumers. Interviewees were selected based on their cloud experience, while ensuring that their professional roles were diverse, to reflect experiences from different areas of cloud-centric expertise. As the most prominent cloud providers supporting energy monitoring capabilities (see Section V) resulted to be unavailable for this investigation, we acknowledge their missing involvement as an unmitigated threat, in the hope to mitigate this threat in our future work.

X. Conclusion and Future Work

In this study, we address the research question: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. To answer this question, based on a study of white literature and interviews with practitioners, we discover 18 architectural tactics to optimize energy efficiency in the public cloud. As sustainability-aware cloud consumers need to measure the impact on energy efficiency of their design decisions, we also review the available methods to monitor energy consumption in the public cloud. Based on our findings, we conduct a case study by applying a discovered tactic (T11, "Apply edge computing") and identify an energy consumption monitoring tool (CCF). The case study allows us to gain further practical insights into our previously obtained findings. Finally, based on the knowledge acquired, we discuss how software energy efficiency can be defined in the public cloud.

As conclusion, despite a recent shift towards providing sustainability measurements to cloud consumers, the topic of software sustainability in the public cloud is still at an early stage. With this study, we identified a set of architectural tactics, and showcased how these can be successfully applied to improve software energy efficiency in the public cloud. However, the availability of tactics alone does not suffice. On one hand, cloud consumers need to be gain sensibility on their software sustainability, and establish processes to address it. On the other hand, current sustainability measurements supported by cloud providers are limited and often coarse-grained. Cloud providers need to gain responsibility for the sustainability of their consumers, by supplying easily accessible, clearly defined, and precise sustainability measurements.

As future work, we plan to empirically evaluate and compare the impact of all the tactics identified in this paper. In addition, we envision to improve the state of the art of sustainability measurements in the public cloud, in tight collaboration with both cloud providers and consumers.

REFERENCES

[1] ReportLinker, "Cloud Computing Market by Service, Deployment Model, Organization Size, Vertical And Region - Global Forecast to 2025," 2020. [Online]. Available: https://www.reportlinker.com/p05749258/Cloud-Computing-Market-by-Service-Deployment-Model-Organization-Size-Workload-Vertical-And-Region-Global-Forecast-to.html

[2] Accenture, "The green behind the cloud," http://www.accenture.com/_acnmedia/PDF-135/Accenture-Strategy-Green-Behind-Cloud-POV.pdf, 2021, accessed on: 2021-07-03.

[3] L. Belkhir and A. Elmeligi, "Assessing ICT global emissions footprint: Trends to 2040 & recommendations," *Journal of Cleaner Production*, vol. 177, pp. 448 – 463, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095965261733233X

[4] R. Verdecchia, P. Lago, C. Ebert, and C. De Vries, "Green IT and Green Software," *IEEE Software*, vol. 38, no. 6, pp. 7–15, 2021.

[5] Amazon Web Services, "Well-Architected Framework," Tech. Rep., December 2021. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/framework/wellarchitected-framework.pdf

[6] K. Djemame, D. Armstrong, R. Kavanagh, A. J. Ferrer, D. G. Perez, D. Antona, J.-C. Deprez, C. Ponsard, D. Ortiz, M. Macias *et al.*, "Energy efficiency embedded service lifecycle: Towards an energy efficient cloud computing architecture," in *CEUR Workshop Proceedings*, vol. 1203. CEUR Workshop Proceedings, 2014, pp. 1–6.

[7] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, pp. 1045–1051, 2010.

[8] R. Kazman, S. Haziyev, A. Yakuba, and D. Tamburri, "Managing energy consumption as an architectural quality attribute," *IEEE, Software*, vol. 35, no. 5, pp. 102–107, Sep. 2018.

[9] G. Procaccianti, P. Lago, and G. Lewis, "Green architectural tactics for the cloud," in *2014 IEEE/IFIP Conference on Software Architecture*, 2014, pp. 41–44.

[10] C. Paradis, R. Kazman, and D. A. Tamburri, "Architectural tactics for energy efficiency: Review of the literature and research roadmap," in *Proceedings of the 54th Hawaii International Conference on System Sciences*. Hawaii: ScholarSpace, 2021, pp. 7197–7206. [Online]. Available: https://hdl.handle.net/10125/71488

[11] V. R. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*. Wiley, 1994, pp. 528–532.

[12] V. Garousi, M. Felderer, M. V. Mäntylä, and A. Rainer, "Benefitting from the grey literature in software engineering research," in *Contemporary Empirical Methods in Software Engineering*. Springer, 2020, pp. 385–413.

[13] F. Richter, "Cloud Infrastructure Market: Amazon Leads $130-Billion Cloud Market," *Statista*, 2021.

[14] AWS Autoscaling, https://aws.amazon.com/autoscaling, accessed on: 2021-09-05.

[15] AWS Cost Management, https://aws.amazon.com/aws-cost-management/aws-cost-optimization/right-sizing, accessed on: 2021-09-05.

[24] Microsoft PowerBI Pro, https://powerbi.microsoft.com/en-us/power-bi-pro, accessed on: 2022-01-06.

[16] AWS Reserved Instances, https://aws.amazon.com/aws-cost-management/aws-cost-optimization/reserved-instances, accessed on: 2021-07-26.

[17] AWS Spot Instances, https://aws.amazon.com/aws-cost-management/aws-cost-optimization/spot-instances, accessed on: 2021-05-09.

[18] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE, Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[19] AWS Athena, https://aws.amazon.com/athena, accessed on: 2021-05-09.

[20] AWS S3, https://aws.amazon.com/s3, accessed on: 2021-05-09.

[21] R. Verdecchia, P. Kruchten, P. Lago, and I. Malavolta, "Building and evaluating a theory of architectural technical debt in software-intensive systems," *Journal of Systems and Software*, vol. 176, p. 110925, 2021.

[22] Microsoft Emissions Impact Dashboard, https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard, accessed on: 2022-01-06.

[23] Greenhouse Gas Protocol, "What are scope 3 emissions?" https://ghgprotocol.org/sites/default/files/standards_supporting/FAQ.pdf, 2021, accessed on: 2022-01-06.

[25] Google Cloud Carbon Footprint Tool, https://cloud.google.com/carbon-footprint, accessed on: 2022-01-06.

[26] AWS S3, https://www.aboutamazon.com/news/aws/aws-re-invent-2021-what-you-need-to-know?p=aws-announces-customer-carbon-footprint-reporting, accessed on: 2022-01-06.

[27] ThoughtWorks, "Cloud Carbon Footprint," https://www.cloudcarbonfootprint.org, 2021, accessed on: 2021-06-26.

[28] E. Sommer, M. Adler, J. Perkins, J. Thiel, H. Young, C. Mozen, D. Daya, and K. Sundstrom, "Cloud Jewels: Estimating kWh in the Cloud," *Code as Craft*. Etsy, https://codeascraft.com/2020/04/23/cloud-jewels-estimating-kwh-in-the-cloud, 2020, accessed on: 2021-06-26.

[29] NVidia Jetson Developer Kit, https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit, accessed on: 2021-07-06.

[30] J. Aslan, K. Mayers, J. Koomey, and C. France, "Electricity intensity of internet data transmission: Untangling the estimates," *Wiley Online Library, Journal of Industrial Ecology*, vol. 22, no. 4, pp. 785–798, 2018.

[31] AWS Optimize CPU, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-optimize-cpu.html, accessed on: 2021-08-10.

[32] Geforce Radeon Power, https://www.tomshardware.com/reviews/geforce-radeon-power,2122-4.html, accessed on: 2021-08-14.

[33] L. Pérez-Lombard, J. Ortiz, and D. Velázquez, "Revisiting energy efficiency fundamentals," *Springer, Energy Efficiency*, vol. 6, no. 2, pp. 239–254, 2013.

[34] 451 Research, commissioned by AWS, "The Carbon Reduction Opportunity of Moving to Amazon Web Services," Tech. Rep., 2019.

[35] P. Lago and T. Jansen, "Creating Environmental Awareness in Service Oriented Software Engineering," in *Service-Oriented Computing*. Springer, 2011, pp. 181–186.

[36] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.