Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

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

Author: Sophie Vos (2551583)

1st supervisor:	Patricia Lago	Professor Software & Sustainability
industry supervisor:	Ilja Heitlager	CIO at Schuberg Philis
2nd reader:	Roberto Verdecchia	Research Fellow Software & Sustainability

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

August 24, 2021

"The most damaging phrase in the language is: 'It's always been done that way'."
Grace Hopper

Abstract

A trend can be observed where companies and organizations migrate their ICT infrastructure to the public cloud (e.g., Amazon Web Services, Microsoft Azure, Google Cloud Platform). Cloud computing reduces the energy footprint of individual ICT workloads due to economies of scale and virtualization technologies. Nevertheless, the energy and carbon footprint of the global ICT sector is increasing due to the growing demand for software services. Unfortunately, cloud consumers cannot trace their energy footprint in the public cloud and, consequently, cannot optimize their ICT workload for energy efficiency.

In this thesis, we aim to support cloud consumers in architecting energy-efficient workloads in the public cloud. We start from the assumption that resource optimization leads to energy efficiency as energy usage is proportional to resource usage. We collaborate with Schuberg Philis[1] to discover an initial set of reusable architectural tactics for energy efficiency in the public cloud based on interviews with 17 practitioners. Afterward, we reviewed and selected available methods to estimate the energy consumption of individual workloads in the public cloud. Next, we selected two tactics and conducted case studies to assess the impact of the tactics on the energy consumption. Last, we define energy efficiency within the context of the public cloud. The main contribution of this thesis is the first scientific description of a method to guide cloud consumers in addressing the quality attribute energy efficiency.

[1] Software consultancy firm that is experienced with cloud development and migrations (`https://schubergphilis.com`).

Acknowledgements

This project would not have been possible without the support of many people. I would like to thank my supervisors from the Vrije Universiteit Amsterdam, Patricia Lago and Roberto Verdecchia, for the useful insights and support during the entire research.

Thanks to the people from Schuberg Philis for sharing their expertise and experience. Foremost to Ilja Heitlager for the guidance throughout the thesis. Also thanks to Martijn van Dongen and Boris Schrijver for reviewing parts of the research.

Last but not least, thanks to the researchers and developers of the Cloud Carbon Footprint tool. Their research is an important building block for this thesis. Especially thanks to Dan Lewis-Toakley for supporting me in setting up the experiment environment.

Table of Contents

TABLE OF CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Acronyms

API Application Programmer Interface. 45

APIs Application Programmer Interfaces. 43, 45

AWS Amazon Web Services. iii, 2, 12, 23, 25, 27, 29, 31, 34–36, 41, 43, 45–47, 55–57, 59, 61, 62

CCF Cloud Carbon Footprint. vii, 42–47, 55, 56, 62, 66, 68, 71, 72, 75

CDN Content Delivery Network. 47

CPU Central Processing Unit. 6, 16, 42, 45, 57

CRAC Computer Room Air Conditioning. 6, 7

DC Data Center. vii, 1–3, 5–9, 12, 43, 46, 67, 68

DCs Data Centers. 1, 2, 4, 5, 7–9, 12, 15, 31, 46, 47

DPs Design Principles. 27, 28

EC2 Elastic Compute Cloud. 59, 63

FaaS Function as a Service. 60

GCP Google Cloud Platform. iii, 2, 4, 12, 33, 41, 43

GHGEs Greenhouse Gas Emissions. 2, 8, 9

GPU Graphical Processing Unit. 6, 56, 57

GPUs Graphical Processing Units. 56

HDD Hard Disk Drive. 42

HDDs Hard Disk Drives. 6

Acronyms

IaaS Infrastructure as a Service. 10, 59

ICT Information and Communication Technology. iii, 1–3, 5, 6, 8, 38, 68

IoT Internet of Things. 49, 67

KPIs Key Performance Indicators. 8

LAN Local Area Network. 51

ML Machine Learning. 35, 51, 54, 57, 67

NIST National Institute of Standards and Technology. 1, 10

OS Operating System. 59, 63, 64

PaaS Platform as a Service. 10, 60

PDUs Power Distribution Units. 7

PUE Power Usage Effectiveness. 4, 7, 8, 12, 43, 47, 57

RAM Random Access Memory. 6

SaaS Software as a Service. 10

SBP Schuberg Philis. iii, 3, 23, 25–27, 29, 34, 43, 51, 59, 62, 65, 71

SLA Service Level Agreement. 17

SOA Service-Oriented Architecture. 14

SSD Solid-State Drive. 42

UPS Uninterruptible Power Supply. 7

vCPU virtual CPU. 43, 45, 55

VM Virtual Machine. vii, 17, 19, 36, 37, 42, 47, 49, 59–61, 63, 64, 75

VMs Virtual Machines. 17, 19, 59, 60, 62–64, 67, 75

WAF Well-Architected Framework. 27, 29

Chapter 1

Introduction

An increasing number of companies and organizations migrate their Information and Communication Technology (ICT) infrastructures to the public cloud. The global cloud computing market was worth over 300 billion dollars in 2020 and is predicted to reach 832 billion dollars by 2025 [7]. The cloud computing market growth is further illustrated in Figure 1.1. Many definitions of cloud computing exist, but the NIST definition is widely accepted and states that: *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [8]. Informally, cloud computing involves the shift from performing computation and data storage on-premise to a geographically remote Data Center (DC). An important benefit of migrating software to the public cloud is shifting the responsibility of maintaining and operating hardware resources to the cloud provider. Accordingly, the cloud consumer can focus on their mission-critical processes [9].

For cloud consumers, migrating ICT workloads to the public cloud is expected to reduce their energy footprint. 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% [10]. This is due to the efficient hardware and cooling architectures of hyperscale cloud DCs [2], economies of scale, and elasticity of the workload. To illustrate, on-premise DCs have a relatively low server utilization ranging between 20% to 30% [11]. Unfortunately, an idle server still consumes about 60% of the energy relative to a fully utilized server [11]. Hence, running idle servers consumes a significant proportion of energy while not contributing to the operation. In the cloud, multiple consumers can share resources, and, due to virtualization techniques, multiple workloads are consolidated

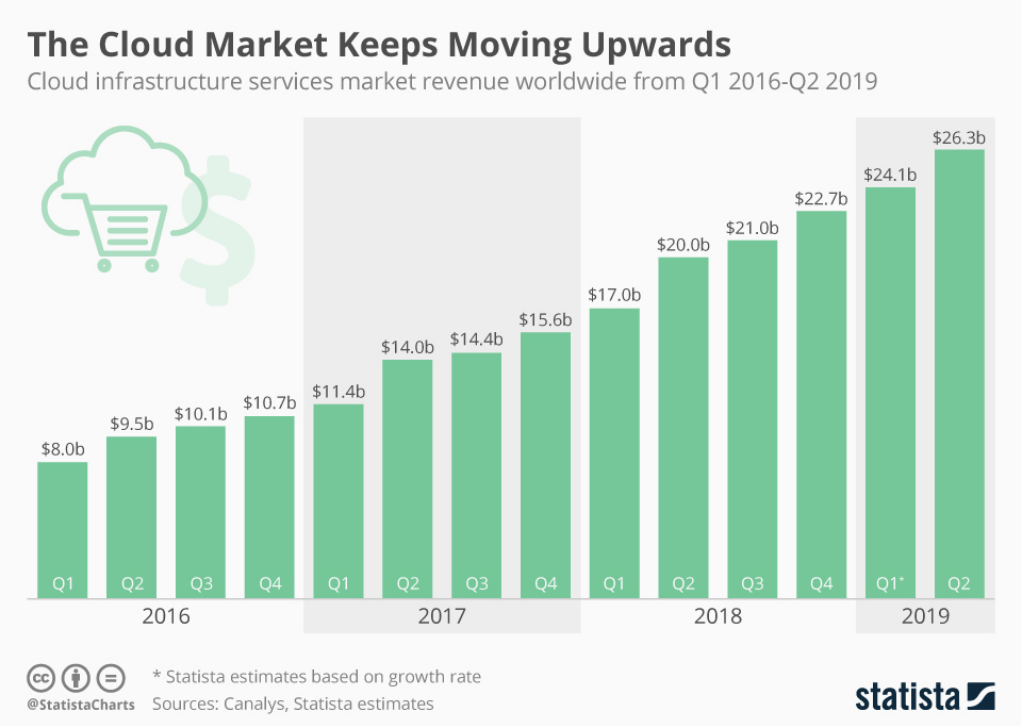


Figure 1.1: Rising cloud infrastructure service market (borrowed from [1]).

onto the same hardware [2]. Hence, servers can be used more efficiently which allows idle servers to be turned off to save energy.

However, 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. [12] predict that the global GHGEs of the ICT sector relative to the worldwide footprint doubles from 1-1.6% in 2007 to 3–3.6% by 2020. As means of comparison, in 2018, the energy footprint of the aviation industry accounts for around 2% of the global GHGEs [13]. DCs have with 45% the largest energy footprint within the ICT sector [12].

Major cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) conduct several measures to increase the environmental sustainability of their cloud platforms. Yet, the strategies published on their websites are mainly focused on the efficiency of the DC infrastructure and use of renewable energy[1][2][3].

[1] `https://sustainability.aboutamazon.com/environment/the-cloud` (Accessed on: 2021-07-21)

[2] `https://azure.microsoft.com/en-us/global-infrastructure/sustainability` (Accessed on: 2021-07-21)

[3] `https://cloud.google.com/sustainability` (Accessed on: 2021-07-21)

Neither of these cloud providers discloses information regarding the energy footprint of individual workloads to the cloud consumers [6, 14]. Hence, cloud consumers cannot directly trace their energy footprint in the public cloud. As follows from our research, this provides practitioners with little incentives and tools to actually take energy efficiency into account when designing and developing workloads in the public cloud.

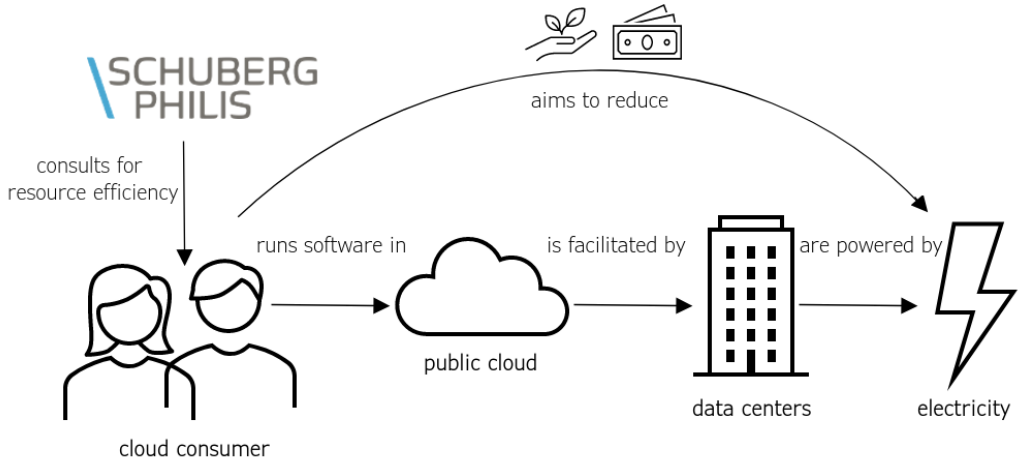


Figure 1.2: Context of the study.

This thesis is executed in collaboration with Schuberg Philis (SBP). SBP is a company that consults and supports diverse organizations and businesses with their mission-critical processes[1]. The **context** of our research is visualized in Figure 1.2. Currently, many organizations focus on migrating ICT workloads to the public cloud. SBP supports these migration processes and is experienced in achieving resource efficiency. In the current landscape, organizations are often interested in reducing their carbon and energy footprint to accomplish sustainability goals. However, due to the abstraction that the cloud creates between the organization and the DC, cloud consumers experience less awareness and control with respect to their energy footprint. In our research, we aim to support cloud consumers to optimize their software architecture for energy efficiency.

The main **research question** is: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. To approach this broad research question, we start by identifying tactics for resource efficiency. These tactics are discovered at SBP as many practitioners are experienced in architecting and developing software in the public cloud.

[1] `https://schubergphilis.com` (Accessed on: 2021-05-16)

We work from the assumption that resource efficiency leads to energy efficiency as optimizing the resources usage, often, fewer resources, computing time, and capacity are required. Hence, less energy is used to power these resources. Next, in order to assess the energy efficiency of workloads, a method is required to measure or estimate the energy consumption in the public cloud. This is not straightforward as the major cloud providers do not disclose energy-related data. Hence, the second phase of our research focuses on finding a method to estimate the energy consumption in the public cloud. Afterward, two discovered tactics for resource efficiency are selected based on their potential impact on energy efficiency. For these tactics, an in-depth case study and analysis are conducted to assess the impact of these tactics on the energy consumption. Last, based on our findings, we reflect on the definition of energy efficiency in the public cloud.

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. We chose to focus on software as opposed to hardware as this is within the control of cloud consumers. Furthermore, the hardware in hyperscale cloud DCs is currently well-optimized. For example, GCP reports a Power Usage Effectiveness (PUE) of 1.1 of their DCs[1]. Moreover, even though hardware is the direct power consumer, software is identified as the true power consumer as it is the software that directs the hardware [15].

Due to the interdisciplinary nature of the thesis, we start by providing extensive background information and definitions to ensure understandability to readers from diverse fields (**Chapter 2**). Afterward, we discuss related scientific literature in **Chapter 3** and present the goal, research questions, and study design in **Chapter 4**. The following four chapters are structured according to the **main contributions** of this thesis. **Chapter 5** presents the model of reusable tactics for resource efficiency and reflects on the impact on energy efficiency. **Chapter 6** elaborates on the method to monitor the energy consumption in the public cloud. Afterward, **Chapter 7** combines the knowledge of the previous two contributions to define and execute case studies for two selected tactics. Thereafter, **Chapter 8** provides a definition of energy efficiency in the public cloud. Last, Chapter **Chapter 10** concludes the research.

[1] `https://www.google.com/about/datacenters/efficiency` (Accessed on: 2021-07-03)

Chapter 2

Background

As this research integrates multiple research fields (e.g., energy, software engineering, distributed systems), background information is required to ensure understandability to readers from diverse backgrounds. Accordingly, this section defines and describes the fundamental concepts as follows: first, Section 2.1 provides a model of a DC. Afterward, Section 2.2 explains concepts related to the energy efficiency of a DC. Next, Section 2.3 defines cloud computing and, last, Section 2.4 defines essential software architecture principles, namely, quality attributes and architectural tactics.

2.1 Data Center

According to CISCO [16], a DC can be defined as a *"physical facility that organizations use to house their critical applications and data"*. A DC contains multiple ICT resources that are connected and form a network. Nowadays, modern DCs frequently adopt a level of virtualization on top of the physical infrastructure. The DCs used by large cloud providers have evolved into Warehouse-Scale Computers [2] or hyperscale DCs. These hyperscale DCs differ from traditional DCs as the hardware is relatively homogeneous and the services share a single management layer. Barroso et al. [2] state that hyperscale DCs are not simply a collection of machines but one integrated system. The three main building blocks of a DC are ICT resources (Section 2.1.1), cooling equipment (Section 2.1.2), and supporting equipment (Section 2.1.3). Metrics used to monitor the energy efficiency of a DC are discussed in Section 2.1.4.

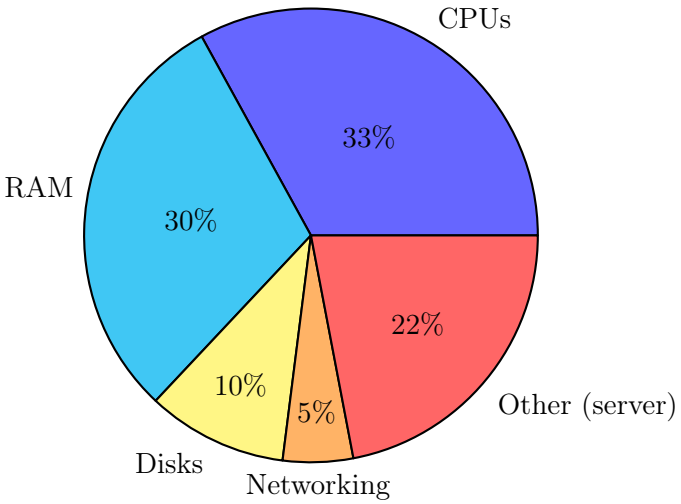


Figure 2.1: Approximate distribution of power usage within a server (borrowed from [2]).

2.1.1 ICT resources

The fundamental ICT resources are servers, network, and storage. **Servers** are computers that process requests and deliver data. These servers consist of computing power (i.e., Central Processing Unit (CPU)/Graphical Processing Unit (GPU)) and memory (e.g., Random Access Memory (RAM)). The computing and memory components are supported by cooling fans and communication equipment. Multiple servers are collected into a rack to scale the computing power. Accordingly, these racks are combined into clusters and can be connected using local Ethernet switches [2]. The approximate distribution of peak power usage by hardware subsystems of servers is shown in Figure 2.1. We observe that the CPU and memory consume relatively most power within a server.

The **network** resources involve routers, switches, modems, firewalls, and cables that enable data transport and communication between the servers and storage equipment. Furthermore, the network can connect the DC to the outside world through the internet [2].

The last component of the ICT resources is **storage** that facilitates long-term data storage in disks (e.g., Hard Disk Drives (HDDs)) [2].

2.1.2 Cooling equipment

Using many ICT resources in close proximity generates a significant amount of heat. Unfortunately, these ICT resources fail when overheating. Hence, cooling equipment is required to cool down the resources and environment. Computer Room Air Conditioning (CRAC)

units are used to cool the servers by blowing cold air at the front of the servers. The cold air is warmed from the server operations and conducted to the back of the servers. This results in warm and cold aisles [2]. Currently, many more advanced techniques exist to cool DCs, for more examples of techniques we refer to [2]. Temperature management (i.e., CRAC, chiller, and humidifier) accounts for a significant proportion of the energy usage in a DC (see Figure 2.2).

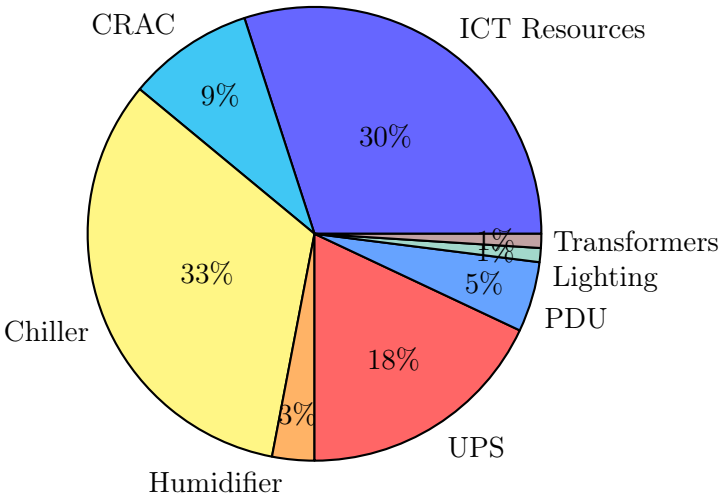


Figure 2.2: Breakdown of a typical DC's energy usage (borrowed from [2]).

2.1.3 Supporting equipment

Additional supporting equipment are lightning and power equipment. The proportion of energy consumed in a DC from lighting is negligible (Figure 2.2). In contrast, power equipment accounts for a relatively large proportion of the consumed energy (Figure 2.2). The power enters the DC from an outside transformer. Accordingly, the power flows to the Uninterruptible Power Supply (UPS). The UPS also ensures backup power in the case of power shortage or failure. The power from the UPS is routed to the Power Distribution Units (PDUs) which direct the power to the relevant resources [2]. The power equipment accounts for approximately 23% of the used energy in a DC.

2.1.4 Metrics

A metric that is well-adopted in the industry to reflect the energy efficiency of a DC is the Power Usage Effectiveness (PUE). The PUE represents the ratio of the power used by the

complete DC to the energy used by the ICT-equipment [17]:

$$PUE = \frac{total\ facility\ power}{ICT\ equipment\ power}$$

The PUE has been criticized as it is an incomplete metric for capturing the energy footprint of DCs. It is certainly useful to understand the energy overhead of the facility, however, the metric does not measure the energy efficiency of the computing hardware, the energy productivity, or the energy performance regarding the carbon emissions [17].

Many other metrics and Key Performance Indicators (KPIs) are available. For the interested reader, we refer to the work of Reddy et al. [18]. In this work, the authors provide a thorough overview and classification of metrics that measure different DC components. An interesting approach to monitoring the sustainability of DCs is presented by Lykou et al. [19]. The authors introduce a novel sustainability scoring model that combines the evaluation of the environmental impact and operational efficiency of DCs.

2.2 Energy

DCs use electricity to power ICT resources, cooling -, and supporting equipment. Energy is defined as power [watt] over time [seconds] and is expressed in joules or kilowatt-hour (kWh). The main motives of stakeholders to cut energy are a reduction in energy-related costs and environmental impact (e.g., GHGEs). There exist many related concepts to evaluate the energy usage of a system. To disambiguate these concepts, we define the terms energy consumption and energy efficiency in Section 2.2.1, carbon footprint and energy footprint in Section 2.2.2, and last, exhaustive energy and renewable energy in Section 2.2.3.

2.2.1 Energy consumption vs. energy efficiency

Perez-Lombard et al. [20] defined and analyzed concepts related to energy efficiency. **Energy consumption** can be defined as *"the service demand times the energy intensity"*. **Energy intensity** is *"the amount of energy needed to provide the unit of service or activity"*. In other words, energy consumption is the absolute energy used by a service or activity. This relates to **energy saving** which is defined as *"a reduction in the use of energy"*. Hence, energy saving means reducing the energy consumption.

In contrast, energy efficiency is commonly used to indicate technological improvements to reduce the energy intensity. **Energy efficiency** can be defined as *"the ratio between*

service output or results and the energy input required to provide it". Hence, energy efficiency involves a relation to the service output. This means that the energy efficiency can increase by increasing the service output using the same amount of energy. Thus, increased energy efficiency does not automatically imply that the energy consumption is reduced. An increase of the energy efficiency *could* lead to energy savings and energy savings *could* lead to increased energy efficiency. Perez-Lombard et al. [20] stress that energy efficiency strategies are not sufficient to address global energy challenges. They state that the amount of energy consumed by systems should be reduced to mitigate the impact on the environment.

2.2.2 Carbon footprint vs. energy footprint

The **carbon footprint** is defined as: *"a measure of the exclusive total amount of carbon dioxide emissions that is directly and indirectly caused by an activity or is accumulated over the life stages of a product"* [21]. To define the **energy footprint** we can replace *carbon* with *energy* in the definition above. This means that not only the direct energy consumption is relevant but the indirect energy as well. To ensure the feasibility of measuring the energy footprint in our study (i.e., energy footprint of cloud computing), we include the energy consumed by all components within a DC as defined in Section 2.1.

2.2.3 Exhaustible energy vs. renewable energy

Currently, DCs most frequently utilize fossil-based fuels as main energy source as these fuels can continuously generate electricity. Unfortunately, fossil-based fuels are exhaustive and a primary source of GHGEs. To reduce the negative effects on the environment, exhaustible (gray) energy sources can be replaced by renewable (green) energy sources such as wind and solar power. Strictly speaking, replacing gray energy with green energy does not affect the energy efficiency and consumption as the amount of energy used remains constant. However, it does reduce the environmental impact [20]. When green energy is used as opposed to gray energy, the energy footprint remains the same, however, the carbon footprint is reduced.

2.3 Cloud Computing

This section defines cloud computing in Section 2.3.1 and describes the currently dominating cloud providers as well as their sustainability strategies in Section 2.3.2.

2.3.1 Definition cloud computing

The well-accepted American National Institute of Standards and Technology (NIST) definition of cloud computing is:

> *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [8].

In cloud computing, there is a distinction between the cloud provider that offers computing resources and the cloud consumer that consumes the computing resources.

The **main characteristics** [8] of the definition are: (1) *on-demand self-service* which applies to automatic provisioning of computing resources to the consumer, (2) *broad network access* which applies to the availability of the computing resources over a standard network that can be accessed by heterogeneous client devices, (3) *resource pooling* which applies to the pooling of computing resources to serve multiple consumers simultaneously, (4) *rapid elasticity* which involves the elastic provisioning and release of the computing resources to match the demand of the consumer, and (5) *measured service* which involves resource metering to ensure transparency between the provider and consumer.

There are three **service models** [8] in which the computing capabilities can be provisioned to the consumers: (1) *Software as a Service (SaaS)* where the consumer can use applications of the provider which runs on a cloud infrastructure. The consumer does not manage the underlying cloud infrastructure. (2) *Platform as a Service (PaaS)* where the consumer can develop an application on top of the platform provisioned by the provider. (3) *Infrastructure as a Service (IaaS)* where the consumer has full control over the software of computing resources provisioned by the provider.

Finally, there exist four **deployment models** [8] in the cloud, namely: (1) *private cloud* in which the cloud infrastructure can be exclusively used within one organization, (2) *community cloud* in which the cloud infrastructure can be used only by a specific community, (3) *public cloud* in which the cloud infrastructure is accessible by the general public, and (4) *hybrid cloud* which is a combination of two or more of the previously defined deployment models.

2.3.2 Cloud providers

In this section, we discuss the three largest cloud providers together with their sustainability strategies according to their own publications. Figure 2.3 presents the worldwide

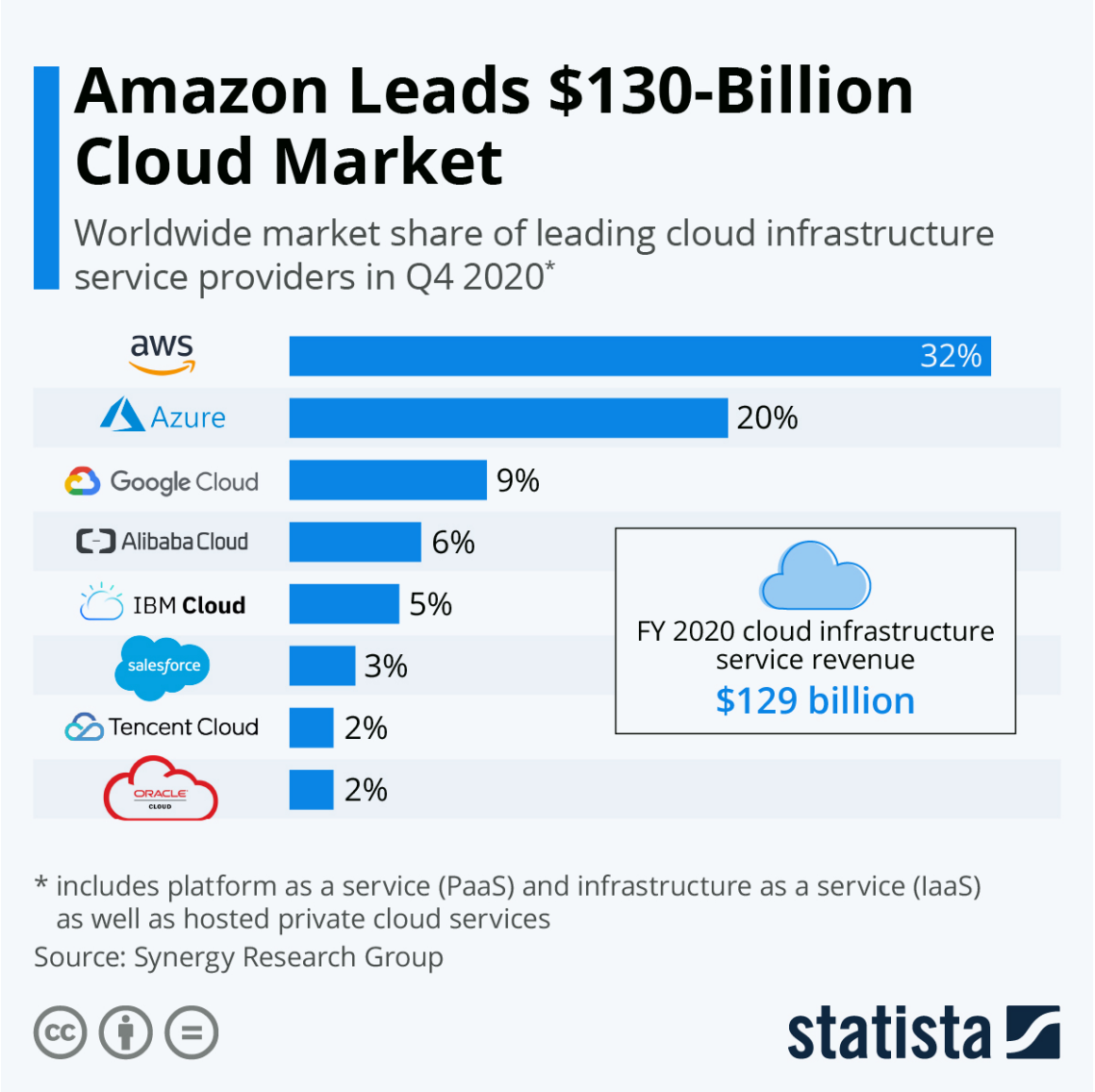


Figure 2.3: Worldwide market share of leading cloud infrastructure service providers in Q4 2020 (borrowed from [3]).

market share of leading cloud infrastructure service providers in the fourth quarter of 2020 (borrowed from [3]). Currently, AWS is dominating the market by 32%. Next up is Microsoft Azure with 20% and, thereafter, Google Cloud Platform (GCP) with 9%.

Amazon Web Services (AWS) is a cloud platform that offers over 200 services from DCs distributed over the globe. AWS has 80 availability zones spread over 25 geographic regions around the world [1]. AWS has as goal to use 100% renewable energy. 451 Research published a study [22] (commissioned by AWS) showing that AWS its infrastructure is 3.6 times more energy-efficient than the median of a sample of enterprise DCs. This energy saving is mostly attributed to energy-efficient hardware and higher server utilization.

Microsoft Azure provides more than 200 products and cloud services designed to build, run, and manage applications[2]. Microsoft published a study [23] claiming that the Microsoft cloud is between 22% and 93% more energy-efficient compared to on-premise solutions. This energy saving is attributed by: (1) *IT operational efficiency* where economies of scale allow more efficient resource usage through dynamic provisioning (i.e., dynamically matching server capacity with demand) and multi-tenancy (i.e., consolidating multiple workloads on the same hardware and thereby reducing idle resources). (2) *IT equipment efficiency* where the use of tailored and efficient IT equipment reduces the energy. (3) *DC infrastructure efficiency* which involves the efficiency of the overall DC by optimizing the PUE. Furthermore, carbon saving is achieved by the use of renewable energy.

Google Cloud Platform (GCP) offers over 100 products and their DCs are divided into 5 worldwide regions [3]. Google profiles themselves as the cleanest cloud in the industry. Currently, Google is carbon neutral and their goal is to power all their DCs on carbon-free energy by 2030. The PUE of their DCs is 1.1 and Google claims that its DCs are twice as energy-efficient as typical DCs[4].

We note that the sustainability strategies mainly involve the use of renewable energy and efficient resource utilization. Little is published on best practices for cloud consumers in contributing by designing and deploying energy-efficient workloads. In our research, we aim to explore this field and understand the role cloud consumers play in increasing the energy efficiency of the cloud.

[1] `https://aws.amazon.com/what-is-aws` (Accessed on: 2021-07-03)
[2] `https://azure.microsoft.com/en-us/overview/what-is-azure` (Accessed on: 2021-07-03)
[3] `https://cloud.google.com/docs/overview` (Accessed on: 2021-07-03)
[4] `https://cloud.google.com/sustainability` (Accessed on: 2021-07-03)

2.4 Architectural Tactics

In this study, we are interested in discovering architectural tactics to increase the energy efficiency of software in the public cloud. In order to understand the definition of architectural tactics, quality attributes need to be well-defined. Hence, we define quality attributes in Section 2.4.1 and architectural tactics in Section 2.4.2.

2.4.1 Quality attributes

A quality attribute can be defined as: *"a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders"* [4]. Quality attributes can be translated into quality attribute requirements to specify the qualification of functional requirements or products. To ensure that the quality attribute requirements are meaningful and testable, Bass et al. [4] define a quality attribute scenario.

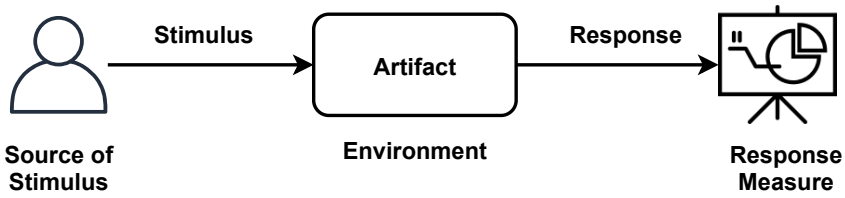


Figure 2.4: Definition of a quality attribute scenario by Bass et al. [4].

A quality attribute scenario consists of six components and is visualized in Figure 2.4. The **Source of Stimulus** (e.g., a user or software) sends a **Stimulus** (i.e., an event) that triggers the system. This system consists of an **Artifact** that exists in an **Environment**. An Artifact is the relevant part of the system and the Environment defines the circumstances of the scenario. The scenario defines the desired **Response** to the Stimulus. To understand whether the Response is adequate, the Response should be observable through a **Response measure**.

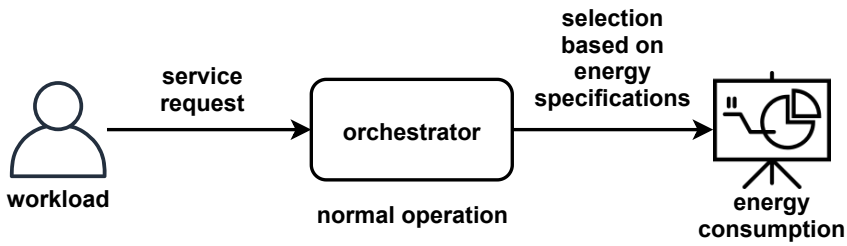


Figure 2.5: Example of a quality attribute scenario for energy efficiency based on the model by Bass et al. [4].

To illustrate, we defined a possible scenario of a Service-Oriented Architecture (SOA) to optimize energy efficiency in Figure 2.5. The workload (Source of Stimulus) requests a service (Stimulus) from an external service provider. Accordingly, the orchestrator (Artifact) needs to select an appropriate service within the pool of available resources. The orchestrator should select a service based on the energy specifications of the service (Response). To assess the impact of the response, the energy consumption (Response Measure) needs to be metered.

Figure 2.6: Illustration of the definition of architectural tactics by Bass et al. [4].

2.4.2 From quality attribute scenarios to architectural tactics

Quality attribute requirements define the response of a system to realize certain objectives [4]. These responses can be achieved through techniques applied by architects. **Architecural tactics** capture these techniques and are formally defined as *"a design decision that influences the achievement of a quality attribute response"* (see Figure 2.6) [4]. In other words, tactics are the techniques that are applied by architects to optimize a system for a quality attribute. To ensure reusability, tactics focus on a single quality attribute response and are not concerned with tradeoffs. Often, multiple tactics exist to realize the same quality attribute, hence, the architect can select the appropriate tactics based on tradeoffs that emerge in combination with other quality attributes. Tactics are not invented but can be derived from practice. In our study, we are interested in tactics to optimize the quality attributes resource efficiency and energy efficiency.

Chapter 3

Related Work

This chapter discusses related studies and the role of our study in the research field. Much research has been published on increasing the energy efficiency of software and hardware in hyperscale DCs [24, 25]. However, these studies assume full control over the infrastructure and can therefore only be applied by cloud providers. In contrast, we did not find peer-reviewed literature that focuses on the perspective of the cloud consumer. Our research aims to fill this gap and explores the role cloud consumers play in increasing the energy efficiency of their workloads running in the public cloud. There are related studies that reflect on the integration of energy efficiency in software architecture. Section 3.1 reflects on energy efficiency as a quality attribute and Section 3.2 presents previously discovered architectural tactics for energy efficiency.

3.1 Energy Efficiency as a Quality Attribute

Kazman et al. [26] performed a case study to explore the management of energy efficiency as an architectural quality attribute. They argue that the behavior of energy efficiency as a quality attribute should be viewed as other more commonly addressed quality attributes such as availability. The authors state that introducing energy efficiency as a quality attribute introduces non-trivial trade-offs such as energy consumption vs. modifiability. They conclude that, through experimentation and prototyping, design decisions can be identified to improve the energy efficiency of an application. Consequently, it is essential that the software is capable of monitoring the energy consumption as it is not possible to optimize your software for a property it cannot measure.

3.2 Tactics for Energy Efficiency

This section discusses related studies that discovered architectural tactics for energy efficiency. The main difference to our study is that we discovered tactics from the industry whereas the related studies discovered tactics from literature reviews. Moreover, we introduce more concrete, practical tactics whereas the tactics from the related studies are at a higher level of abstraction. The most recent and relevant study is from Paradis et al. (Section 3.2.3). The authors introduced a set of reusable tactics which we extend in this thesis.

3.2.1 Jagroep et al.

Jagroep et al. [27] investigated positioning energy consumption in software architecture. The authors classified four green architectural tactics from a literature review:

Increase modularity. Increasing the modularity reduces the size of more frequent database calls. Hence, less CPU power is required to process the call. Note that this tactic assumes that an increase in disk usage has a marginal impact on the energy consumption.

Network load optimization. The downside of the previous tactic is the increase in network load. It is expected that less communication has a positive effect on the energy consumption.

Increase hardware utilization. Inefficient hardware utilization leads to energy inefficiency. Hence, servers should be consolidated allowing idle servers to be switched off.

Concurrency architecture variation. Use the half synchronous concurrency architecture instead of the leader/followers concurrency architecture.

These tactics are relatively high-level and can be contradicting (increase modularity vs. network load optimization). A major difference to our work is that the authors target on-premise software and we focus on software that runs in the cloud.

3.2.2 Procaccianti et al.

Procaccianti et al. [28] discovered tactics for energy efficiency of software in the cloud through a systematic literature review. They defined three categories to classify tactics.

The first category is *Energy Monitoring*. This category contains tactics that gather energy consumption information and present it to the administrator. Tactics in this category are often the prerequisite of tactics from the other categories. An example of a tactic within this category is **Energy Modeling**.

The second category is *Self-Adaptation* which concerns the implementation of mechanisms to modify runtime software configurations to decrease the energy consumption. An example of a tactic within this category is **Consolidation**. This involves consolidating VMs to increase resource utilization. Accordingly, the remaining servers that are idle can be turned off to save energy.

The third and final category is *Cloud Federation*. This category involves tactics that allow cloud-based software systems to select cloud services based on energy consumption information. An example of a tactic within this category is **Service-Adaptation**. This tactic contains an Energy Orchestrator, which discovers and registers energy-efficient services, and a SLA Violation Checker, which ensures that the selected services meet the system objectives.

The study of Procaccianti et al. [28] offers a useful categorization of tactics for energy efficiency. Paradis et al. [5] modified and extended this categorization including a set of reusable tactics. This set of reusable tactics is discussed in the following section.

3.2.3 Paradis et al.

Paradis et al. [5] performed a taxonomic literature review to discover architectural tactics for energy efficiency. The authors classified 10 tactics that provide a basis for architectural design and analysis for energy efficiency. An overview of the tactics is presented in Figure 3.1. The tactics are organized into the categories *Resource Monitoring*, *Resource Allocation*, and *Resource Adaptation*.

The tactics that fall into the category Resource Monitoring are Metering, Static Classification, and Dynamic Classification [5]. The tactic **Metering** concerns real-time data collection of the energy consumption through 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 tactics as it can be applied when real-time data is not available and Static Classification is inadequate. Dynamic Classification considers responds to different workloads and adapts the classification accordingly.

The Resource Allocation category contains the tactics Vertical Scaling, Horizontal Scaling, Scheduling, and Brokering [5]. **Vertical Scaling** concerns adding or activating resources (e.g., CPU or memory) to meet the 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 VM) to the currently available set of resources. A frequently adapted

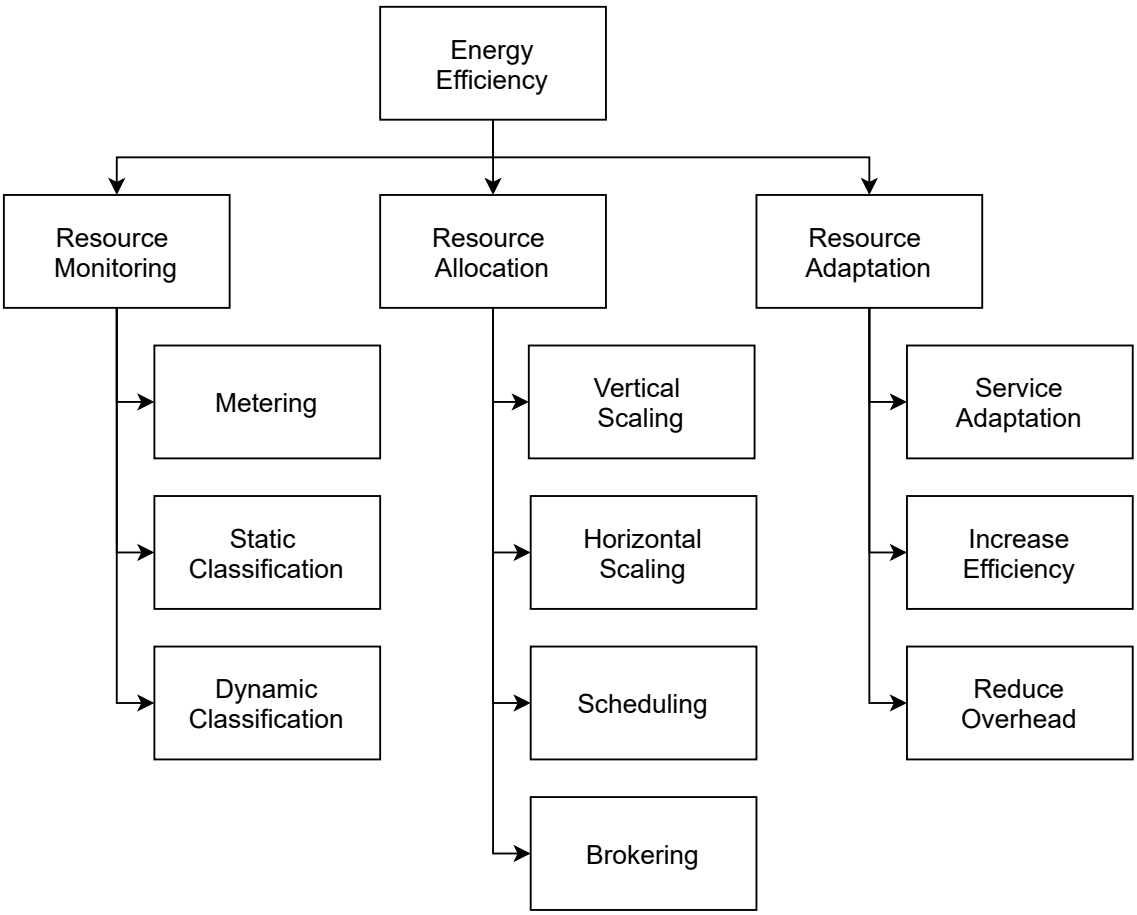


Figure 3.1: Tactics discovered by Paradis et al. [5].

example that falls within this tactic is VM consolidation: where multiple VMs are consolidated onto the same physical server in order to save energy by switching idle servers off. The next tactic, **Scheduling**, involves the allocation of tasks among the resources. This relates to energy efficiency as the tasks can be scheduled to machines to realize optimal vertical and horizontal scaling. Last, the tactic **Brokering** involves including energy-related information to services such that service requests can prioritize services based on energy efficiency.

The final category is Resource Adaptation. The discovered tactics within this category are Service Adaptation, Increase Efficiency, and Reduce Overhead [5]. The tactic **Service Adaptation** extends the Brokering tactic as this tactic involves the selection of services based on their energy efficiency. Next, the tactic **Increase Efficiency** concerns increasing the efficiency of software and algorithms such that it utilizes a minimal number of resources to optimize the energy usage. Finally, the tactic **Reduce Overhead** involves reducing redundant processes. This often introduces a clear trade-off between energy efficiency and other quality attributes.

Paradis et al. [5] stress that empirical research is required to build a framework that reveals the tradeoffs between energy efficiency and other quality attributes. Furthermore, the authors state that the literature lacks research based on the industry. To fill this research gap, we extend their model by adding tactics that are discovered from the industry.

Chapter 4

Study Design

This section discusses the scope and design of the study. Section 4.1 defines the goal and Section 4.2 the research questions. Last, Section 4.3 presents a schematic overview of the study design.

4.1 Goal

The goal of the study is defined according to the template introduced by Wohlin et al. [29] and presented in Table 4.1.

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

Table 4.1: Goal of the study.

In this study, we aim to support cloud consumers in architecting their software workload in the public cloud to optimize for the quality attribute energy efficiency.

4.2 Research Questions

The main research question is: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. As this is a broad question in a relatively new field of research, we divided the study into four sub-questions:

Q1 *What are cloud-native tactics to increase the resource efficiency of cloud consumers?*

Q2 *To what extent can the energy consumption of individual workloads be monitored in the public cloud?*

Q3 *What is the impact of the identified tactics on energy consumption?*

Q4 *How can energy efficiency be defined in the public cloud?*

This research is driven by the assumption that resource efficiency in the cloud leads to energy efficiency. Hence, we start by discovering tactics for resource efficiency (**Q1**) as these tactics are well-adapted in the industry. Afterward, in order to estimate the energy consumption of the identified tactics, a method needs to be defined to monitor the energy consumption in the public cloud (**Q2**). This is not straightforward as most hardware is abstracted from cloud consumers and the available metrics do not directly reflect the energy consumption. Once **Q1** and **Q2** are answered, we perform two case-studies regarding the impact of selected tactics on the energy consumption (**Q3**). Finally, we conduct a first step in defining a metric for energy efficiency in the context of the public cloud (**Q4**).

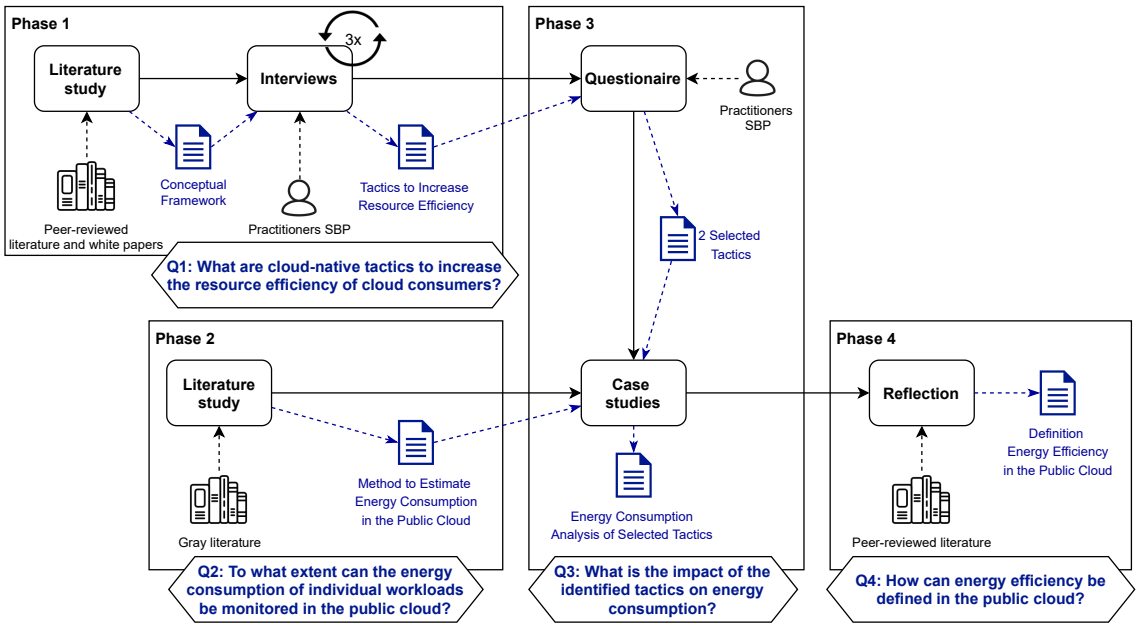


Figure 4.1: Schematic overview of the study design.

4.3 Overview of the Study Design

Figure 4.1 presents an overview of the study design. **Phase 1** aims to answer **Q1**. This is realized through a literature study and interviews with practitioners at SBP. After the initial set of tactics is constructed, we sent the model of reusable tactics together with a questionnaire to the participants to verify the model. After incorporating the feedback, we sent the model and set of tactics to the AWS technical practice lead at SBP for a final iteration of verification. The output of this phase is a set of reusable tactics to increase the resource efficiency of cloud consumers. Moreover, we reflect on the potential impact of the discovered tactics on energy efficiency. This contribution is presented in Chapter 5.

In parallel, **Phase 2** is executed in which we aim to answer **Q2**. This is realized through a gray literature study as no peer-reviewed literature was available on the subject matter. We looked at blog posts from the computer science community to answer this sub-question. The output of this phase is the discovered method to estimate the energy consumption of software running in the public cloud. The method is described in Chapter 6.

Next, in **Phase 3**, we selected two potentially impactful tactics from the set of discovered tactics to answer **Q3**. We conducted two case studies in which we analyzed the impact on the energy consumption of applying edge computing (Section 7.1) and choosing the fitting deployment paradigm (Section 7.2).

Last, **Phase 4** involves a reflection on the definition of energy efficiency in the public cloud to answer **Q4**. This reflection is presented in Chapter 8.

Chapter 5

Architectural Tactics for Energy Efficiency

We observed that energy efficiency is not well adapted as a quality attribute in the industry. Therefore, we could not discover tactics to increase energy efficiency directly. Fortunately, the industry is experienced and knowledgeable in increasing the efficiency of cloud software. We 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 to energy saving. Accordingly, we begin with discovering tactics for resource efficiency and from there on derive potential tactics for energy efficiency. Jagroep et al. [27] argue that software architecture is the right level of abstraction to assess the energy consumption of software and state that research is required to discover a set of tactics to address concerns related to the energy consumption of software. We mainly focused on AWS as SBP has executed many projects in AWS and it is currently the market-leading cloud provider [3]. The methodology of discovering the tactics is described in Section 5.1. We discovered tactics from SBP and white papers of AWS (Section 5.2). Last, we mapped the discovered tactics to the tactics found in the literature and constructed a model of potential tactics for energy efficiency (Section 5.3).

5.1 Methodology of Discovering Tactics

Figure 5.1 presents a detailed overview of the study design of research phase 1.

❶ We start by discussing tactics to increase energy efficiency from the literature. It is important to note that we did not discover the tactics ourselves but reuse already discovered tactics. These tactics were discovered through systematic literature reviews. We

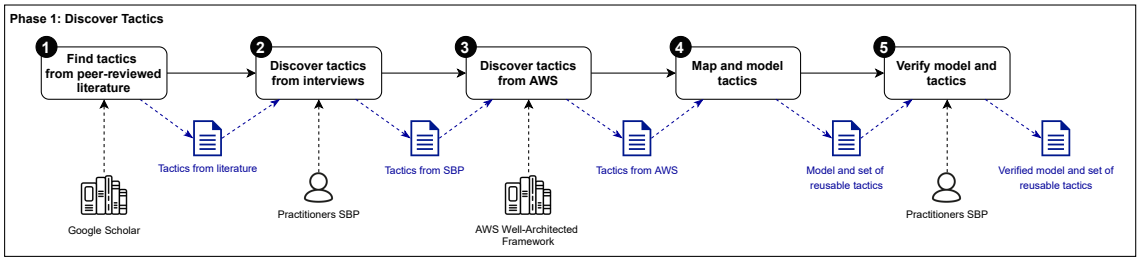


Figure 5.1: Detailed overview of the study design of research phase 1.

obtained the relevant literature by querying Google Scholar for `"Architectural Tactics" AND Energy`. We selected studies based on the Inclusion Criteria (ICs) and Exclusion Criteria (ECs) as listed in Table 5.1. We selected three studies and discuss them in Section 3 (Related Work). In the end, we only embedded one study (from Paradis et al. [5]) into our final model of reusable tactics as their work is the most recent (from 2021) and is buitd on top of the other works. Hence, by selecting their categorization of the tactics for energy efficiency, we automatically incorporate the findings from other related works.

IC1	The study introduces discovered architectural tactics.
IC2	The architectural tactics aim to optimize an energy-related objective.
IC3	The architectural tactics apply to software.
EC1	The study focuses on a specific software type of which the findings cannot be generalized to cloud computing (e.g., Robotics Software).
EC2	The study is not written in English.
EC3	The study is not peer-reviewed.
EC4	The study is in the form of a poster, presentation, tutorial,thesis, or book.
EC5	The study is a duplicate or extension of another study within the review.
EC6	The study is unavailable

Table 5.1: Criteria to select related literature regarding architectural tactics for energy efficiency.

❷ Afterward, we conducted interviews with practitioners from SBP to discover tactics for resource and performance efficiency. The first round of interviews was semi-structured. The objectives are to (1) meet the practitioners and understand their role and expertise, (2) introduce our research to the company, (3) inform whether energy efficiency is currently

adopted as a quality attribute, and (4) create an initial overview of tactics for resource efficiency. We interviewed 17 practitioners with diverse roles such as developer, architect, operation officer, and cloud vendor relation manager. This ensures a broad range of tactics from several levels of abstraction. The transcripts of the interview are not public due to the confidentially of the information. An overview of the respondents is published in Appendix A and an overview of the interview questions is published in Appendix B.

❸ As many practitioners at SBP referred us to the Five Pillars of the Well-Architected Framework (WAF) introduced by AWS, we decided to use this framework to discover more tactics. We read the documentation and derived potential tactics for energy efficiency.

❹ Afterward, we mapped the tactics from the literature to the discovered tactics and created a model of reusable tactics together with their relations.

❺ Last, to ensure the accuracy and completeness of the model and tactics, we asked practitioners at SBP to verify the model and tactics using a questionnaire. Furthermore, we evaluated which tactics are feasible to implement and potentially impactful for energy efficiency. The questionnaire questions are published in Appendix C. Last, we asked the AWS Practice Lead at SBP to review the complete model and set of reusable tactics.

5.2 Architectural Tactics Discovered from AWS

AWS introduces Five Pillars of the WAF [30]. This framework guides cloud consumers in designing cloud applications and workloads. The five pillars are: *Operational Excellence*, *Security*, *Reliability*, *Performance Efficiency*, and *Cost-Optimization*. As of now, energy efficiency is not yet adopted as a pillar. However, we conjecture that energy efficiency does relate to performance efficiency (Section 5.2.1) and cost-optimization (Section 5.2.2) due to more efficient use of resources to run the workload. Hence, we utilize these pillars to discover potential tactics for energy efficiency.

5.2.1 Performance efficiency

Performance efficiency concerns the efficient use of computing resources to meet requirements and maintaining efficiency when demand changes and technologies evolve [30]. The performance efficiency pillar consists of five Design Principles (DPs). We use these DPs to discover potential tactics for energy efficiency. The related tactics are described in detail in Section 5.3.

DP1 *Democratize advanced technologies.* This principle entails that services that do not provide direct business value should be delegated to the cloud vendor [30]. A related tactic is: **Reuse software services (T10)**.

DP2 *Go global in minutes.* This principle involves bringing the workload closer to customers to decrease latency [30]. A related tactic is: **Apply edge computing (T11)**.

DP3 *Use serverless architectures.* This principle depicts removing the operational burden of managing physical servers [30]. A related tactic is: **Chose fitting deployment paradigm (T13)**.

DP4 *Experiment more often.* This principle includes that the cloud enables easy experimentation by providing access to a diverse range of resources without the need to purchase, install, and maintain them [30]. A related tactic is: **Experiment to discover optimal architecture (T1)**.

DP5 *Consider mechanical sympathy.* This principle entails that the use of technology should align with the objectives of the workload [30]. A related tactic is: **Continuously evaluate right sizing (T4)**.

5.2.2 Cost-optimization

The cost-optimization pillar is defined by the ability to run systems to deliver business value at the lowest price [30]. From the related DPs we discovered the following tactics:

DP1 *Implement cloud financial management.* This principle involves the process of investing time and resources to understand and manage the interaction between business value and cloud costs [30]. A related tactic is: **Automatically monitor efficiency per workload (T2)**.

DP2 *Adopt a consumption model.* This principle describes to solely purchase resources that are actually consumed [30]. This means that the used resources should scale according to the business requirements. A related tactic is: **Apply auto scaling (T3)**.

DP3 *Measure overall efficiency.* This principle involves mapping the business value created by a certain workload to the costs to understand the benefit of a certain workload [30]. A related tactic is: **Automatically monitor efficiency per workload (T2)**.

DP4 *Stop spending money on undifferentiated heavy lifting.* This principle involves focusing on business value rather than supporting processes [30]. A related tactic is: **Reuse software services (T10)**.

DP5 *Analyze and attribute expenditure.* This principle encompasses the identification of costs to specific workloads [30]. A related tactic is: **Automatically monitor efficiency per workload (T2)**.

5.3 Model of Reusable Tactics

This section presents the constructed model and describes the discovered tactics. The diagram is presented in Figure 5.2. There are two classes of tactics: tactics for energy efficiency borrowed from the work of Paradis et al. [5] (colored gray) and tactics for resource efficiency discovered from SBP (colored blue). The tactics that are related to the WAF [30], are labeled with the logo ofAWS. The discovered tactics are potentially impactful to increase the energy efficiency of cloud consumers but this still needs to be (empirically) assessed. The discovered tactics are labeled from **T1** - **T18** and described per category: *resource monitoring* (Section 5.3.1), *resource allocation* (Section 5.3.2), and *resource adaptation* (Section 5.3.3).

5.3.1 Resource monitoring

The category *resource monitoring* considers the metering and classification of cloud workloads to optimize the performance [5]. Resource monitoring does not directly affect energy reduction, however, it is a prerequisite for most tactics in the 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 easy to compare different cloud services as opposed to purchasing, installing, and operating similar components on-premise. The measured performance of the different resources for a specific workload can be compared to allow data-driven decisions for the fitting architecture.

Observation. From the interviews, we learned that even though the cloud supports elastic scaling, there are still billing thresholds. For example, in services where a software function consistently runs for 103 [ms] and the billing granularity for the service is 100 [ms], the total cost per execution is 200 [ms]. By optimizing the code

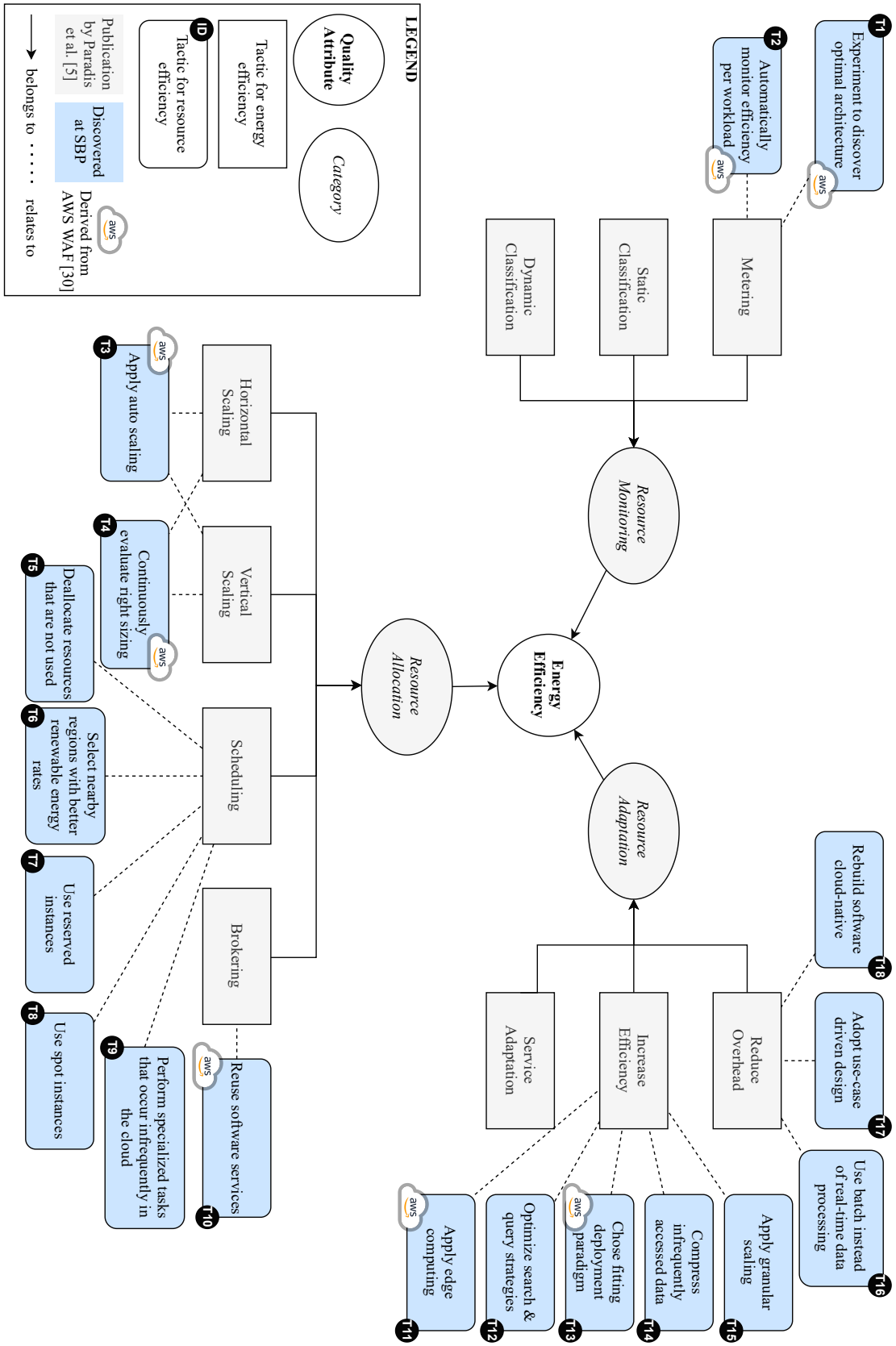


Figure 5.2: Model and Set of Reusable Tactics.

so that the maximum duration is less than 100 [ms] the costs can be halved. We are not sure if this also means that the energy consumption is halved. However, a valid assumption is that the deallocated compute resources can be reused, resulting in increased energy efficiency.

Relation to energy efficiency. Once information about the energy consumption is available in the public cloud, developers could perform experiments to discover the most energy-efficient architecture. Furthermore, targets can be defined to optimize energy efficiency. We expect that these experiments can result in increased energy efficiency. However, it requires business requirements to optimize the energy efficiency and instruments to monitor the energy consumption in the public cloud.

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. Monitoring the energy consumption is a prerequisite for optimizing a workload for energy efficiency.

5.3.2 Resource allocation

The category *resource allocation* involves assigning tasks and workloads to the instances and resources [5].

T3 Apply auto scaling.

Description. Auto scaling[1] involves scaling the application to optimize the performance and costs. The application is monitored and automatically adjusted to ensure stable performance at the lowest costs. This enables on-demand resource usage which is a different approach compared to the traditional method where extra resources are constantly available to be prepared for peak load.

Observation. In the DCs of AWS, extra resources also need to be available to be ready for new requests, however, due to the economies of scale, these can be relatively fewer resources compared to an on-premise architecture. When developing in the cloud, not all developers are adapted to this new approach and still use more resources than necessary to be ready for peak load rather than automatically scaling up.

Relation to energy efficiency. When applying auto-scaling, the number of resources

[1] `https://aws.amazon.com/autoscaling` (Accessed on: 2021-09-05)

used are adjusted based on the workload. Hence, the energy to power the resources is proportional to the workload. Thus, we expect a positive effect of auto-scaling on energy efficiency.

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. Moreover, it involves the identification of opportunities to downsize without compromising capacity or other requirements [1].

Observation. Developers are often specialized in one particular area of the cloud and tend to use these services to solve most problems. The cloud services are very diverse and it is thus important to select the appropriate service for the workload. Furthermore, sometimes it is more cost-effective to increase the resource capacity and accordingly run the software for a shorter time frame[2]. For example, running 2 CPUs with 4 [GB] for 2000 [ms] costs $20, whereas 4 CPUs with 8 [GB] for 500 [ms] costs $10. Possibly, vertical resource scaling and accordingly decreasing the runtime decreases the energy usage as well.

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 the data is stored, impacts the energy consumption.

T5 Deallocate resources that are not used.

Description. Resources that are not used should be deallocated to save costs. This deallocation could be static (at a fixed moment in time) or dynamic (based on the changing environment).

Observation. Selected services might not be required outside office hours (e.g., test environments). Hence, the workload could be architected to automatically deallocate resources during the evening, night, and weekend to save costs.

Relation to energy efficiency. If the workload is not used during a specific moment in time, switching off (idle) resources saves energy.

[1] `https://aws.amazon.com/aws-cost-management/aws-cost-optimization/right-sizing` (Accessed on: 2021/05/09)

[2] `https://www.github.com/alexcasalboni/aws-lambda-power-tuning` (Accessed on: 2021-08-03)

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.

Observation. Many cloud providers are transparent regarding the energy sources and allow users to select regions based on the amount of generated renewable energy. For example, GCP introduces a dashboard to chose a cloud region according to the lowest CO_2 footprint[1]. Note that regions with higher renewable energy rates could be more costly.

Relation to energy efficiency. Strictly speaking, replacing gray energy with green energy does not increase energy efficiency. However, it does decrease the carbon footprint. If the objective to reduce energy is to mitigate the negative effects on the environment, replacing gray energy with green energy is a suitable strategy.

It should be noted that there exists a trade-off between the distance of the region to the end-users and the renewable energy rate. Selecting a region that is far away but offers more renewable energy might not save carbon emissions as more energy is required to transport the data to the end-users. Furthermore, as the distance of a region with more renewable energy increases, the latency might not be acceptable at some point. Hence, a balance needs to be found between the distance to the region and the gain in renewable energy.

T7 Use reserved instances.

Description. Reserved instances[2] are long-term commitments for resource usage at a discounted price.

Relation to energy efficiency. The relation between reserved instances and energy efficiency is not straightforward as the same amount of energy will be consumed by the software regardless of whether the instances are reserved or not. Reserving instances does increase the predictability of the energy that will be consumed. Hence, the cloud provider can provision the required resources and does not need many idle resources that are ready for changing demand. A downside of reserved instances with respect to the energy consumption is that cloud consumers experience fewer incentives to turn off the resources as they are paid for (with discounts) in advance

[1] `https://cloud.google.com/blog/topics/sustainability/pick-the-google-cloud-region-with-the-lowest-co2`
(Accessed on: 2021-08-03)

[2] `https://aws.amazon.com/aws-cost-management/aws-cost-optimization/reserved-instances`
(Accessed on: 2021-07-26)

and turning them off will not provide financial benefits. It would be up to the cloud provider to come up with different incentives for cloud consumers to take the energy efficiency of the reserved instances into consideration.

T8 Use spot instances.

Description. AWS EC2 spot instances[1] allow access to spare EC2 capacity. 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.

Observation. SBP has consulted a medical company with the task to analyze a large dataset with medical data (images). The data analysis is resource-intensive and, therefore, relatively costly. Furthermore, the research was not time-bound and could be stopped at any given moment. These properties made the case fitting for the use of spot instances. By using spot instances, the price of the data analysis was more than halved compared to using regular AWS instances. The downside is that the data processing took relatively longer and would be regularly interrupted, but this was acceptable for this particular workload.

It is important to note that the software needs to be adjusted to be optimally compatible for the use of spot instances. For example, the intermediate program states need to be carefully stored and, whenever new spot instances are released, the program should be capable of resuming from the previously stored state.

Relation to energy efficiency. The energy consumed by software will not reduce when using spot instances. On the contrary, as the software needs to perform extra tasks to be compatible with spot instances (e.g., storing intermediate states), the overall energy consumption of the program might even increase. Nevertheless, spot instances allow the use of spare capacity, instead of requesting AWS to run more physical machines for the on-demand workload. Hence, more output is produced for relatively little extra energy. Thus, using spot instances might have a positive effect on energy efficiency. This effect is, however, difficult to assess as there is little transparency on the internal processes of the spot instances architecture.

T9 Perform specialized tasks that occur infrequently in the cloud.

Description. Specialized tasks that occur infrequently might need specialized hardware. To benefit from economies of scale, it is more efficient to share these resources

[1] `https://aws.amazon.com/aws-cost-management/aws-cost-optimization/spot-instances` (Accessed on: 2021-05-09)

among more consumers.

Observation. Training a ML model is an example of a time-consuming but infrequent task that requires many (specific) resources. Hence, it can be more efficient to perform this task in the cloud as cloud consumers do not need to purchase these specific resources for a workload that is not frequently used.

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.

T10 Reuse software services.

Description. Using services from AWS instead of implementing all functionalities from scratch is often more cost-efficient. The cloud-native software services are developed by specialists and native to the infrastructure.

Observation. An example is the key-encryption functionality. When this service is infrequently used, the difference in cost is negligible. However, when the workload scales up, the potential cost saving increases significantly.

Relation to energy efficiency. It is difficult to assess whether reusing AWS services has a positive effect on the energy efficiency of the workload. This is partly due to the lack of transparency of AWS native services concerning their energy consumption. Moreover, AWS services are versatile and potential effects on energy efficiency should be assessed per case. A potential positive effect on energy efficiency is the fact that the service might be more specialized and native to the architecture which causes less overhead. On the other hand, a service-driven design also introduces overhead due to communication mechanisms and orchestrators. Furthermore, it should be noted that there are also commercial incentives for offering AWS services at competitive prices.

5.3.3 Resource adaptation

The category *resource adaptation* involves changing software and hardware resources to increase the efficiency [5].

T11 Apply edge computing.

Description. Moving computing resources closer to users decreases the latency. 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.
Relation to energy efficiency. Transporting less data over the network is expected to reduce the energy consumption. In Section 7.1, we perform an in-depth case study around the relation between energy efficiency and edge computing.

T12 Optimize search & query strategies.

Description. Search and query strategies can be optimized to increase efficiency and, therefore, reduce costs.
Observation. AWS Athena[1] is a query service to analyze data in Amazon S3[2] using standard SQL. Athena charges for the amount of data being scanned to retrieve the query data. Hence, to optimize the costs of a query, the developer needs to ensure that a minimum amount of data is scanned. For example, whenever a key is frequently positioned at a certain location, first, this location should be scanned to prevent redundant scans.
Relation to energy efficiency. Processing less data results in less computing power and memory usage. Thus, a reduction in energy usage is expected.

T13 Choose fitting deployment paradigm.

Description. The currently most prominent deployment paradigms are VM, container, and serverless architectures. Choosing the fitting paradigm for the workload will optimize the performance.
Observation. There is no one-size-fits-all solution regarding choosing the fitting deployment paradigm.
Relation to energy efficiency. A serverless architecture ensures that services are automatically shut off when they are finished. Moreover, when using a serverless architecture the service utilization is much lower compared to using a VM as the overhead is much smaller. This is expected to have a positive effect on the energy consumption. A possible negative effect of a serverless architecture on energy efficiency occurs in the scenario where the service is frequently called but not constantly on. Starting the service up and down could consume relatively more energy compared to spinning one VM that is constantly on and frequently called. Little research has

[1] `https://aws.amazon.com/athena` (Accessed on: 2021-05-09)
[2] `https://aws.amazon.com/s3` (Accessed on: 2021-05-09)

been conducted on the effect of the deployment paradigms on energy consumption. In Section 7.2, we perform a case study where we compare the energy consumption of a workload that runs on a VM and serverless architecture. The objective of this experiment is to guide cloud consumers in choosing a fitting deployment paradigm with respect to energy consumption.

T14 Compress infrequently accessed data.

Description. Data that is used infrequently should be compressed to optimize the storage costs. In contrast, data that is more frequently accessed might not be efficient to compress as the CPU power that is required to compress and extract the data might cost more than the costs saved of storing a smaller volume of data. Understanding the exact threshold to compress the data depends on the underlying hardware and can be defined through experimentation.

Observation. Compressing large amounts of data that are not frequently accessed can result in major cost savings.

Relation to energy efficiency. In this case, we expect a correlation between cost and energy savings. Whenever less data is stored, less energy is used for storage. The only 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 the workload into smaller components. Accordingly, the used resources can be scaled down into smaller chunks. This results in a better match between the physical resources and the workload.

Observation. The workload needs to be re-architected to be able to scale up and down in smaller granularities.

Relation to energy efficiency. Granular scaling allows a precise match of the physical hardware to the workload. To illustrate, 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 components, one component can be switched off to facilitate the 75% resource utilization. Switching off resources results in cost and energy savings.

T16 Use batch instead of real-time data processing.

Description. If the system objectives allow, transporting data in batches rather than in real-time could optimize the costs as there is less overhead.

Observation. This decision depends on the workload. Something that should be considered is that batch processing involves a fluctuating pattern that requires resource scaling. Real-time processing is more constant and predictable.

Relation to energy efficiency. If the business requirements allow, sending data in batches could reduce the energy as less data needs to be transmitted over the network due to reduced network overhead. Experimentation is required to measure whether this reduction in overhead has a significant effect on energy efficiency.

T17 Adopt use-case driven design.

Description. From the interviews, we learned that often not all used ICT services create direct business value. Use-case driven design is an approach to detect redundant software and data storage. This entails that services that are not used should be turned off or eliminated.

Relation to energy efficiency. By eliminating redundant software and data storage, costs and energy can be saved.

T18 Rebuild software cloud-native.

Description. Instead of migrating existing software, rebuilding the software cloud-native could save costs as the system becomes more efficient and optimally uses the underlying infrastructure.

Observation. This tactic does not apply when the software is not owned by the consumer.

Relation to energy efficiency. It is difficult to generalize the effect of cloud-native software on energy consumption and, hence, should be assessed per case.

5.4 Reflection on Discovered Tactics for Energy Efficiency

The discovered tactics presented in this chapter are selected based on the assumption: *"resource efficiency leads to energy efficiency"*. This selection mechanism should be viewed as a heuristic rather than a rule as resource efficiency in the cloud is often quantified in terms of costs. But resource efficiency is not the only factor that determines the price of a service. For example, marketing, effective development, and scale heavily influence the price as well. Hence, it should be noted that optimizing resources that are quantified in costs, does not always lead to energy efficiency.

Nevertheless, the hypothesis is a useful starting point as we ended up with a relatively large set of concrete tactics. For each tactic, we reasoned about the impact on energy

efficiency. Even so, to prove the impact of the identified tactics on energy efficiency, further (empirical) research is required.

We surveyed the practitioners on which tactics they expect to be most impactful for energy efficiency. Many respondents conjectured that autoscaling (**T3**) and right sizing (**T4**) have a large potential to increase the energy efficiency. This is due to the potential to switch off resources that are not being used. Besides these tactics, many different tactics were mentioned. A majority of the practitioners also expressed their doubts and lack of knowledge regarding the energy consumption to select potentially impactful tactics.

Hence, we decided to select tactics based on the feasibility of the experiments and tactics that concretely guide cloud consumers in architecture decisions. The selected tactics for which we analyze the impact on energy consumption based on case studies are **T11** (Apply edge computing) in Section 7.1 and **T13** (Chose fitting deployment paradigm) in Section 7.2.

Chapter 6

Monitor Energy Footprint in the Public Cloud

Currently, the major cloud providers do not provide direct feedback on the energy consumption of individual cloud workloads [6, 14]. This can be attributed to concerns of cloud providers who aim to prevent disclosure of their financial position and internal architecture [14]. Moreover, AWS introduces the concept of *shared responsibility* where the hardware resources used to support the workloads of individual users cannot be traced for security purposes[1].

From our interviews (Section 5.1), we found that there is little incentive for developers to write energy-efficient code in the public cloud. Developers optimize software based on available metrics. As cloud providers do not provide information regarding the energy consumption, practitioners cannot optimize their workload for this purpose. It is not evident that optimizing for time and costs always leads to energy efficiency. For instance, not all code need to run extremely fast. Perhaps running code slower would be more energy-efficient. However, due to the currently available metrics, this cannot be accurately assessed. This could be solved by adding energy- and sustainability-related metrics to a public cloud dashboard. Accordingly, developers have the opportunity to optimize their code for energy efficiency. Moreover, a sustainability score can be awarded to motivate developers to write energy-efficient code.

For our research, we approached AWS and GCP to collaborate on an experiment to measure the impact of the previously identified tactics (Section 5). Unfortunately, this contact proceeded slowly and we did not manage to schedule a meeting to introduce our research.

[1] `https://aws.amazon.com/compliance/shared-responsibility-model` (Accessed on: 2021-06-07)

Hence, in order to execute an experiment, an alternative method is required to estimate the energy footprint of cloud workloads. We encountered the Cloud Carbon Footprint (CCF) tool that estimates the carbon and energy footprint of individual cloud instances. Section 6.2 describes the CCF tool in detail. The CCF tool is inspired by the Cloud Jewels approach. Hence, we start by introducing the Cloud Jewels approach in Section 6.1.

6.1 Cloud Jewels

Cloud Jewels is an approach to estimate the energy consumption in the public cloud. It is an approach published at a blog from Sommer et al. [14] who are part of the engineering team of Etsy[1]. All information discussed in this section is retrieved from their blog. Sommer et al. explain that they faced difficulties in estimating their energy footprint in the public cloud. To tackle this issue, they defined conversion factors called *Cloud Jewels* that map available cloud usage data to the consumed energy based on public information. The main sources from which information is retrieved are the United States Data Center Energy Usage Report [31], The Data Center as a Computer [2], and the SPEC power report [32]. Several experts in the industry have reviewed the Cloud Jewels method.

The authors share their estimated coefficients for the number of watt-hours consumed by a VM, Hard Disk Drive (HDD), and Solid-State Drive (SSD) in a cloud computing environment. The coefficients are:

```
2.10 Wh per vCPUh [server]

0.89 Wh/TBh for HDD storage [storage]

1.52 Wh/TBh for SSD storage [storage]
```

These point estimates do not include confidence intervals due to the experimental nature of the method. The authors state that they are more likely to over- than underestimate the energy usage to take full responsibility for the consumed energy in the public cloud.

The conversion factor for the sever is based on the formula:

$$\text{Average Watts} = \text{Min Watts} + \text{Server utilization} \times (\text{Max Watts} - \text{Min Watts})$$

To retrieve the min and max watts of a CPU, the average values reported by manufacturers of servers that are likely used by cloud providers based on [32] are used. The server utilization estimate is 45% (as reported in [31]). The conversion factors of storage are

[1] `https://www.etsy.com/about` (Accessed on: 2021-06-07)

estimated using [31] as well. Accordingly, the average capacity of disks and disk wattage are used to estimate the coefficients. In the Cloud Jewel blog, no conversion factors for memory and storage are defined.

6.2 Cloud Carbon Footprint

The Cloud Carbon Footprint (CCF) [6] tool allows the estimation of emitted CO_2 and consumed energy by individual instances in the public cloud. The tool supports several instances from the major cloud providers (AWS, Azure, and GCP). Moreover, the project is open-source and sponsored by ThoughtWorks[1]. All information in this section is derived from the CCF documentation [6].

CCF takes as input usage data (compute, storage, networking) as reported by the cloud provider and estimates the energy [Wh] using the previously introduced conversion factors (see Section 6.1). Next, the estimated energy is multiplied by the PUE to account for the supporting equipment. Furthermore, the carbon intensity of the region in which the DC is located is considered to calculate the carbon emissions. Thus, the carbon emissions are estimated according to the formula:

$$\begin{aligned}\text{Carbon Emissions} = \text{Cloud Provider Service Usage} \times \text{Cloud Energy Conversion Factors} \\ \times \text{Cloud Provider PUE} \times \text{Grid Emissions}\end{aligned}$$

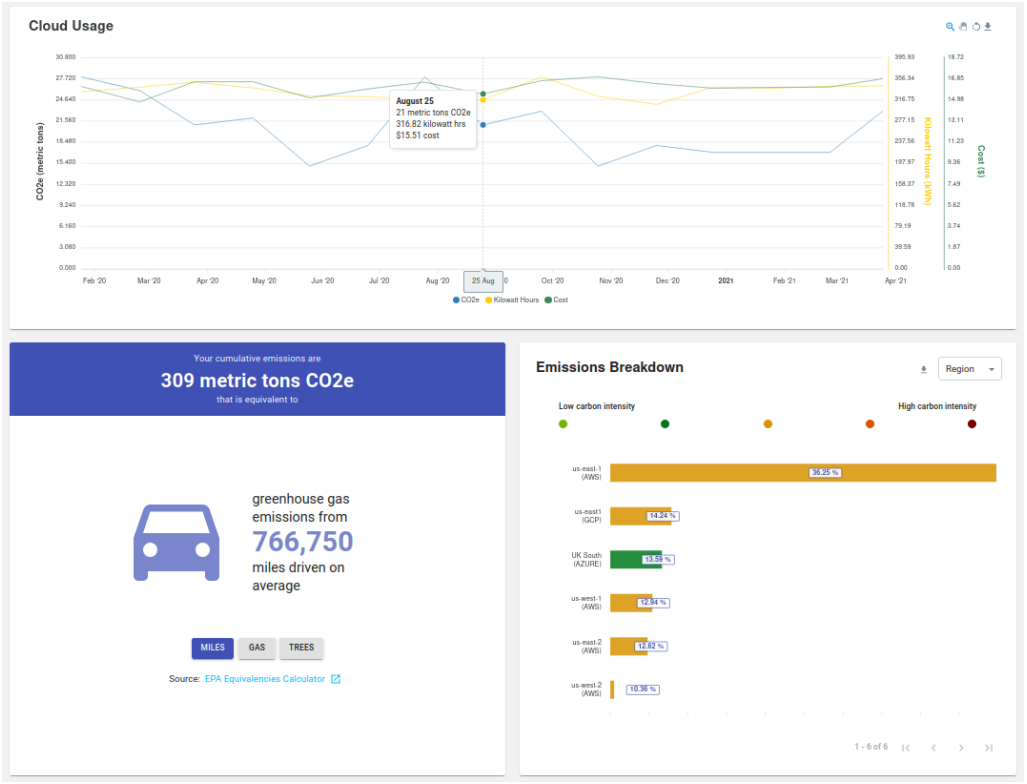
Each part of the formula is explained in the following sub-sections. Exempting the grid emissions from the multiplication results in the energy consumption. The CCF tool offers a user-friendly dashboard that visualizes the costs together with the estimated CO_2 emissions and energy consumption (see Figure 6.1).

6.2.1 Cloud provider service usage

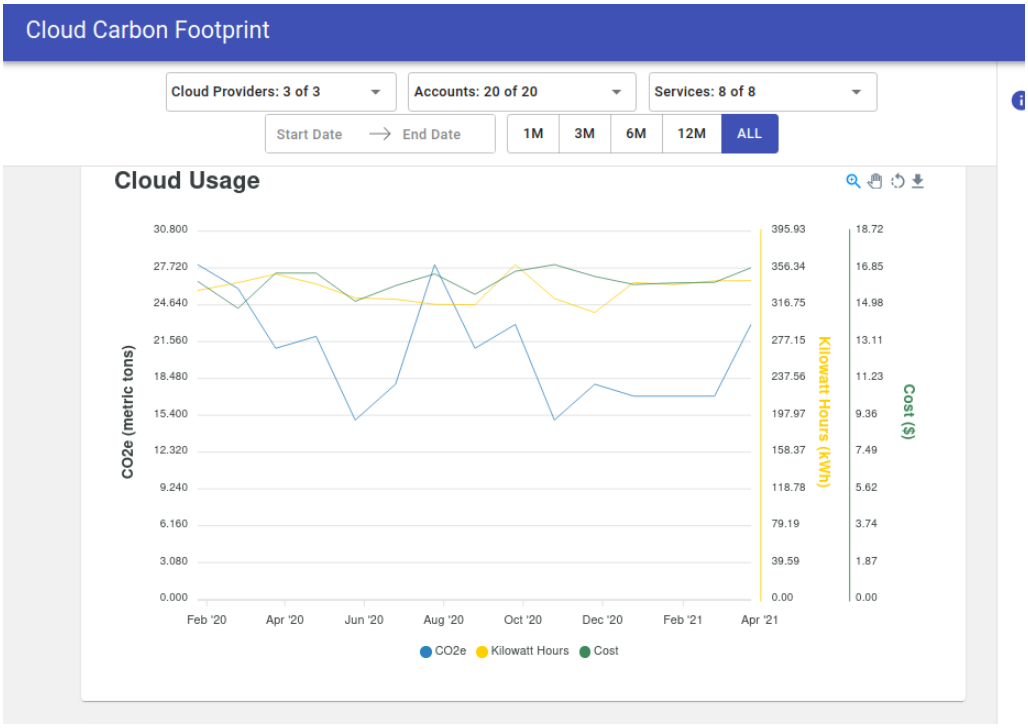
The *cloud provider service usage* involves metrics provided by the cloud provider that are available to the cloud consumers (e.g., vCPU hours). These metrics are used to calculate the cloud costs. The CCF tool offers two approaches to calculate the energy footprint: (1) using billing data (holistic) and (2) using cloud usage APIs (high accuracy).

In this research, we focus on AWS as this is currently the largest cloud provider [3] and many customers in SBP run workloads in AWS. CCF utilizes AWS `CloudWatch` and `Cost`

[1] `https://www.thoughtworks.com/` (Accessed on: 2021-06-07)



(a) Dashboard.



(b) Graph representing the CO_2, energy consumption [kWh], and costs of mocked cloud instances.

Figure 6.1: Screenshots of the CCF tool [6].

`Explore` APIs to retrieve usage and cost data. The supported compute services are `EC2`, `Lambda`, `RDS`, and `Elasticache`. The supported storage services are: `EBS`, `RDS`, and `S3`.

6.2.2 Cloud energy conversion factors

The second part of the formula to estimate the carbon emissions are the *cloud energy conversion factors*. Based on the usage metrics, the energy consumption [kWh] is estimated according to the Cloud Jewels conversion factors (see Section 6.1). The authors of the CCF tool extended the Cloud Jewels methodology for networking and memory. AWS does not disclose the energy information of their custom processors and other hardware. Therefore, an estimation is made based on hardware that is expected to be similar.

Energy conversion for server usage

To estimate the average energy consumed by a cloud server, the minimum watt of an idle server is used and the variable watts are added depending on the server utilization. The vCPU utilization is either pulled from the cloud provider API and otherwise set to 50%. The vCPU hours (variable) are pulled from the cloud usage APIs. Whenever the underlying processor micro-architecture is not available, the min and max wattage are taken from the average of all architectures. For AWS, these averages are: 0.71 (Min Watts) and 3.46 (Max Watts).

Energy conversion for serverless computing

For AWS `Lambdas` (serverless computing), there are no metrics for CPU utilization and vCPU hours available. Hence, the authors of the CCF tool propose a different approach. The memory consumption of a lambda ranges from 128 [MB] and 10240 [MB]. At 1792 [MB], the lambda utilizes one vCPU. Accordingly, the number of vCPUs can be estimated as a ratio of the memory. The watt-hours for a lambda are estimated according to the formula:

$$\text{Total Watt-Hours Lambda} = \text{Average Watts} \times \text{Running Time [hours]} \times \text{Estimated number of vCPUs}$$

where,

$$\text{Average Watts} = \text{Min Watts} + 50\%(\text{Average for hyperscale data centers}) \times (\text{Max Watts - Min Watts})$$

$$\text{Running Time} = \text{Lambda execution time} \,/\, 3600\,[\text{seconds}]$$

$$\text{Estimated number of vCPUs} = \text{Lambda memory allocated (MB)} \,/\, 1,792\,[\text{MB}]$$

Energy conversion for storage

The authors of the CCF tool updated the values from the Cloud Jewels methodology with more recent data. The estimated conversion factors for storage are:

$$\text{Watts per Terabyte for HDD} = \text{Watts per disk} \,/\, \text{Terabytes per disk} = 6.5\,[\text{W}] \,/\, 10\,[\text{TB}] = 0.65\,[\text{Watt-Hours per Terabyte-Hour for HDD}]$$

$$\text{Watts per Terabyte for SSD} = \text{Watts per disk} \,/\, \text{Terabytes per disk} = 6\,[\text{W}] \,/\, 5\,[\text{TB}] = 1.2\,[\text{Watt-Hours per Terabyte-Hour for SSD}]$$

To ensure durability and availability, cloud providers often replicate the data across the DC or another DC. To account for this energy usage, the energy used for storage is multiplied by certain replication factors (for more information see [6]).

Energy conversion for memory usage

There is limited data available on the power consumption of memory systems. The authors of the CCF tool assume that the hyperscale DCs use more efficient memory systems such as DDR4 or DDR5. To define the energy conversion factor, the average is taken of two available coefficients from Crucial[1] (0.375 [W/GB]) and Micron[2] 0.4083 [W/GB]. The used memory coefficient (average of the previously described memory coefficients) is 0.000392 [kWh/GB].

For AWS, the kilo-watt hours for memory are calculated according to the formula:

$$\text{Kilo-Watt Hours Memory} = \text{Memory [GB] exceeding SPECPower database average} \times \text{Memory coefficient} \times \text{Usage amount [hours]}$$

Energy conversion for networking

At the moment, the CCF tool only accounts for the energy consumed by communication across co-located DCs. The authors assume that the electricity used by the internal network is negligible compared to the power used by the servers. Furthermore, the energy consumed to deliver the cloud services to the end-users is not considered as this is often delivered by

[1] `https://www.crucial.com/support/articles-faq-memory/how-much-power-does-memory-use` (Accessed on: 2021-07-17)

[2] `https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn4007_ddr4_power_calculation.pdf` (Accessed on: 2021-07-17)

external Content Delivery Network (CDN) providers. The authors of the CCF tool assume that the hyperscale DCs have an efficient network between their DCs and exchange data at a high bitrate. Hence, the smallest available coefficient is assumed to estimate the energy consumption of data transport among co-located DCs, namely, 0.001 [kWh/GB].

6.2.3 Cloud provider Power Usage Effectiveness (PUE)

The average PUE of AWS is 1.135. The estimated energy is multiplied by the PUE to account for the energy consumed by supporting equipment.

6.2.4 Grid emissions factors

After estimating the consumed energy from the cloud services, the emitted metric tons CO_2 can be calculated. This is out of scope for our current research but important to mention for the interested reader as reducing the CO_2 emissions is often a main incentive to monitor and reduce the energy consumption. The source of the electricity is required to accurately estimate the CO_2 emissions of the consumed energy. As these are not directly published by AWS, region estimates from open data sources are considered to calculate the CO_2 emissions per region.

6.3 Reflection on Monitoring the Energy Footprint in the Public Cloud

The CCF method is a valuable first step in measuring the energy consumption of public cloud instances. However, it is an estimation and future research is required to refine the method. The CCF method has some limitations. For example, the method does not attribute for certain software overhead in the cloud such as hypervisors, orchestrators, and other cloud services if they are not included in the AWS billing statement. Other supporting services, such as `IAM` functionalities are not included in the billing statement either and therefore not included in the energy estimations. For more accurate metrics of the energy consumption of cloud services, we urge public cloud providers to share internal models and publish relevant data to their cloud consumers. In the next section, we perform two case studies in which we apply the CCF methodology to estimate the energy consumption for two tactics, namely, edge computing (Section 7.1) and VM vs. serverless (Section 7.2). For the edge computing case, the energy consumption of data transport over the Internet plays a significant role. Hence, we extend the CCF method to account for the energy consumption of transporting data over the Internet.

Chapter 7

Case Studies

After discovering the tactics and establishing a method to estimate the energy consumption in the cloud, we analyze two tactics in-depth to reason about their impact on the energy consumption. The first tactic we selected is **T11** (Apply edge computing) and is described in Section 7.1. Afterward, we compare the computing paradigms VM and serverless with respect to energy consumption according to tactic **T13** (Choose fitting deployment paradigm) in Section 7.2.

7.1 Edge Computing

The first in-depth analysis is based on tactic **T11** (Apply edge computing). Edge computing is defined as

> *"the enabling technologies allowing computation to be performed at the edge of the network; on downstream data on behalf of cloud services and upstream data on behalf of IoT services"* [33].

In this context, *edge* refers to computing and network resources on the path between the data source and the cloud data center. This involves sending the generated data to the cloud (upstream) and sending data back from the cloud (downstream). An example of edge computing is degrading the quality of a photo on a mobile device before it is uploaded to a social network to reduce network traffic [33]. 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 [33].

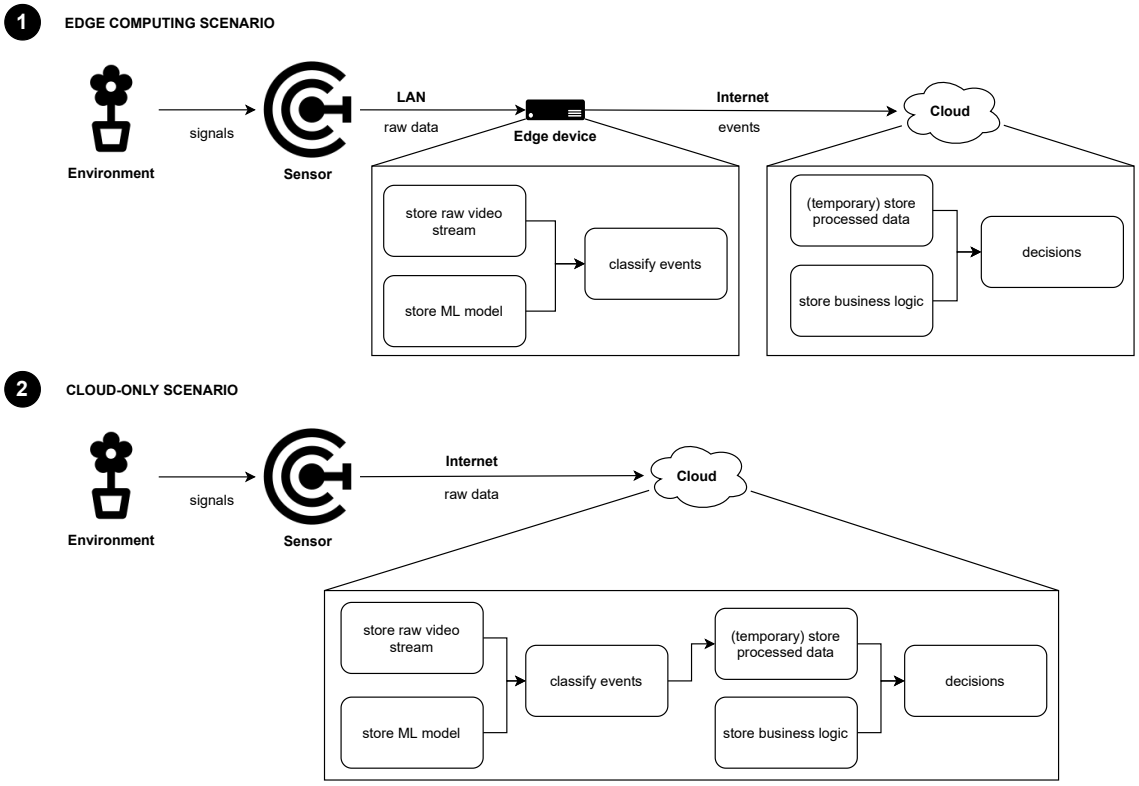


Figure 7.1: Edge computing case overview.

7.1.1 Edge computing case study description

SBP has consulted a start-up company that provides technology for fall detection. They developed a service that, based on camera images, detects whether a person fell down using ML classification. The case is visualized in Figure 7.1. There are two scenarios possible: (❶) using edge computing or (❷) adopting a cloud-only strategy.

In scenario ❶, the environment (e.g., a bedroom in a nursing home) is captured by a sensor. In this case, the sensor is a camera from which the raw video data is transmitted to an edge device over a Local Area Network (LAN). The edge device stores the ML model and classifies the incoming video data on whether a person fell down or not. Only when a fall is detected, will selected data be sent to the cloud. Accordingly, based on implemented business logic, the system will react to the event. 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. An alternative is visualized in scenario ❷. In this scenario, the raw video data is directly transferred to the cloud. In this case, all steps (from classification to decision-making) are performed in the cloud.

The sketched scenarios, together with the trade-off edge vs. cloud-only, are not limited to fall detection but can also be applied to other cases where sensors are utilized to detect events in an environment. Examples are detecting whether restricted areas are entered or appropriate safety protection is worn (e.g., masks or helmets).

From a privacy perspective, applying edge computing ensures that privacy-sensitive data (e.g., images within a home) are not sent over the Internet and stored on location rather than in a public cloud. Hence, data owners experience more control and data can be removed directly whenever deemed appropriate. Furthermore, sending a large volume of raw data over the Internet is a time-consuming task. Hence, it is more efficient to restrict data transport to classified events as opposed to a raw data stream.

In this analysis, we investigate the sketched scenarios from an energy perspective. SBP provides access to descriptive and performance data of the fall detection system. Using the available data, we estimate the impact of applying edge computing on the energy consumption. This analysis contributes to answering the question: *"What is the impact of applying edge computing on the energy consumption?"*. For this analysis, we zoom into the areas that are distinctive in both scenarios (edge vs. cloud-only) as introduced in Figure 7.1. The relevant components of which we estimate the energy consumption are shown in Figure 7.2. In the edge computing scenario (❶), we consider the energy consumed by the edge device and the data transfer over the Internet. In the cloud-only scenario (❷),

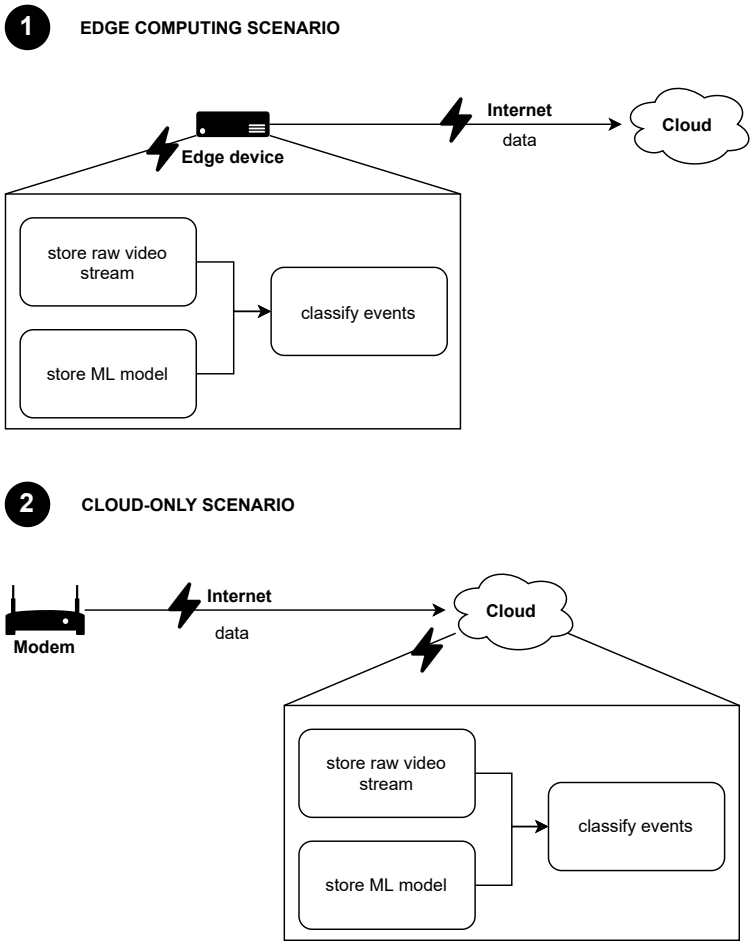


Figure 7.2: Considered components of which we estimate the energy consumption in the edge computing case.

we consider the energy consumed by the data transferred over the internet and the energy consumed in the cloud (of the tasks that are otherwise conducted on edge). The following two sub-sections show the estimates of the energy consumption in the edge scenario (Section 7.1.2) and cloud-only scenario (Section 7.1.3). Last, Section 7.1.4 discusses the results and limitations of the method.

GPU	512-core Volta GPU with Tensor Cores
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
Memory	32GB 256-Bit LPDDR4x | 137GB/s
Storage	32GB eMMC 5.1
DL Accelerator	(2x) NVDLA Engines
Vision Accelerator	7-way VLIW Vision Processor
Encoder/Decoder	(2x) 4Kp60 | HEVC/(2x) 4Kp60 | 12-Bit Support
Size	105 mm x 105 mm x 65 mm
Deployment	Module (Jetson AGX Xavier)

Table 7.1: Jetson AGX Xavier specifications.

7.1.2 Estimate energy consumption edge computing scenario

This section covers the energy consumption estimate of the edge computing scenario as illustrated in Figure 7.2 (❶). A single camera generates 4.1 Megabit/second (mbit/s) and sends this data over a cable to the edge device. Eight cameras can be connected to the edge device to realize full hardware utilization. Hence, $8 \times 4.1 = 32.8$ mbit is generated every second and sent to the edge device. This data is transferred through a cable. We ignore the energy consumption of the cable that connects the camera to the edge device as it is comparable to transferring the data from the camera directly to the modem in the second scenario. The edge device considered is the `NVIDIA Jetson AGX Xavier`[1]. The specifications of this device are listed in Table 7.1.

The documentation of the edge device[2] specifies that there are three different power modes, namely, 10 W, 15 W, and 30 W. In the current scenario, the resources of the edge

[1] `https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit` (Accessed on: 2021-07-06)

[2] `https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3243` Section: Clock Frequency and Power Management (Accessed on: 07/07/2021)

device are fully utilized and the power mode is 30 W. Due to the full hardware utilization and continuous data stream from the cameras, we reason that the power is relatively stable over time. The energy consumption [kWh] of the edge device for one day [24 hours] can be calculated using the following formula:

$$E_{[kWh]} = \frac{P_{[W]} \times t_{[hours]}}{1000} = \frac{30 \times 24}{1000} = 0.72\,[\text{kWh}]$$

where E is the energy consumption in kWh, based on the power (P) in W and time (t) in hours. In the fall detection case (i.e., being attached to 8 cameras), the **edge device** consumes **0.72 kWh** in 24 hours.

The next step is to calculate the energy consumption of the data sent over the Internet from the edge device to the cloud. Only when an event is detected, will data be sent to the cloud. The events concern relatively small messages of 2.4 KB each. Each day, on average, 20 status updates are sent to the cloud from each room (i.e., each camera). These status updates contain the detected event such as `person fell down` or `person went in bed`. In this case study, we consider 8 cameras as these result in full resource utilization of the edge device. Each room has one camera, accordingly, in one day, 8 [cameras] × 20 [messages] × 2.4 [KB]= 384 [KB] = 0.000384 GB of data is sent from the edge device to the cloud. To calculate the energy consumption of transferring this data over the Internet, we use the electricity intensity rate of 0.06 kWh/GB as defined by Aslan et al. [34] (the electricity intensity is further explained in the next section). Consequently, the energy consumption of **transporting the status updates from the edge device to the cloud over the Internet** for 24 hours equals 0.000384 [GB] × 0.06 [kWh/GB] = **0.00002304 kWh**.

7.1.3 Estimate energy consumption cloud-only scenario

In order to estimate the energy consumption of the cloud-only scenario, two sub-problems are identified: first, the energy consumption of transmitting the video data over the Internet needs to be estimated. Second, the energy consumption of the ML classification in the cloud needs to be estimated.

Many studies attempted to estimate the energy consumption of the Internet. This is a difficult task as the Internet is a large and complex system [34]. Aslan et al. [34] reviewed 14 studies to estimate the electricity intensity of the Internet. Electricity intensity is defined as: *"energy consumed per amount of data transmitted"* and measured in kWh/GB [35]. Hence, this metric is useful for our study to estimate the energy consumption of transferring the video data over the Internet to the public cloud. Aslan et al. [34] estimate

that sending data over the Internet consumes approximately 0.06 kWh/GB in 2015. This is the most recent and accurate estimate we found in the peer-reviewed literature.

From the cameras, 32.8 mbit of data is generated each second. In our analysis, we consider the energy consumption for one day [24 hours]. Hence, in a day, $32.8\,[\text{mbit}] \times 60\,[\text{minute}] \times 60\,[\text{hour}] \times 24\,[\text{day}] = 2,833,920\,[\text{mbit}]$ of data is generated. 1 mbit is equal 0.000125 GB. Hence, 2,833,920 mbit is equal to $2,833,920 \times 0.000125 = 354.24\,\text{GB}$. Accordingly, we estimate the energy consumption of **transferring the camera stream from the edge device to the cloud over the Internet** for 24 hours to be approximately $345.94\,[\text{GB}] \times 0.06\,[\text{kWh/GB}] =$ **20.7564 kWh**.

Container ID	CPU %	Memory usage
1	40.19	218.2 MiB
2	40.61	219.8 MiB
3	0.90	479.1 MiB
4	0.03	1.185 GiB
5	0.02	2.62 GiB
6	0.49	46.43 MiB
7	2.14	23.64 MiB

Table 7.2: Runtime resource usage of the image processing and classification software.

Next, we need to estimate the energy consumption of the classification in the cloud. For this, we use the methodology introduced by the CCF tool [6] and Cloud Jewels method [14] which are described in detail in Sections 6.1 and 6.2 respectively. The data processing and classification are deployed in containers. The CPU and memory usage for each container are shown in Table 7.2. We use this data (together with the hardware specification from the edge device as listed in Table 7.1) to estimate the energy consumption of the same workload in AWS for 24 hours. Note that this is an estimation of the workload as we do not have the same program running in AWS.

Running the program for one day results in 24 [hours] × 8 [cores] = 192 vCPU hours. Each threat is represented by a vCPU[1] and the edge device contains 8 cores running one threat each. The minimum and maximum watts for a vCPU in AWS are 0.71 and 3.46

[1] `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-optimize-cpu.html` (Accessed on: 2021-08-10)

respectively. We calculate the energy consumption of computing according to the formula introduced by the CCF tool (see Section 6.2.2):

$$\text{CPU utilization} = 40.19 + 40.61 + 0.90 + 0.03 + 0.02 + 0.49 + 2.14 = 84.38\% \text{ (see Table 7.2)}$$

$$\text{Average Watts} = \text{Min Watts} + \text{Avg vCPU Utilization} \times (\text{Max Watts - Min Watts}) =$$

$$0.71 + 0.8438 \times (3.46 - 0.71) = 3.03045$$

$$\text{Compute Watt-Hours} = \text{Average Watts} \times \text{vCPU Hours} =$$

$$3.03045 \times 192 = 581.8464\,[\text{Wh}] \approx 0.5819\,[\text{kWh}]$$

The energy consumption of the GPU need to be considered as well. When connected to 8 cameras, container 5 (from Table 7.2) utilizes 90% of the GPU of the edge device (for specification see Table 7.1). 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, namely 18 W [1]. Accordingly, in 24 hours, the expected energy consumption of the GPU is:

$$E_{[kWh]} = \frac{P_{[W]} \times t_{[hours]}}{1000} = \frac{18 \times 24}{1000} = 0.432\,\text{kWh}$$

Next, we calculate the energy consumption of the memory usage. The total used memory can be calculated using Table 7.2. The memory is reported in Gibibyte (GiB) and Mibibyte (MiB). The total memory usage equals:

$$218.2\,[\text{MiB}] + 219.8\,[\text{MiB}] + 479.1\,[\text{MiB}] + 46.43\,[\text{MiB}] + 23.64\,[\text{MiB}] = 987.17\,[\text{MiB}]$$

$$1.185\,[\text{GiB}] + 2.62\,[\text{GiB}] = 3.805\,[\text{GiB}] = 3896.32\,[\text{MiB}]$$

$$987.17\,[\text{MiB}] + 3896.32\,[\text{MiB}] = 4883.49\,[\text{MiB}]$$

Converting this results in 4883.49 [MiB] $\approx$ 5.12 GB. According to the formula used by the CCF tool (see Section 6.2.2), the energy consumption of the memory is equal to:

$$\text{Memory [GB]} \times \text{Memory coefficient} \times \text{Usage amount [hours]} =$$

$$5.12\,[\text{GB}] \times 0.000392\,[\text{kWh/GB}] \times 24\,[\text{hours}] \approx 0.0482\text{ kWh.}$$

[1] https://www.tomshardware.com/reviews/geforce-radeon-power,2122-4.html (Accessed on: 2021-08-14)

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] × 1.135 [PUE] ≈ 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**.

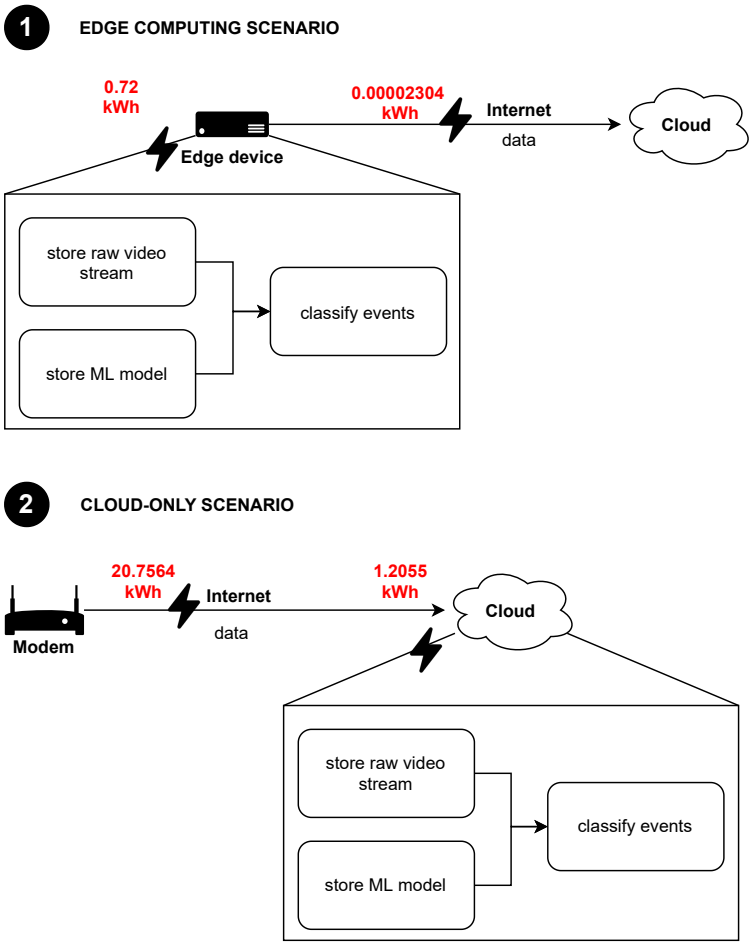


Figure 7.3: Estimated energy consumption for the edge computing and cloud-only scenarios for 24 hours.

7.1.4 Discussion energy consumption edge computing vs. cloud-only

The results of this case study are presented in Figure 7.3. According to our estimations, the energy consumption of the data processing and ML classification is relatively similar in the edge and cloud-only scenarios. The energy consumption of the cloud-only scenario is a bit higher, this can be explained by the edge device being specialized and optimized for this

specific purpose. It should be noted that the estimated energy consumption in the cloud is theoretically calculated and not the result of direct measurements. Hence, the result is an approximation without confidence intervals due to the experimental nature of the used sources. Further research is required to refine these estimates. However, to accomplish the refinements, cloud providers need to provide more information regarding the energy consumption of individual workloads in the public cloud. For now, we can assume that in the considered fall detection scenario, the computation on edge versus in the cloud is not a significant factor with respect to the energy consumption.

The energy consumption of the data transport, on the other hand, differs several orders of magnitude when comparing the on edge versus cloud-only scenarios. The reduction in data transport is also a main motivation for applying the edge architecture in the fall detection scenario. Transmitting the video data of 8 cameras for 24 hours directly to the cloud is estimated to consume 21 kWh of electricity. This energy consumption is significantly larger compared to the edge scenario in which only signals (e.g., "person fell down") are sent to the cloud.

Therefore, we argue that, in a scenario where large volumes of data need to be processed, applying an edge architecture has a positive effect on the energy consumption of the workload. It should be noted, however, that our method has limitations due to the theoretical nature of the calculations. Moreover, the considered real-world environment introduces several insecurities. These insecurities are due to the complex nature and lack of transparency when considering the Internet and the public cloud which do not provide visibility of the internal architectures and routes. Further research is required to perform empirical measurements. Unfortunately, this is currently not possible due to the complexity of measuring the electricity intensity of the Internet and the lack of information provided by cloud providers with respect to the energy consumption.

7.2 Virtual Machine vs. Serverless

The second analysis that we conduct is concerning tactic **T13** (Choose fitting deployment paradigm). In this analysis, we focus on comparing the deployment paradigms VM and serverless with respect to energy consumption. When organizations decide to migrate their software to the public cloud, cloud consumers need to decide upfront which deployment paradigm to embed as it is costly and time-consuming to change the deployment paradigm after launching the application. Hence, with this analysis, we aim to support cloud consumers in choosing the fitting deployment paradigm. We designed an experiment to measure the impact of the deployment paradigm on the energy consumption. Furthermore, we conduct a theoretical analysis on the impact of selected deployment paradigms on the energy consumption based on the literature, documentation of AWS, and interviews with practitioners at SBP. This analysis contributes to answering the question: *"In which scenarios is it more energy-efficient to deploy a workload using VMs or a serverless architecture?"*. First, we define the deployment paradigms VM and serverless in Sections 7.2.1 and 7.2.2 respectively. Afterwards, we propose the experiment design in Section 7.2.3 and conduct the theoretical analysis in Section 7.2.4. We conclude this section with a reflection on the lessons-learned in Section 7.2.5.

7.2.1 Virtual machine

Virtualization is a popular computing technique where a complete computer is abstracted in software. This abstraction includes a simulation of the hardware and OS [36]. In a cloud computing context, VMs are a fundamental building block that provides cloud consumers with access to remote cloud computing resources. Moreover, virtualization is the foundation of the multi-tenant model where multiple cloud consumers share computing resources to increase the efficiency and benefit from economies of scale [11]. Providing VMs to cloud consumers is part of the IaaS classification of cloud computing where developers experience most control over the application code and operating infrastructure [37].

AWS provides access to VMs through the compute service `Elastic Compute Cloud (EC2)`[1]. EC2 is a virtual infrastructure service where computing capabilities are provisioned on-demand. This reduces the overhead of infrastructure management as opposed to an on-premise infrastructure. EC2 instances include an OS and virtual copy of the hardware.

[1] `https://aws.amazon.com/ec2` (Accessed on: 2021-08-12)

7.2.2 Serverless

Serverless computing emerged from advances around the computing paradigms VM and containerization. VMs are an initial abstraction over the hardware, containers introduce a further abstraction, and serverless computing abstracts most of the underlying infrastructure away from the cloud consumer. Serverless computing can be defined as:

> *"a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs; it is by no means an indication that there are no servers, simply that the developer should leave most operational concerns such as resource provisioning, monitoring, maintenance, scalability, and fault-tolerance to the cloud provider"* [37].

Subsequently, the serverless paradigm can be classified as PaaS, in which the infrastructure is abstracted away from the cloud consumers. Serverless platforms are designed as event processing systems where the cloud consumer defines a set of functions and the platform executes these functions based on events. This is also referred to as Function as a Service (FaaS). Functionality-wise serverless and other applications are similar. Hence, deciding on a certain paradigm mainly depends on non-functional requirements such as the amount of control or costs [37].

An advantage of serverless computing from the perspective of cloud consumers is more time to focus on the actual code and business logic rather than managing and provisioning servers. A disadvantage of serverless computing for cloud consumers is restricted capabilities and control as serverless computing abstracts the underlying infrastructure away from the cloud consumer. Moreover, developers need an appropriate understanding of the underlying platform to design the serverless application well. This can be challenging as serverless platforms are fundamentally different as opposed to traditional computing paradigms [37].

An advantage of serverless computing from the perspective of the cloud provider is to reduce operational costs as a result of economies of scale. Furthermore, most cloud providers offer a large ecosystem of services to monitor and extend serverless applications. This results in more business opportunities to deliver services and gain more control over the entire development stack. For cloud consumers, this ecosystem of services introduces several opportunities but also a dependence on the provider's ecosystem and risk of vendor lock-in [37].

AWS introduces the service `Lambda`[1] for serverless computing. Lambda is a compute service that enables code deployment without the need to provision and manage servers. Moreover, services such as auto-scaling and monitoring are automated. Using Lambda, the complete code base is organized into Lambda functions which are only called when an event occurs. The cloud consumer solely pays for functions that are called and idle functions waiting for events are free of charge. The Lambda functions scale automatically with the number of events (e.g., user requests). When the function is invoked for the first time, AWS Lambda creates an instance that processes the event. After the event has been processed, the instance remains active to process other events. Whenever two events occur simultaneously, AWS lambda sets up another instance to process the event concurrently. As the number of requests increases, AWS lambda routes the events to the available instances and adds new instances if required. Whenever the demand decreases, the unused instances will be freed [38].

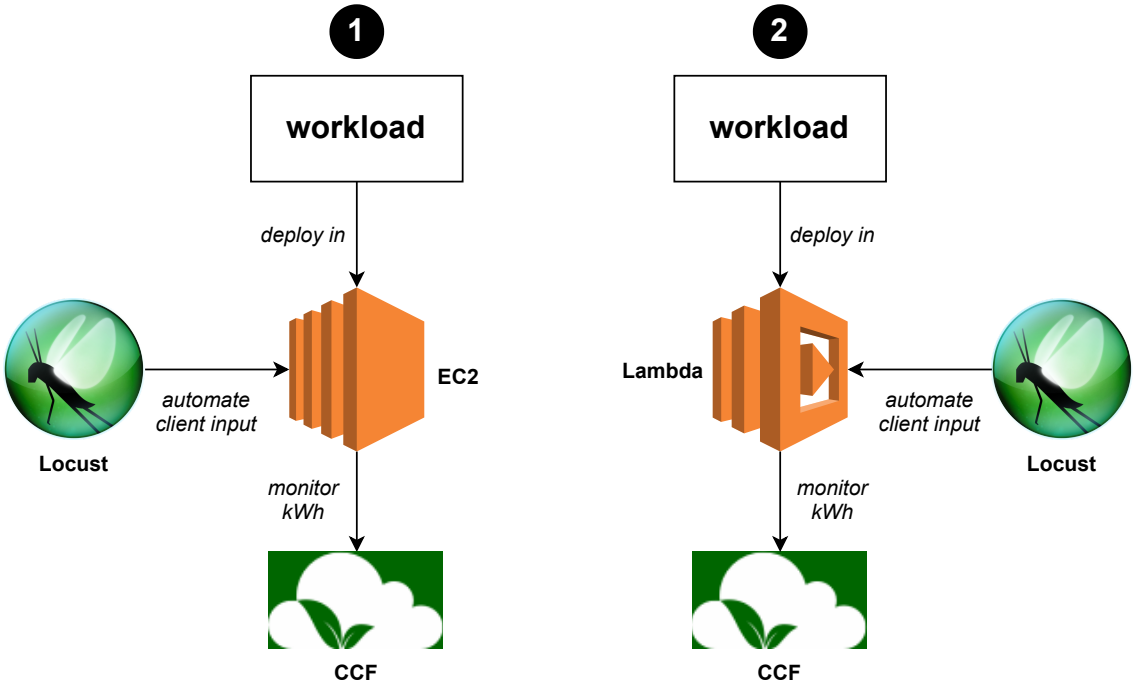


Figure 7.4: Proposed experiment design to assess the impact of VM vs. serverless on energy consumption.

[1] `https://aws.amazon.com/lambda` (Accessed on: 2021-08-01)

7.2.3 Proposed experiment design energy consumption virtual machine vs. serverless

We designed an experiment to assess the impact of the deployment paradigm on the energy consumption. The experiment design is presented in Figure 7.4. The objective is to compare two similar workloads (i.e., the same functional requirements) and deploy one version using VMs (EC2) and the other serverless (Lambda). To capture real-world behavior of the workload, Locust[1] can be used to simulate user requests. Eventually, the energy consumption [kWh] can be estimated using the CCF tool [6].

Unfortunately, we did not execute this experiment due to time constraints. We searched for two comparable workloads, from which one is deployed using VMs and the other serverless, on the Internet and within SBP. However, we could not find two programs that were similar enough to execute a sound experiment. Besides, we ran small serverless web applications[2], however, the workload was too small to estimate the energy consumption using the CCF tool. The last option is to develop both programs from scratch to ensure similarity and a large enough workload to measure the difference in energy consumption. This is out of scope for the current thesis, but an interesting contribution to retain more insight into the impact of the deployment paradigm on the energy consumption.

A limitation of a certain experiment is that the effect of the deployment paradigm on the energy consumption is heavily dependent on the workload type and request frequency. Hence, in order to gain a thorough understanding of the difference between VMs and serverless, multiple workloads need to be assessed. We tackle this limitation by providing a theoretical analysis that considers the general behavior of workloads. Yet, more experiments are required to support the claims.

7.2.4 Theoretical analysis energy consumption virtual machine vs. serverless

This section presents a theoretical analysis around the impact of the deployment paradigms VMs and serverless on the energy consumption based on the AWS Lambda documentation [38], study by Baldini et al. [37] around the serverless paradigm, and interviews with practitioners at SBP. From these works, we identified six principles (**P1** - **P6**) of the deployment paradigms that are likely to affect the energy consumption. Similar to identifying

[1] `https://locust.io` (Accessed on: 2021-08-11)

[2] e.g., `https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito` (Accessed on: 2021-08-11)

the tactics from Chapter 5, we used the assumption *"resource efficiency leads to energy efficiency"* as a starting point.

P1 Runtime. Lambda functions timeout after 15 minutes. Hence, for programs that are compute-intensive and require long execution times, Lambda functions are not recommended as restarting after 15 minutes requires storing the intermediate state and adapting the program to deal with interruptions. Moreover, there exists a risk of losing compute results due to the timeout. The use of extra software to manage the timeout and risk of data loss have a negative impact on the resource efficiency and thus energy efficiency. We would like to add that the timeout value can be adjusted by the cloud consumer. Hence, it is important for cloud consumers to experiment to retrieve the optimal timeout value to prevent wasting energy on a Lambda function that has finished its operation. In general, programs that require a long computation time are more efficient in EC2, mainly due to the timeout period.

P2 Request frequency. If the request frequency is low, a VM will run idle when waiting for requests. Running idle resources consumes energy without contributing to the operation and hence has a negative effect on the energy efficiency. In contrast, Lambdas are only activated when the specified events occur. Thus, for a low number of requests, Lambdas are expected to be more energy-efficient as they only use resources when called for. On the other hand, when the request frequent is high (and stable), EC2 might be preferred as starting up and releasing Lambdas regularly might have a negative effect on the energy efficiency as the start-up time to invoke a Lambda might consume more energy than running a stable EC2 instance.

P3 Load predictability. This principle is related to the request frequency. Lambdas are more efficient for an unpredictable workload as no computing resources are used when the functions are idle. Moreover, the compute instances auto-scale with the load. When the load is predictable, EC2 might be more efficient as a single EC2 instance can be fully utilized and there is no need to spend energy on constantly invoking and terminating Lambda functions.

P4 Overhead. A possible downside for the energy efficiency of VMs is the overhead of the hardware, OS, and even shared software packages. If the program has to execute

small tasks, it is not efficient to run a whole OS alongside this as this consumes relatively much energy. Serverless functions are more lightweight and multiple functions share the same resources, thus this could be more energy-efficient.

P5 Orchestration. As opposed to VMs, Lambdas consist of relatively small units. These units need to be scheduled and orchestrated by dedicated software units. This can therefore result in energy overhead. As VMs consist of larger units of a longer life-span, less scheduling and supporting activities are required.

P6 Auto-scaling. With AWS Lambda, the computing resources scale up and back down automatically based on real-time demands. This is energy-efficient as all the resources are contributing to the operation. When using VMs, auto-scaling can also be achieved, however, this is not automated and therefore more error-prone.

7.2.5 Reflection energy consumption virtual machine vs. serverless

In this chapter, we compared the deployment paradigms VM and serverless with respect to their energy consumption. Currently, no research has been conducted on the impact of the deployment paradigm on the energy consumption. In this chapter, we present an initial reflection on this topic. We present six principles relevant to the energy consumption of the deployment paradigms. Baldini et al. [37] state that if costs are the main driver for a paradigm choice, serverless architectures are recommended for bursty workloads. This is due to the cloud provider being responsible for the elasticity and the functions can scale to zero. If the workload is steady, a VM might be preferred as the resources can run without having to be constantly called and released. In our analysis, we expect a similar pattern to hold for energy efficiency.

Chapter 8

Discussion on Energy Efficiency in the Public Cloud

This chapter answers the research questions and reflects on the implications of the results. The first four sub-sections are structured according to the sub-questions and the last sub-section answers the main research question.

8.1 Tactics for Energy Efficiency

The first research phase considers sub-question **Q1**: *"What are cloud-native tactics to increase the resource efficiency of cloud consumers?"*. To answer this research question, we interviewed 17 practitioners at SBP to discover tactics that are potentially impactful to optimize cloud workloads for energy efficiency. We discovered 18 tactics and linked them to the literature. These tactics are more concrete than the tactics currently described in the related literature. This has as implication that the tactics are specific to certain use cases and it is therefore difficult to generalize the tactics for every workload. To illustrate, the tactic "Apply edge computing" is not relevant for workloads that are not network-intensive. Hence, the tactics cannot be perceived as guidelines but rather as a catalog containing potential tactics. To choose the relevant tactics from this 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 easily be applied to an existing infrastructure. Cloud providers could support their consumers in deciding which tactics are relevant and feasible.

8.2 Method to Estimate Energy Consumption in the Public Cloud

The second phase of the research is considered with finding a method to estimate the energy consumption in the public cloud. The relevant research question (**Q2**) is: *"To what extent can the energy consumption of individual workloads be monitored in the public cloud?"*. For cloud consumers, it is currently a difficult process to monitor their footprint in the public cloud. First of all, the major cloud providers do not provide direct feedback on the energy consumption of individual workloads to their cloud consumers. Hence, another method is required to estimate the energy consumption. The only tool that we encountered using Google search and Google Scholar was the Cloud Carbon Footprint (CCF) tool [6]. This tool estimates the energy and carbon footprint of individual cloud workloads based on the billing data as communicated by the cloud providers. The billing data is converted into estimated energy usage based on the predicated hardware usage and corresponding wattage. A limitation of the method is that the estimations do not account for energy consumed by cloud services that are not reported on the billing statement. Nevertheless, the CCF tool is currently the only available open-source tool to estimate the energy footprint. The CCF tool can be used by cloud consumers to understand their energy footprint over time. Unfortunately, for small-scale experiments, the tool lacks accuracy due to the experimental nature and required assumptions. Nevertheless, cloud consumers with a larger infrastructure can utilize the tool to perform experiments to understand which tactics have a positive effect on their energy consumption. We recommend cloud providers to invest more effort in disclosing the energy-related metrics to cloud consumers. Larger cloud consumers have more leverage in convincing cloud providers to disclose energy-related information.

8.3 Measure Impact of Selected Tactics on Energy Consumption

After we completed Phase 1 and 2 of the research we addressed the sub-question: *"What is the impact of the identified tactics on energy consumption?"*. Due to time restraints, we selected two tactics, namely **T11** and **T13**, for which we investigate the impact on the energy consumption. The results and implications of these case studies are respectively discussed in the following two sub-sections.

8.3.1 Apply edge computing

The first case study focuses on whether applying edge computing has a positive effect on the energy consumption. The case study involves a ML classification model for fall detection based on camera streams. We found that sending all the raw video data through the Internet to the public cloud has a negative effect on the energy consumption as opposed to processing the data on edge. When processing the data close to the source, the only data that needs to be sent over the network are the detected events. Sending solely messages describing events over the Internet significantly reduces the energy consumption of this particular use case. Accordingly, we advise cloud consumers to apply edge computing for IoT-related cases from an energy standpoint. Yet, it should be noted that this mainly applies to network- and data-intensive workloads.

8.3.2 Apply fitting computing paradigm

In the second case study, we analyzed the computing paradigms VMs and serverless to guide cloud users in choosing a suiting architecture for energy efficiency. We reason that VMs are more suitable for steady, predictable workloads, whereas serverless architectures are more energy-efficient for bursty workloads. It should be noted that we conducted a theoretical analysis and no empirical experiment, therefore, further experiments are required to support this claim. Moreover, changing the computing paradigm is a costly and time-consuming activity that can only be applied prior to cloud migration or whenever the current architecture is evidently inefficient.

8.4 Defining Energy Efficiency in the Public Cloud

Using these findings, we are reflecting on the last research questions. Namely, *"How can energy efficiency be defined in the public cloud?"*. Currently, no metric exists to assess the energy efficiency of individual workloads in the public cloud. Such a metric is required in order to monitor workloads or perform experiments to increase the energy efficiency. As explained in Section 2.2.1, energy efficiency can be defined as: *"the ratio between service output or results and the energy input required to provide it"* [20]. Translating this into a formula results in: $\frac{\text{service output}}{\text{energy consumption}}$.

Hence, in order to assess the energy efficiency of a workload, the energy consumption [kWh] needs to be metered. Due to the abstraction that the public cloud introduces between the cloud consumer and DC, cloud consumers experience little insight on their

energy consumption. Currently, the major cloud providers do not administer feedback on the energy consumption of individual workloads to cloud consumers. This is problematic as the energy consumption is a key metric to assess the energy efficiency. The CCF tool [6] provides an alternative approach to estimate the energy consumption of cloud services based on billing data of the cloud provider and open data on the energy consumption of ICT resources. This method is a valuable direction in understanding the energy consumption of individual workloads in the public cloud, however, these are no direct measurements and are based on several assumptions.

Afterward, the estimated (or preferably measured) energy consumption needs to be related to a quantifiable service output. Ultimately, the service output is related to the value software services deliver to end-users as this is the ultimate goal of running the computing resources. Unfortunately, end-user experiences are difficult to quantify and monitor in an automated fashion. To solve this, in computer science, the service output is often expressed in terms of resource usage. To illustrate, the service output can be quantified in terms of storage size, memory usage, compute power, or data transfer. However, when the resource usage is related to the energy consumption, the energy efficiency from the perspective of the cloud provider is assessed. To illustrate, the metric $\frac{\text{storage [GB]}}{\text{energy consumption [kWh]}}$ reflects the energy efficiency of the cloud DC storage infrastructure. This is independent of the architecture of the cloud consumer and can therefore not be optimized by the cloud consumer.

Consequently, in order to define the energy efficiency from the perspective of the cloud consumer, the service output should be defined at a higher level of abstraction. In other words, the quantified service output needs to represent the architecture and value of the workload. Quantifying the service output of a workload is case-specific. But we can provide some examples to express the level of abstraction. In the edge computing scenario, the service output can be quantified in terms of the amount of processed data at a certain time frame. For a web application, the service output can be defined in terms of the size of the hosted web app and the number of processed requests.

Defining the service output in terms of accomplished tasks also enables experiments to measure the impact of on-premise vs. cloud workloads on the energy efficiency. However, for these experiments to be accurate, the same method needs to be applied to measure the energy consumption on-premise vs. in the cloud. Due to the limited feedback of cloud providers with respect to the energy consumption, this is currently not possible.

As often in computer science, the no-free-lunch theorem applies as the metric to define the energy efficiency of a cloud workload is case-dependent. To conclude, measuring the

energy efficiency of a workload in the public cloud requires two metrics, namely, the service output and energy consumption. We state that cloud providers should provide feedback on the energy consumption of individual workloads to enable cloud consumers to assess and increase their energy efficiency. Moreover, we argue that an appropriate level of abstraction to define the service output lies in-between raw 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 include design decisions of the cloud consumer and not solely the efficiency of the underlying infrastructure. Furthermore, capturing these metrics can be automated.

8.5 Architect Energy-Efficient Software in the Public Cloud

Lastly, we can reflect on the main research question: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. First of all, it is important that awareness is created on the importance of an energy reduction strategy. On a company level, there need to be relevant strategies and requirements in place to achieve energy reduction. Next, in order to optimize a quality attribute, it should be measured so it can be monitored over time. To optimize energy efficiency, two metrics are required. First of all, the cloud consumers need to find a method to measure or estimate their energy footprint, and second, the cloud consumer needs to define an appropriate metric reflecting their service output. Once this is established, the cloud consumer should investigate opportunities to optimize their resource efficiency as this is proportional to energy efficiency. To accomplish this, the discovered tactics from our study can be considered to find relevant strategies. It is important to note that not all tactics are applicable for each workload and scenario. Accordingly, the appropriate architecture should be defined through experimentation on the energy impact of certain design decisions.

8. DISCUSSION ON ENERGY EFFICIENCY IN THE PUBLIC CLOUD

Chapter 9

Threats to Validity

Threats to validity are inevitable in research. This section reports on the potential threats to validity and measures taken to mitigate them. Validity is defined as the extent to which results are sound and applicable to the real world [29]. The used validity classification is proposed by Cook and Campbell [39] and further explained in the book by Wohlin et al. [29].

9.1 External Validity

External validity concerns the generalizability of the study [29]. We discovered the tactics at one company, this is a threat to the external validity as the tactics may not be representative of the overall sector. However, this threat is mitigated as SBP collaborates with many companies and organizations from different sectors (e.g., public, financial, transport) and therefore reflects the experiences of heterogeneous cloud consumers. Similarly, we assessed the impact of edge computing on energy efficiency through one case study. This threat is mitigated by generalizing the experiment set-up and expressing that the findings only hold for network- and data-intensive infrastructures.

9.2 Internal Validity

Internal validity considers the causality between treatment and outcome [29]. We selected one method to estimate the energy consumption in the cloud, namely, the CCF tool. This tool introduces some limitations as it is based on several assumptions. Furthermore, the calculations in the case studies are of theoretical nature as it was not possible to perform empirical experiments. This potentially harms the internal validity of the research. Also,

the proposed tactics merely focus on the quality attribute energy efficiency. This introduces a limitation as the tactics are recommended from one perspective. Furthermore, integrating the discovered tactics into an existing workload potentially introduces large investments that are not in proportion to the energy reduction. In other words, it could be infeasible to adopt certain tactics as the investments are relatively high in relation to the impact on energy efficiency. Hence, domain experts (e.g., cloud providers) are required to aid decision-making on whether implementing certain tactics is beneficial for specific workloads.

9.3 Construct Validity

Construct validity concerns the relation between theory and observation [29]. A threat to the construct validity arises as the research is exploratory and no other peer-reviewed studies could serve as an example. Hence, the methodology was experimental. We mitigated this threat by carefully designing our research using well-established principles from the book by Wohlin et al. [29]. Furthermore, we found inspiration from industry blog posts and gray literature. We also ensured that the studied concepts are well-defined and related to existing literature. Another potential threat is identified as we study cloud workloads from solely one quality attribute (i.e., energy efficiency). To understand the overall landscape, the trade-offs and impact on other quality attributes should be considered as well. Furthermore, the method introduced by the CCF tool in our research can be considered a *mono-method bias* as this is an experimental method where a different method could lead to different results. We mitigated this measure by ensuring that the method is introduced by a well-established organization and reviewed by domain experts.

9.4 Conclusion Validity

Conclusion validity concerns whether the relationship between the treatment and outcome is statistically correct and significant [29]. A potential threat to the conclusion validity arises as only one researcher conducted the interviews and identified the tactics. Furthermore, the interview transcripts are not made public due to the confidential nature of the conversations. We mitigated this threat by asking a domain expert to review and verify the tactics. Furthermore, the process of discovering the tactics is monitored by both academic and industrial supervisors. Another threat to the conclusion validity is that we did not collaborate with cloud providers and consumers. We tried to reach out to several cloud providers but unfortunately did not manage to schedule an appointment. We mitigated

the effect of this threat by ensuring that the interviewed practitioners frequently collaborate with cloud consumers. The interviewees were selected by the industry supervisor based on their cloud experience and availability. Hence, the subject selection was not completely randomized which could impose threats to the random heterogeneity of the subjects. To mitigate this threat, we ensured that their professional roles were diverse to reflect experiences from different areas of expertise.

Chapter 10

Conclusion

In this thesis, we address the research question: *"How can cloud consumers architect energy-efficient software in the public cloud?"*. We started by discovering tactics for resource efficiency from industry based on the assumption that optimizing the resource usage leads to energy efficiency. We discovered 18 tactics and reasoned on their impact on energy efficiency. However, proofing the impact on energy efficiency requires empirical experiments. Unfortunately, the major cloud providers do not disclose energy-related information about individual workloads to cloud consumers. Hence, we used the method from the CCF tool [6] to estimate the energy consumption of cloud instances. This is useful for cloud consumers to gain insights into their energy footprint, however, it lacks the accuracy to perform small-scale experiments. We did perform two case studies to assess the impact of two selected tactics on energy consumption. We found that applying edge computing has the potential to reduce energy consumption when dealing with a large amount of sensor data. Furthermore, we assessed the impact of deploying a VM versus a serverless computing paradigm on energy efficiency. We expect that serverless computing is more energy-efficient when dealing with bursty loads whereas VMs might be more energy-efficient for stable, compute-intensive tasks. We learned that it is difficult to define generic tactics for energy efficiency, as the impact on energy consumption is case-dependent. This further stresses the importance of experimentation. Last, based on our findings, we defined the concept of energy efficiency in the public cloud. We argue that the right level of abstraction to assess energy efficiency is at task level (e.g., amount of data processed or number of requests handled) to incorporate the architecture of the cloud consumer into the metric.

This is pioneering research as no previous peer-reviewed studies are available that approach the quality attribute energy efficiency from the perspective of cloud consumers in the public cloud. Hence, we invested effort in defining a method to approach energy-related

concerns in the public cloud. This is by no means a final answer to the research question but an important step to bring this topic to the attention of the research community and industry as in the current landscape, each sector should take responsibility for its carbon footprint. Future work is required to assess the impact of the discovered tactics on the energy efficiency for diverse workloads. For this, it is of great importance that cloud providers take responsibility and provide energy and sustainability-related feedback to their cloud consumers. In the meantime, our thesis shows that cloud consumers are currently able to reason and experiment to shape their architecture for energy efficiency.

Appendix A

Interview Respondents

Table A.1: Positions of interviewees.

ID	Position
1	Technology Officer
2	Mission Critical Engineer
3	Mission Critical Engineer
4	Cloud Architect
5	Cloud Acceleration Team Lead
6	Mission Critical Engineer
7	Cloud Vendor Manager
8	Cloud Architect
9	AWS Technical Practice Lead/Cloud Engineer
10	Chief Technology Officer
11	Mission Critical Engineer
12	Cloud Architect
13	Mission Critical Engineer
14	Cloud Architect
15	Mission Critical Engineer
16	Mission Critical Engineer
17	Chief Information Officer

A. INTERVIEW RESPONDENTS

Appendix B

Interview Questions

The interviews are semi-structured. The questions below illustrate the main topics discussed, however, the questions varied depending on the experience and role of the practitioner. Furthermore, during the conversations, more questions arose to understand the details of certain experiences and claims.

1. Could you introduce yourself and tell something about your position, experience, and career path?
2. Do you take energy efficiency into account when architecting/ designing/ developing in the cloud?
3. Could you give some examples of projects you worked on? Together with its challenging and design decisions?
4. Could you give some examples of strategies you applied to optimize cloud software for costs?

Appendix C

Model Validation Survey

1. What is your opinion on the overall model? E.g., is it clear?
2. Do you agree with the mentioned tactics for cost optimization (T1 - T16)?
3. Do you miss any tactics to optimize the costs/energy efficiency for software in the cloud?
4. Which tactics (T1 - T16) do you think have the largest impact on energy efficiency?

References

[1] Sarah Feldman. **The Cloud Market Keeps Moving Upwards**. *Statista*, 2019. vii, 2

[2] Luiz André Barroso and Urs Hölzle. **The datacenter as a computer: An introduction to the design of warehouse-scale machines**. *Synthesis lectures on computer architecture*, **4**(1):1–108, 2009. vii, 1, 2, 5, 6, 7, 42

[3] Felix Richter. **Cloud Infrastructure Market: Amazon Leads $130-Billion Cloud Market**. *Statista*, 2021. vii, 11, 12, 25, 43

[4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice (Third Edition)*. Addison-Wesley Professional, 2003. vii, 13, 14

[5] Carlos Paradis, Rick Kazman, and Damian Andrew Tamburri. **Architectural Tactics for Energy Efficiency: Review of the Literature and Research Roadmap**. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, pages 7197–7206, Hawaii, 2021. ScholarSpace. vii, 17, 18, 19, 26, 29, 31, 35

[6] ThoughtWorks. **Cloud Carbon Footprint**. `https://www.cloudcarbonfootprint.org`, 2021. Accessed on: 2021-06-26. vii, 3, 41, 43, 44, 46, 55, 62, 66, 68, 75

[7] ReportLinker. **Cloud Computing Market by Service, Deployment Model, Organization Size, Vertical And Region - Global Forecast to 2025**. 2020. 1

[8] Peter Mell and Tim Grance. **The NIST Definition of Cloud Computing**. *National Institute of Standards and Technology, U.S. Department of Commerce*, 2011. 1, 10

[9] Brian Hayes. **Cloud computing**. *ACM New York, NY, USA*, 2008. 1

[10] Accenture. **The green behind the cloud**. http://www.accenture.com/_acnmedia/PDF-135/Accenture-Strategy-Green-Behind-Cloud-POV.pdf. Accessed on: 2021-07-03. 1

[11] Luiz André Barroso and Urs Hölzle. **The Case for Energy-Proportional Computing**. *IEEE, Computer*, **40**(12):33–37, 2007. 1, 59

[12] Lotfi Belkhir and Ahmed Elmeligi. **Assessing ICT global emissions footprint: Trends to 2040 & recommendations**. *Journal of Cleaner Production*, **177**:448 – 463, 2018. 2

[13] International Energy Agency (IEA). **Aviation**. http://www.iea.org/reports/aviation, 2020. Accessed on: 2021-03-26. 2

[14] Emily Sommer, Mike Adler, John Perkins, Joshua Thiel, Hilary Young, Chelsea Mozen, Dany Daya, and Katie 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. 3, 41, 42, 55

[15] Yuzhong Sun, Yiqiang Zhao, Ying Song, Yajun Yang, Haifeng Fang, Hongyong Zang, Yaqiong Li, and Yunwei Gao. **Green challenges to system software in data centers**. *Frontiers of Computer Science in China*, **5**(3):353–368, 2011. 4

[16] CISCO. **What Is a Data Center**. https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html. Accessed on: 2021-03-09. 5

[17] Nathaniel Horner and Inês Azevedo. **Power usage effectiveness in data centers: overloaded and underachieving**. *The Electricity Journal*, **29**(4):61–69, 2016. 8

[18] Dinesh Reddy, Brian Setz, Subrahmanya Rao, G. Gangadharan, and Marco Aiello. **Metrics for Sustainable Data Centers**. *IEEE Transactions on Sustainable Computing*, **2**(3):290–303, 2017. 8

[19] Georgia Lykou, Despina Mentzelioti, and Dimitris Gritzalis. **A new methodology toward effectively assessing data center sustainability**. *Computers & Security*, **76**:327 – 340, 2018. 8

[20] Luis Pérez-Lombard, José Ortiz, and David Velázquez. **Revisiting energy efficiency fundamentals**. *Springer, Energy Efficiency*, **6**(2):239–254, 2013. 8, 9, 67

[21] Thomas Wiedmann and Jan Minx. **A definition of 'carbon footprint'**. *Nova Science Publishers, Ecological economics research trends*, **1**:1–11, 2008. 9

[22] 451 Research, commissioned by AWS. **The Carbon Reduction Opportunity of Moving to Amazon Web Services**. Technical report, 2019. 12

[23] Microsoft. **The carbon benefits of cloud computing: a study on the Microsoft Cloud in partnership with WSP**. Technical report, 2020. 12

[24] Karim Djemame, Django Armstrong, Richard Kavanagh, Ana Juan Ferrer, David Garcia Perez, David Antona, Jean-Christophe Deprez, Christophe Ponsard, David Ortiz, Mario Macias, et al. **Energy efficiency embedded service lifecycle: Towards an energy efficient cloud computing architecture**. In *CEUR Workshop Proceedings*, **1203**, pages 1–6. CEUR Workshop Proceedings, 2014. 15

[25] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. **Energy-Efficient Cloud Computing**. *The Computer Journal*, **53**(7):1045–1051, 2010. 15

[26] Rick Kazman, Serge Haziyev, Andriy Yakuba, and Damian Tamburri. **Managing Energy Consumption as an Architectural Quality Attribute**. *IEEE, Software*, **35**(5):102–107, Sep. 2018. 15

[27] Erik Jagroep, Jan Martijn van der Werf, Sjaak Brinkkemper, Leen Blom, and Rob van Vliet. **Extending software architecture views with an energy consumption perspective**. *Springer, Computing*, **99**(6):553–573, 2017. 16, 25

[28] Giuseppe Procaccianti, Patricia Lago, and Grace Lewis. **Green Architectural Tactics for the Cloud**. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 41–44, 2014. 16, 17

[29] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012. 21, 71, 72

[30] Amazon Web Services. **Well-Architected Framework**. Technical report, July 2020. 27, 28, 29

[31] Arman Shehabi, Sarah Josephine Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. **United States Data Center Energy Usage Report**. Technical report, Berkeley Lab, 2016. 42, 43

[32] Standard Performance Evaluation Corporation (SPEC). **Power Usage Report**. `http://www.spec.org/power_ssj2008`, 2008. Accessed on: 2021-06-26. 42

[33] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. **Edge Computing: Vision and Challenges**. *IEEE, Internet of Things Journal*, **3**(5):637–646, 2016. 49

[34] Joshua Aslan, Kieren Mayers, Jonathan Koomey, and Chris France. **Electricity intensity of Internet data transmission: Untangling the estimates**. *Wiley Online Library, Journal of Industrial Ecology*, **22**(4):785–798, 2018. 54

[35] Vlad Coroama and Lorenz Hilty. **Assessing Internet energy intensity: A review of methods and results**. *Elsevier, Environmental impact assessment review*, **45**:63–68, 2014. 54

[36] Robert P. Goldberg. **Survey of virtual machine research**. *Computer*, **7**(6):34–45, 1974. 59

[37] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, and Philippe Suter. *Serverless Computing: Current Trends and Open Problems*, pages 1–20. Springer, Singapore, 2017. 59, 60, 62, 64

[38] Amazon Web Services (AWS). **AWS Lambda Developer Guide**. `https://docs.aws.amazon.com/lambda/latest/dg`, 2021. Accessed on: 2021-08-11. 61, 62

[39] Thomas D Cook, Donald Thomas Campbell, and Arles Day. *Quasi-experimentation: Design & analysis issues for field settings*, **351**. Houghton Mifflin Boston, 1979. 71