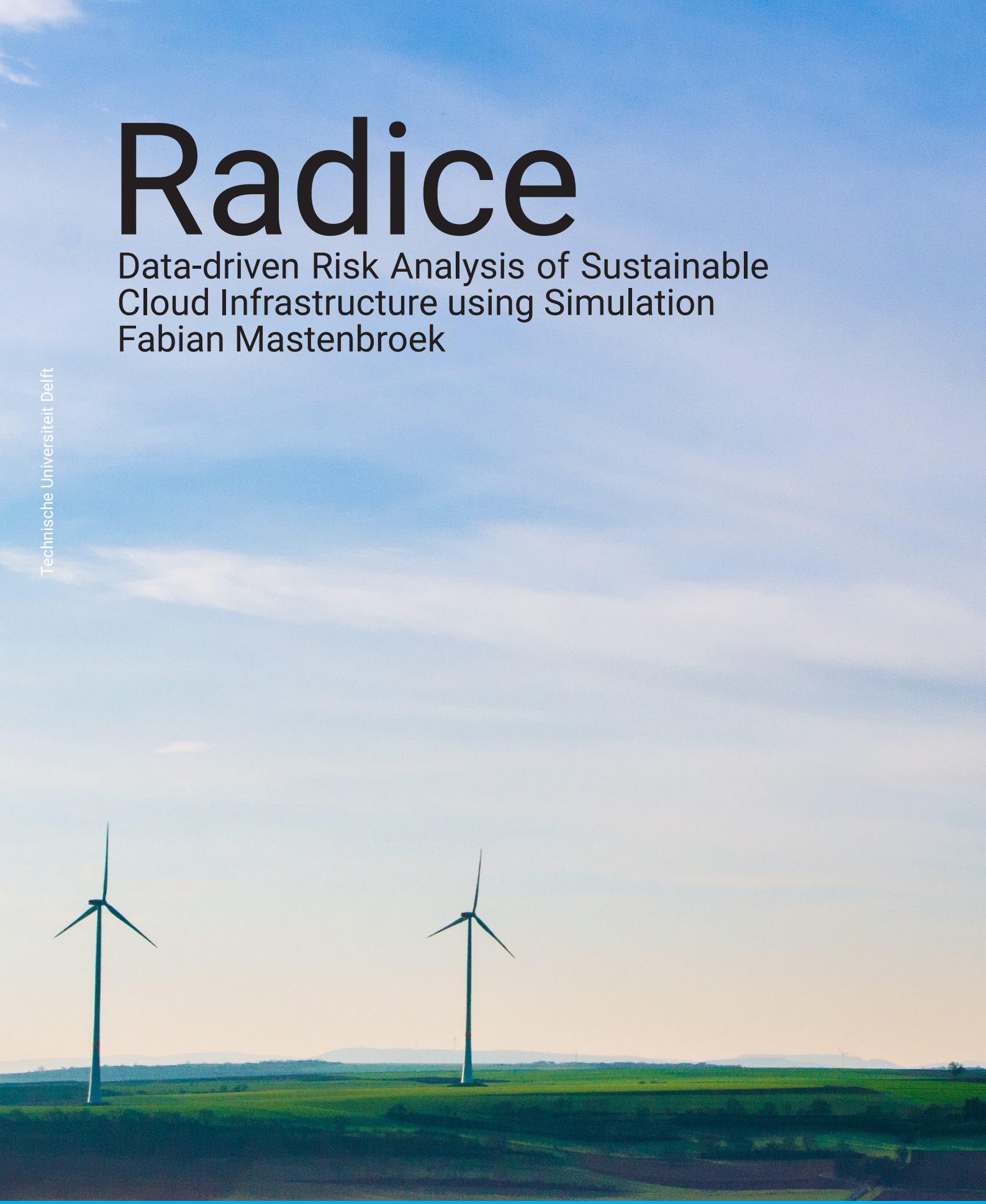
Radice

Data-driven Risk Analysis of Sustainable Cloud Infrastructure using Simulation

Fabian Mastenbroek

Technische Universiteit Delft



TU Delft

Radice

Data-driven Risk Analysis of Sustainable Cloud Infrastructure using Simulation

by

Fabian Mastenbroek

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, March 16, 2022, at 1:00 PM.

Student number:	44552199	
Project duration:	May 17, 2020 – March 16, 2022	
Thesis committee:	Prof. dr. ir. D.H.J. Epema,	TU Delft, chair
	Prof. dr. ir. F. A. Kuipers,	TU Delft, committee member
	Prof. dr. ir. A. Iosup,	VU Amsterdam and TU Delft, daily supervisor
	Ir. V. van Beek,	TU Delft and Solvinity, external expert
	Representative,	TU Delft, Board of Examiners

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Cloud datacenters underpin our increasingly digital society, serving stakeholders across industry, government, and academia. These stakeholders have come to expect reliable operation and high quality of service, yet demand low cost, high scalability, and corporate (environmental) responsibility. Datacenter operators are confronted frequently with highly complex decisions that involve numerous aspects of risk. The consequence of bad decisions can be financial penalties or even loss of customers on the one hand, or a competitive disadvantage or unsustainable environmental impact on the other hand. Despite risk analysis being an integral part of the design and operation of cloud infrastructure, relatively few comprehensive approaches and tools exist, leaving many datacenter operators ill-equipped to make informed decisions with confidence.

We propose Radice, an instrument for data-driven analysis of IT-related operational risks in sustainable cloud datacenters. Unlike most state-of-the-art approaches used by the industry, Radice automates the process of risk analysis in datacenters and utilizes the large and diverse volume of data reported by the monitoring systems in datacenters, including environmental data. Underpinning this system is the trace-based, discrete-event simulator OpenDC, which enables the exploration of many risk scenarios through its support for diverse workloads, datacenter topologies, and operational phenomena. Radice's interactive and explorative user interface assists datacenter operators in addressing complex decisions involving risks, providing them with actionable insights, automated visualizations, and suggestions to reduce risk.

We implement Radice and conduct a comprehensive evaluation of the system to demonstrate how it can aid datacenter operators when confronted with fundamental risk trade-offs. Although Radice is designed to work across many kinds of datacenters, in this work, we focus on private-cloud, business-critical workloads, and on public-cloud operations, representing the majority of workloads in Dutch datacenters. Our experiments show many interesting findings, supporting our claim for a need for data-driven risk analysis in datacenters. We highlight the increasing risk faced by datacenter operators due to price surges in the electricity and CO_2 bond markets, and demonstrate how Radice can be used to control such risks. We further show that Radice can automatically optimize topology and operational settings in datacenters for risk, revealing configurations that reduce the overall risk by 10%–30%. Following extensive performance engineering, Radice is able to evaluate risk scenarios by a factor 70x–330x faster than others, opening possibilities for interactive risk exploration. We release Radice as free and open-source software for the community to inspect and re-use.

Preface

This thesis is the result of multiple years of education and research at Delft University of Technology, and completes the challenging, but rewarding journey towards my master's degree. Along the way, I had the unique opportunity to bootstrap a platform for datacenter research, (co-)author several scientific articles, present at national and international conferences, and educate other students about clouds and datacenters. This all would not have been possible without the support from many kind and helpful people, towards whom I would like to express my sincere gratitude.

I am especially grateful to my two supervisors, Alexandru Iosup and Vincent van Beek, who guided me throughout this journey and helped me shape this project into its current form. You were an endless source of inspiration, motivation, and feedback; I remember coming out of each meeting inspired and with many new ideas to try. Alexandru, thank you for introducing me to the world of Computer Science research, inspiring me to go beyond what is necessary, and pushing me to be the best version of myself. I am incredibly grateful for the many opportunities you provided me with, even as I was at the beginning of my academic career. Vincent, your guidance and profound expertise have been immeasurable to me. You taught me a great deal about the technical details of actually running a datacenter at scale, as well as the science behind it.

I am also grateful to the people at Solvinity for providing me with the opportunity to conduct this research project at their company. Solvinity's efforts to engage in such kind of open-ended, open-access research collaborations is unique for companies in this sector and of this scale. I am very thankful for this opportunity.

I would like to express my gratitude towards all people at the AtLarge Research group. Thank you for the many discussions and collaborative sessions together, which were always a source of joy and inspiration to me. A big thank you as well to all the people that have worked on the OpenDC project. Together, you have shaped the project into the established research platform for datacenters that it is today. Thanks especially to Hongyu for your enthusiasm and inspiring work ethic during our collaboration. I also want to extend a special thank you to Georgios Andreadis, my co-author and former OpenDC colleague, with whom I had the pleasure to collaborate extensively throughout my journey. I look back with great joy to the projects we have worked on together, to the moments of brainstorming, implementation, and dissecting results. Georgios, your passion, skill, and knowledge have been a true source of inspiration to me. Your tireless efforts were instrumental in getting OpenDC where it is today, and without it, this project would not have been possible.

Finally, I would like to thank my friends and family for supporting and encouraging me throughout this journey. You helped me take my mind off the thesis when I needed to relax, never failing to put a smile on my face. Thank you, Maaike, for all your love and support, for your willingness to listen, and for always being there while I was busy and stressed. I could not have done this without you.

Fabian Sebastian Mastenbroek
Delft, March 2022

Contents

1 Introduction

Our society is becoming increasingly digital. Each day, over 2.5 quintillion bytes of data is produced, and this number is estimated to grow to 463 exabytes per day by 2025 [126]. The emergence of paradigms such as the Cloud and Internet of Things (IoT) coupled with improved connectivity (with the deployment of 5G) is accelerating the digitalization of our economy and our society at an unprecedented rate [60, 63, 64].

To reliably process, store, and serve data at this scale, operators of digital services across industry, government, and academia employ *datacenters*. Datacenters are physical facilities composed of large numbers of interconnected computers and storage, accompanied by reliable power distribution, cooling equipment, and high-speed internet access [19]. In the past, it was common for organizations to procure, service, and operate IT infrastructure necessary for their digital services *in-house*. Organizations would usually build their own datacenters (*on-premise*), or rent space in existing facilities (*co-located*), such as those run by Equinix, Interxion, and Digital Realty. In contrast, currently most organizations deploy their digital services in the cloud [60, 133], where the responsibility of the hardware or even the full operational lifecycle of the service (e.g., through *serverless computing* [54]) is taken over by the cloud operator. These operators, which include Amazon and Google, often employ massive "hyperscale" datacenters to benefit from economies of scale [19]. Rising server equipment purchases, which have grown 3% each year, are almost entirely attributable to these "hyperscale" datacenters, with estimates suggesting that 40% of servers already run in such datacenters [133].

The impact of datacenters on the Digital Economy is substantial. Datacenters are a critical component of almost any organization today [161]. In 2019, digital services hosted in datacenters directly powered 33% of the Dutch economy, or €242 billion in terms of GDP [122], and enabled over 2.1 million jobs nationwide [68]. Furthermore, Google contributes almost €500 million to the European economy annually from their datacenters alone [20]. This work aims to provide an instrument to explore key challenges in today's datacenters, related to (risks in) ensuring their long-term, sustainable operation.

As information technology becomes more entrenched in our society, *How to ensure reliable and efficient operation of datacenters at unprecedented scale?*. Despite technological advancements and better management of availability, failures in datacenters continue to be a major source of concern for many operators, and increasingly, for customers and regulators [100]. The financial consequences of outages can be serious, and the situation is deteriorating. In 2021, four in ten outages cost between $100,000 and $1 million, and about one in six cost over $1 million [100]. Facebook's recent outage reportedly cost the company over $60 million, with similar numbers being reported for outages at Amazon and Google [13, 145].

Furthermore, as our society is confronted with climate change, *How to overcome challenges of sustainability in datacenters?* Currently, datacenters are responsible for 1%–2% of the global electricity demand [75, 86, 106], with some studies suggesting that it could quadruple by 2030 [10], whereas others show the growth stagnating [106]. The electricity demand of Dutch datacenters represents already 3% of all electricity supplied by the national power grid [137] and is expected to grow to 12% by 2030 [156]. Datacenter operators have responded to this problem by improving energy efficiency and adopting clean electricity. As a result, the IT sector is now the world's largest purchaser of renewable energy [88, 115, 146]. In the Netherlands, 88% of electricity purchased by datacenters comes from sustainable sources [68].

However, even with the increasing use of renewable energy in the industry, emissions and other environmental impacts need to be accounted for. Datacenters in both the Netherlands and US consume 0.1% of all drinking water [68, 114, 133]. Yet, only half of all datacenters monitor their water consumption [29]. Little

also is known about the greenhouse gas emissions of datacenters [113]. These emissions lead to deteriorated air quality (linked to adverse health effects and millions of deaths worldwide) [40, 119] and increased concentrations of carbon dioxide in the atmosphere (contributing to climate change).

Datacenter operators must act fast. Recent plans of Facebook and Microsoft to build new hyperscale datacenters in the Netherlands have sparked a debate among the general public about the desirability of such projects [128]. A recent survey found that datacenter operators are increasingly concerned about negative sentiments towards datacenters [68]. Under pressure from the media and public, lawmakers and regulators might be looking to introduce new laws and regulations for datacenters to meet sustainability targets [29]. For example, the Climate Neutral Data Centre Pact[1] stipulates that datacenters in Europe reduce their power usage effectiveness (PUE) [1] to 1.4 and use 100% renewable energy by 2030. However, datacenters themselves are also threatened by climate change. Nearly half of the datacenters report being struck by extreme weather events in 2021 [98, 99]. An even more pressing issue is the lack of available grid capacity for new datacenters. In some regions of the Netherlands, there is currently a construction stop for datacenters due to limits of the power grid [68], a growing problem as the energy transition moves forward.

1.1. Risks in Datacenters

Risk management is an integral aspect of the design and operation of cloud datacenters. Datacenter architects and operators are confronted with major research and engineering challenges, and over the lifetime of a datacenter, they face fundamental risk trade-offs. Since the consequences of bad decisions can be serious financial penalties or even loss of customers, as a natural mechanism for risk management, datacenter operators tend to be very conservative in adopting new methods and technologies [67, 91]. Their position is not surprising, given that the majority of outages in datacenters result from software, IT (mis)configuration, or network issues. Often, human error is cited as the root cause of these issues [100]. Even very successful companies in this field, such as Google or Facebook, have to rely on highly-trained site reliability engineering (SRE) teams to solve the numerous problems that arise on a daily basis in their hyperscale datacenters [27, 66]. An instrument to support the risk management process could enable more effective and informed decision-making, by assisting datacenter architects and operators with exploring the impact of many different risk scenarios. Below, we present four use-cases in cloud datacenters that could benefit from such an instrument.

Long-term capacity planning (*procurement*) of cloud infrastructure is a critical yet non-trivial optimization problem faced by datacenter architects that deals with various risk trade-offs. The primary goal of capacity planning is to ensure adequate infrastructure exists for incoming workloads, by acquiring and installing sufficient resources ahead of time, to be able to support all incoming workloads. This necessitates accurately predicting the required capacity for years in advance and shaping the infrastructure topology to meet demand while minimizing costs and environmental impact. Simply over-provisioning capacity is expensive, considering that servers account for 45% of the total costs of a datacenter [67] and each CPU or GPU can cost upwards of hundreds of euros [148], not to mention a waste of resources and a waste of electricity. Conversely, under-provisioning capacity could lead to risks of not meeting SLAs [8, 27], inability to absorb catastrophic failures [19, p.37], or even unwillingness to accept new users. Despite the many approaches that have been proposed to address this problem [12, 35, 166], companies still rely on simplistic, rule-of-thumb reasoning for decisions. To minimize operational risks, many such industry approaches lead to significant over-provisioning [65, 67]. This has significant financial and environmental implications, with research showing that right-sizing cloud infrastructure could lead up to a 90% reduction in electricity expenses [24].

Risk trade-offs also appear when operating a datacenter. Critical for datacenter operation is the *scheduler*, which provisions resources for a user or application, and decides which parts of the application (typically *tasks* or virtual machines) to map to provisioned resources. Since optimal scheduling is NP-hard, schedulers typically employ various heuristics and other online methods that merely approximate or satisfice the solution. Current industry approaches for scheduling are brittle [57, 91, 140] and consequently industry-wide server utilization is only 15%–25% [89, 134, 151]. Despite the community developing many new scheduling techniques every year, only few techniques are adopted by datacenters, since conservative datacenter operators are highly unlikely to deploy new techniques without an established track record [91]. On the other hand, co-locating too many workloads on machines might lead to performance interference [93, 147, 150]. The discovery of the Meltdown and Spectre vulnerabilities [92, 102] has further complicated this problem, by introducing the risk of information leaks to neighboring workloads.

Furthermore, efforts to improve energy efficiency of datacenters have mostly focused on the mechanical

[1] `https://www.climateneutraldatacentre.net`

and electrical infrastructure (e.g., cooling, power distribution). The average PUE [1], a ratio of the total facility energy to IT energy, in datacenters has dropped significantly since 2007, but over the past few years, the decline has stagnated [29]. Most of the low-hanging fruit regarding facility efficiency have been addressed already, and further gains likely need larger investments. Meanwhile, major opportunities to reduce energy usage of IT infrastructure in datacenters remain unexploited. Yet, these opportunities are among the least-implemented measures in the *EU Code of Conduct for Data Centres* [23]. Only hyperscale operators and a few leading-edge enterprises do so rigorously [23]. For instance, the majority of datacenter operators in the Netherlands do not utilize power saving features in servers [72]. Despite research showing that power saving functionality in servers offers the same (or even better) performance with reduced power consumption [72], datacenter operators are often unwilling to enable these features, to avoid any chance of performance degradation. Often, there is a lack of awareness of a clear business case, or there are split incentives, where the IT team is not concerned with energy consumption (and the resulting expenses), but they are held responsible for performance issues [23]. Notwithstanding environmental concerns, with 15 percent of the total costs of datacenters going to electricity [67], consuming too much of it leaves datacenters vulnerable to price spikes; a serious risk given the nearly tenfold price increase that occurred in 2021.

Finally, the lifecycle of datacenter infrastructure presents additional risk trade-offs. Through hardware refreshes, datacenter operators can steadily increase compute capacity, while reducing energy consumption, as a result of the increased energy efficiency of servers over the past several decades. However, replacing equipment too early may cost more in hardware than is saved on energy efficiency [23]. Lifecycle analysis and timely hardware refreshes potentially offer more energy savings than further decreasing PUE [8, 22]. In a survey of European datacenters, IT equipment older than 5 years was found to consume 66% of electricity, yet accounted for only 7% of the capacity [23].

If these challenges were to be fully addressed, it could lead to significant reductions in costs, energy usage, and carbon footprint of datacenters [23]. Clearly, datacenter operators are unable to grasp the impact of changes, lacking insight into the risk profile of datacenters. Customers of datacenters, and increasingly regulators and auditors, are also demanding more transparency from datacenter operators about the infrastructure and its risks [29], given today's reliance on IT infrastructure and the severe impact of outages. Although there exist already various qualitative approaches in the industry for analyzing the risks that appear in cloud datacenters, such as the risk assessment offered by Uptime Institute [97], these processes are mostly manual. There are also a few qualitative approaches, such as the AssessGrid [50] project, but these models are often restrictive and do not cover the diverse risk trade-offs that datacenter operators are confronted with today. In many cases, datacenter operators do not even have a formal process when addressing these risk trade-offs. Faced with large volumes of data produced by the monitoring systems, datacenter operators need to make complex risk trade-offs, while often lacking dedicated tooling. An instrument that formalizes this process and assists datacenter operators in investigating the many possible different risk scenarios could help make more fine-grained and deliberate considerations, and reduce overall risk.

1.2. Problem Statement

We identify and aim to address in this work four key problems that emerge when addressing the problem of assessing IT-related risks in sustainable cloud infrastructure.

First, we observe the **lack of a common model for expressing IT-related risks of sustainable cloud infrastructure**. Although the community has developed several models for expressing risk in datacenters [50, 52, 57, 59, 74, 97, 150], most models focus solely on availability or quality-of-service as an indicator for risk, and do not take into account other business goals. For instance, sustainability or societal impact is considered rarely in these models. From the few models that do, none also cover the diverse phenomena and/or substantial innovation in cloud datacenters. Qualitative models that exist for risk analysis in datacenters require subjective interpretation of the data, which make their results difficult to reproduce and compare. The need for holistic insight into the risk profile of datacenters, which includes not only customer-facing risks, but also other aspects such as sustainability, necessitates a flexible model that incorporates different dimensions of operational data available in datacenters, including the diverse metrics collected by monitoring services.

Second, we observe the **need for an instrument for facilitating risk analysis of datacenters**. The formal processes for risk analysis in datacenters are often manual, which makes it difficult to repeat the process frequently and identify actual risks that may arise during daily operation. The few tools that are available [50, 59] incorporate only a subset of risk factors, and often rely on high-level models, which struggle to capture the heterogeneity and complex interplay of the hardware and software ecosystems present in datacenters. Lack-

ing comprehensive tools and techniques for risk analysis, datacenter operators are forced to rely on rules-of-thumb based on casual visual interpretation of the complex data provided by datacenter monitoring, when assessing the potential risk in the datacenter. Consequently, this practice has resulted in over-provisioning of datacenter infrastructure, low resource utilization, and unnecessary high environmental impact.

Third, we observe a **lack of comprehensive evaluations of risk analysis approaches for datacenters**. The existing approaches for risk analysis are seldom tested with real-world scenarios, and their results are only rarely peer-reviewed [8, 162]. To evaluate risk analysis approaches thoroughly, we advocate comprehensive experiments using real-world operational traces and diverse risk scenarios. This allows us to replicate the real-world circumstances faced by datacenter operators and put the system we propose to the test.

Finally, we observe the **need for accessible and comprehensive risk analysis tooling for datacenters**. The existing tools are rarely publicly available, and even fewer are open-source. Of the available tools, none model cloud datacenters to the level of detail present in Section 2.2, failing to capture emerging technologies and applications, such as serverless computing [54] and machine learning workloads [4]. A tool to evaluate potential future scenarios and analyze the risk profile, can assist datacenter operators in making better informed decisions, especially in the face of large volumes of data available in datacenters.

1.3. Research Questions

We divide the problem of risk analysis of sustainable cloud infrastructure into three research questions:

RQ1 How to model IT-related risks in sustainable cloud infrastructure?
It is necessary that we establish a common definition of *IT-related risks*. We must develop a holistic model that identifies factors that contribute to risk in the datacenter, as well as understanding how the impact of these risks can be quantified. This is very challenging, because the common model should address many kinds of operational and technical details, for both datacenter and energy infrastructure, and link these to economics and (environmental) sustainability.

RQ2 How to design a system for simulation-based analysis of IT-related risks in cloud infrastructure?
There is currently a lack of research on the design of systems for simulation-based risk analysis, focused specifically on sustainable cloud infrastructure. In this work, we aim to understand the design implications and opportunities that this model brings to such systems, by exploring the design space of such systems. Through (discrete-event) simulation, we can provide precise and accurate analysis of complex situations, and thus provide timely information about the risks the Dutch ICT industry faces.

RQ3 How to evaluate and validate a system for risk analysis of cloud infrastructure?
To understand the operation of the proposed system and whether it satisfies its design goals, we need to quantify its performance. Conducting a comprehensive and sound evaluation methodology is difficult and presents challenges of design and reproducibility [120, 147].

1.4. Research Methodology

In this work, we approach the problem statement and the subsequent research questions with a distributed systems approach, a combination of conceptual, technical, and experimental work, guided by the state-of-the-art AtLarge Design Process [81].

Towards addressing RQ1 and RQ2, we survey in Chapter 2 the state of the art in risk analysis. We consider literature of closely-related fields, as well as separate sciences, such as finance. This will help identify the current methods and models that exists for quantifying risk. We analyze how these methods are applicable in the context of datacenters or how we can adapt them to incorporate sustainability.

To address RQ2, we propose Radice, a system for risk analysis of cloud datacenters. We first conduct a requirement analysis, identifying the stakeholders of such a system and describing relevant use-cases. From there, we formalize the design of Radice according to the AtLarge Design Process [81]. We put emphasis on co-evolving the solution and the problem (i.e., by adding requirements as we understand the problem better, and iterating on the solution to include these new requirements). Our approach differs from other approaches in the field in that we employ *discrete-event simulation* [90], compared to high-level mathematical models frequently used in related work. The use of discrete-event simulation enables complex and long-term analysis, and thus enables answering key questions related to risks and sustainability. However, the use of discrete-event simulation also introduces additional challenges of performance, of validity, of interoperability, and

of human-computer interaction. Radice builds upon OpenDC, an open-source datacenter simulation platform [107], extending it with capabilities for risk analysis. By building on top of OpenDC, instead of creating a stand-alone tool, we ensure that the users of Radice automatically benefit from all future improvements to the OpenDC platform without requiring additional integration.

To address RQ3, we implement a working prototype of Radice and conduct an extensive evaluation of it through experimental analysis, performance analysis, and validation. First, we implement a prototype of Radice, extending OpenDC with many functionalities that benefit both the prototype and the simulation platform's broader user base in the process. Second, we conduct an experimental analysis of various risk scenarios using real-world workload traces and considering complex operational phenomena. We consider a selection of common use-cases for Radice and address them with the designed instrument. Third, we perform a performance analysis [82, 120], where we assess different dimensions of performance (e.g., runtime and memory usage) of the prototype and compare it against a reference cloud simulator. Finally, we validate Radice. This includes outlining the process of ensuring validity by using manual inspection and proactive discussion with experts, conducting an empirical comparison of the prototype against a reference cloud simulator, and validating Radice using mathematical analysis of queuing processes [71, 144].

Overall, our approach is focused on *open and reproducible science* [26, 120]. The development of Radice follows modern engineering practices, from using continuous integration, to extensive testing and documentation. We adhere to open standards and formats that are commonly used in the field where possible [78, 154]. Our experiments are fully reproducible and self-contained [147]. The prototype of the system as well as the extensions to the OpenDC simulator are released as free and open-source software (FOSS) on GitHub[2].

1.5. Thesis Contributions

This thesis has resulted in the following contributions:

1. **(Conceptual)** Dissemination of system design, findings, and experiences:

   (a) **Design of Radice**: *the first system for data-driven risk analysis of sustainable cloud infrastructure using simulation* (Chapter 3). The contribution here includes several deep conceptual advances: (1) a new process for computing the risk fine-grained, high-precision models of ICT operations and their energy consumption, (2) new capabilities that change fundamentally the risk abstraction typically related to traditional and cloud performance metrics to higher-level metrics and decisions, (3) a groundbreaking and flexible approach to quantify and then optimize risk in cloud datacenters, (4) a new genetic algorithm to optimize risk, and (5) specific technical solutions that address a specific focus on reproducibility. We see these points, individually but especially together, as a strong contribution to the field.

   (b) **Analysis and Evaluation of Radice**: using experiments and queuing theory, we have evaluated and analyzed the capabilities and performance of Radice. The capabilities of Radice are unique in the published domain, so there Radice does not have alternatives to compare against; we use instead queuing theory for validation. Regarding performance, we analyze the runtime and memory requirements of Radice, and contrast for these important performance indicators Radice with a state-of-the-art simulator much used in today's practice.

   (c) **Analysis using Radice**: analysis of risk in contemporary datacenters, with extensive setup diversity and many important findings. This analysis is unique in the public discourse and adds important insights with societal impact.

   (d) Article on OpenDC 2.0 accepted to a top-tier conference in the field, as first author: *OpenDC 2.0: Convenient Modeling and Simulation of Emerging Technologies in Cloud Datacenters* [107], CCGrid 2021, presented in May 2021 (acceptance ratio 26%).

   (e) Article on Capelin accepted to a tier-1 journal in the field, as second author: *Capelin: Data-Driven Compute Capacity Procurement for Cloud Datacenters using Portfolios of Scenarios* [12], TPDS, published in January 2022.

   (f) Article on our experiences using OpenDC for education, in development.

   (g) Article on the Radice core, aiming for the tier-1 conference SC, to be submitted in April 2022.

[2] `https://github.com/atlarge-research/opendc`

2. **(Technical)** Development and publication of open science artifacts:
   (a) Experiment setup, validation, and evaluation of the Radice system for risk analysis of sustainable cloud datacenters (§4.4-§4.6), published on the Zenodo Open Science platform[3].
   (b) Extensions to the FOSS datacenter simulation platform OpenDC [107] and a prototype of Radice (§4.2), integrated into the main repository for inspection and reuse: `https://github.com/atlarge-research/opendc`
   (c) Significant software engineering of Radice, OpenDC, and their interconnects. The engineering uses a state-of-the-art software engineering process that improves the credibility of OpenDC in comparison with all other simulators available in the community.
   (d) Performance engineering that enables large-scale experiments. The result is that OpenDC can simulate three months of datacenter operation in a matter of seconds, which means this work enables online risk-evaluation scenarios with OpenDC and also that OpenDC is 70x-330x faster (!) than one of the most used cloud simulators in the field.
3. **(Outreach)** Interaction with various stakeholders on our research on datacenter infrastructure:
   (a) Presented in national conferences for industry and academia, such as ICT.OPEN and CompSys NL.
   (b) Delivered key learning experiences in classroom-based (traditional) courses, as part of the Netherlands-wide doctoral course on Cloud and Big Data, Vrije Universiteit M.Sc. course on Distributed Systems, and TU Delft B.Sc. Honours Programme courses.
4. **(Leadership)** Coordination of the OpenDC datacenter simulation project[4]:
   (a) Oversaw the development of OpenDC, leading a team of 10+ members and resulting in a tested, trusted, and flexible simulator for the datacenter community.
   (b) Supervised the B.Sc. thesis projects of two *cum laude* students from the Vrije Universiteit Amsterdam, with one student receiving the *2021 ADS Thesis Award*[5].
   (c) Guided a team of B.Sc. students at the TU Delft doing a Software Project on OpenDC.
   (d) Integrated external contributions to the project from students of Eindhoven University of Technology and Vrije Universiteit Amsterdam.

1.6. Thesis Structure

The remainder of the thesis is structured as depicted in Figure 1.1. In Chapter 2, we describe relevant background information. We present in Chapter 3 the design of Radice. In Chapter 4, we evaluate a prototype of this system. Finally, in Chapter 5, we summarize the contributions of this thesis and propose future work that could emerge from this project.

[3] Link made available on: `https://opendc.org`
[4] `https://opendc.org`
[5] `https://amsterdamdatascience.nl/news/winners-of-the-ads-thesis-awards-announced/`

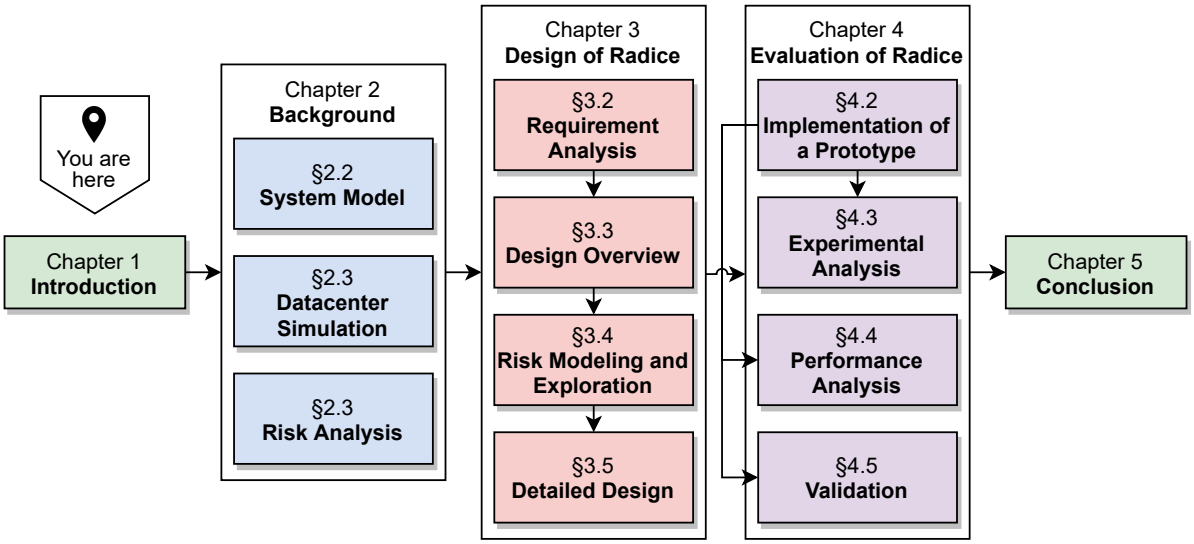


Figure 1.1: Structure of this thesis.

2

Background

We present in this chapter a comprehensive overview of subjects related to risk analysis for sustainable cloud infrastructure, serving as a foundation for the content in the remainder of this thesis.

2.1. Overview

The contribution of this chapter is three-fold:

1. We construct a system model for datacenter operation (Section 2.3). This model provides a top-down perspective of the structure and operation of datacenters, serving as a useful context for the contributions in the remainder of this thesis.

2. We provide a primer on datacenter simulation (Section 2.2).

3. We survey the state-of-the-art concerning risk analysis (Section 2.4).

2.2. System Model for Datacenter Operations

Datacenters are complex, large-scale computer ecosystems that are necessary to sustain the global demand for data access and processing. To properly understand the process of risk analysis for datacenters, it is vital that we comprehend the context in which the risk analysis is conducted. In the field, *system models* are a common means to encapsulate the context in which the work placed. System models have also been proposed in other lines of research, such as in the area of cloud resource scheduling [11, 152] or capacity planning [12]. Each of these models represent the aspects of the environment relevant to the work at an appropriate level of abstraction, and help describe the context of research contributions. Below, we summarize the current state-of-the-art that exists across these aspects and present our model for datacenter operations, depicted in Figure 2.1, and based on [11].

2.2.1. Workload

The workload consists of applications executing in *virtual machines (VMs)*, *containers*, or directly on *physical machines*. We focus in this work primarily on *business-critical workloads*, which are long-running, enterprise services integral to an enterprise's business, where unavailability or even degraded performance results in significant business interruption [135]. Such workloads span a wide range of user-facing and back-end services, such as email, databases, CRM, or management services, and by nature, differ vastly from workloads running in the datacenters of Google [127] and Microsoft [69].

Our model also considers scientific workloads deployed on virtualized environments. These workloads are primarily comprised of conveniently (embarrassingly) parallel tasks—e.g., Monte Carlo simulations—forming *batch bags-of-tasks*. Large high performance computing (HPC) workloads, such as scientific workloads from the healthcare sciences, also fit in our model. We consider also *app managers*, such as the big data framework Apache Spark, the machine learning framework TensorFlow, and the serverless framework OpenFaaS, which orchestrate virtualized workflows and dataflows for their users.

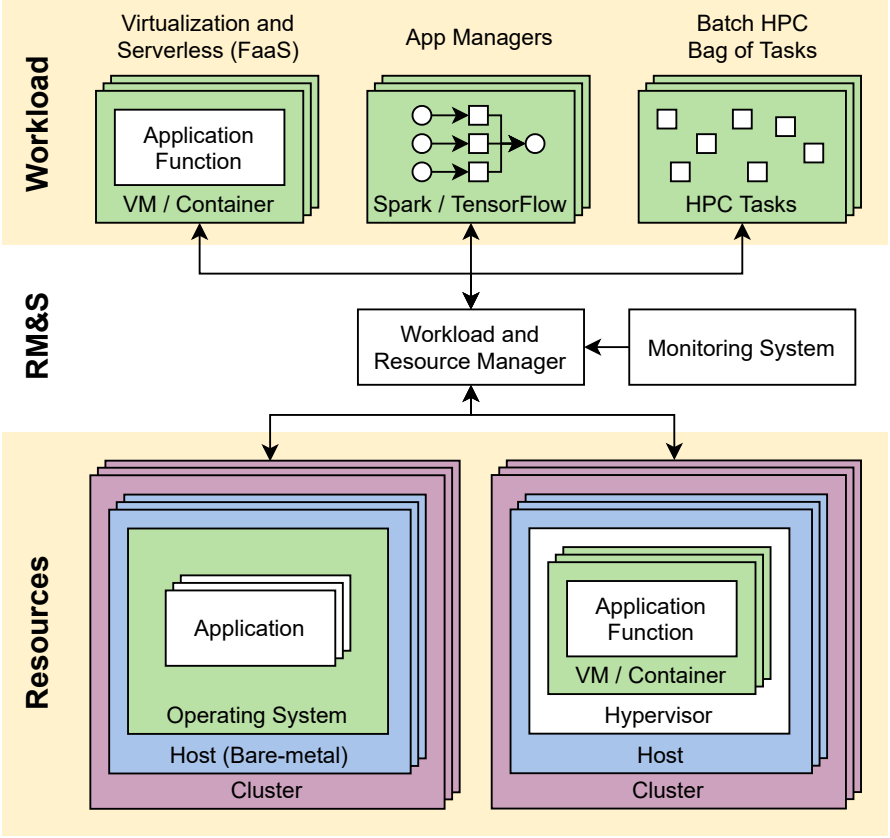


Figure 2.1: Generic model for datacenter operation.

2.2.2. Datacenter Resources

Workloads run on physical datacenter infrastructure. We model datacenter infrastructure as a set of physical clusters of possibly *heterogeneous hosts*, each host a node in a datacenter rack. A host can execute multiple VM- or container-workloads, managed by a *hypervisor*.

We model in this work resource consumption of applications (e.g., CPU usage) per discretized time slices. Workloads report at each time slice their resource consumption to the hypervisor, which consolidates the requests and distributes the resources based on some scheduling policy: CPU resources are allocated between the workloads that request it, through *time-sharing* (if on the same cores) or *space-sharing* (if on different cores). We assume a generic memory model, with memory allocation constant over the runtime of a machine. As is currently common in industry, we allow overcommission of CPU resources [21], but not of memory resources [135].

2.2.3. Operational Phenomena

Cloud datacenters are complex hardware and software ecosystems, in which complex phenomena emerge [80]. Given the absence of a general model, we consider two very common operational phenomena: (i) *performance variability* caused by performance interference between collocated VMs [93, 150], modeled using a CPU-contention predictor for demanding business-critical workloads [150], and (ii) *correlated cluster failures*, based on a common model for space-correlated failures [62], where a failure may trigger more failures within a short time span, which together form a group. Currently, we model only *full-stop failures*: machines crash fully, with subsequent recovery after some duration.

2.2.4. Resource Management and Scheduling

We model a workload and resource manager that performs management and control of all clusters and hosts, and is responsible for the lifecycle of submitted workloads, including their placement onto the available resources [11]. The resource manager is configurable and supports various *policies* to distribute workloads over the available resources.

2.2.5. Service Level Agreements

Customers of cloud providers demand high quality of service and the ability to scale rapidly on demand, but at a low cost. Even in absence of explicit guarantees, customers often develop their own expectations about the desired performance of services offered by the cloud provider. These expectations may, however, not match the assumptions of the designers and operators of the service. This dynamic can lead to disappointment, reputational damage, or even loss of customers, when the reliability or performance of services is lower than expected by the customers [27, Chapter 4].

Furthermore, the incentives of the cloud provider might not be aligned with that of its customers. While customers want optimal performance, cloud providers usually overcommit resources to serve more customers and reduce operational costs, leading to performance variability [147]. There is thus an inherent mismatch between the interests of the cloud provider and the customer (the "Principal-Agent Problem") [111].

To prevent potential disputes with customers and to set expectations about the performance of a service, the industry uses so-called *service level agreements (SLAs)*. An SLA is an explicit or implicit contract between the cloud provider and customer that governs the obligations and responsibilities between both parties regarding the provided service. SLAs establish (i) what kind of service is to be provided and how, (ii) constraints for the level of service (e.g., for availability or performance), (iii) the costs to be paid by the customer to the cloud provider, and (iv) the consequences for not upholding the agreement. Lawyers or business experts usually compose these SLAs, since they are closely connected to business decisions, and the consequence of breaching them can be substantial, from financial penalties to legal action from customers. Often, cloud providers share periodic reports with customers to demonstrate the provided level of service and compliance to the agreed SLAs. In case the cloud provider is unable to meet the terms of the agreed SLAs, the consequence is usually a financial penalty or rebate, but it can take other forms as well.

SLAs used by public cloud providers focus primarily on availability as key performance indicator. For cloud providers running business-critical workloads, SLAs usually include consolidation limits of virtual machines on physical machines, since these workloads cannot afford high latencies. There are also SLAs that require complete failover of customer workloads in the event of a total datacenter outage (e.g., as a result of a fiber cut or power outage).

Generally, the SLAs employed by the industry include a set of service level objectives (SLOs) that together define the expected service between the cloud provider and the customer. Each of these SLOs expresses unambiguously the service level that is guaranteed by the cloud provider for some measurable characteristic, for example, as a "monthly uptime percentage" of at least 99.995% for a VM. Often, actual values for SLOs are far from their expected outcomes, since they need to capture worst-case scenarios.

Designing good SLOs is a non-trivial problem [111]. The requirements from customers are complex and unclear, making them difficult to measure and to commit to promising reasonably for a cloud provider. Adding new SLOs also requires additional resources for collecting and processing the necessary data, all without significantly interfering with customer workloads, and without compromising the security or privacy of customers. Furthermore, certain SLOs depend on customer behavior; for example, customer running software may not be capable of achieving maximal performance (e.g., network throughput), making it difficult to provide effective performance guarantees.

SLOs are frequently specified as predicates over service level indicators (SLIs) that indicate whether the desired service level is met. A SLI is a quantitative measure of some aspect of the level of service that is provided [27, Chapter 4]. SLIs usually come from the monitoring systems in datacenters, and should, ideally, measure a service level of interest directly. When the desired metric is difficult to measure or compute, often a proxy is used. For instance, it is usually only possible to measure server-side latency, while client-side latency is often the apter metric for users. It is important that SLIs used in agreements can be measured objectively, as this avoids disagreements over interpretation and allows customers to verify SLOs independently.

2.3. A Primer on Datacenter Simulation

We employ in this work simulation to explore and experiment with datacenter infrastructure. In this section, we explain what simulation is and why it is useful in the context of datacenters.

2.3.1. What is Simulation?

Simulation is the "imitation of a real-world process or system over time, enabling the study of, and experimentation with, the internal interactions of complex systems" [17]. It is widely used in many domains of science and industry, such as computer systems [34, 36], aerospace engineering [123], and epidemiology [5, 56].

Table 2.1: Comparison of selected datacenter simulators.

Project	Environment	Stakeholders	Highlighted Features	GUI
CloudSim [34]	Cloud, Fog [70], Edge	Research	VC*, N, S, E, WF† [42], FD†, EXP†, CM, PI†	✓† [158]
SimGrid [36]	Grid, P2P, Cloud [108]	Research, Edu. [141]	VC*† [76], N*, S, E*, WF* [37]	✓† [37]
DGSim [79]	Grid	Research	WF, F, EXP	✗
GroudSim [118]	Grid, Cloud	Research	WF, CM, F	✗
iCanCloud [117]	Cloud	Research	VC, N*, S, CM	✓*
OpenDC [107]	Cloud	Research, Education	VC*, N, S, E*, CM, FS*, ML, WF, F*, PI, EXP*	✓*

Models: VC = VMs and containers, N = Network, S = Storage, E = Energy, CM = Cost models, FS = FaaS, ML = Machine learning, WF = Workflows, FD = Federation; **Phenomena**: F = Failures, PI = Performance interference, **Tools**: EXP = Experiment automation. **Support**: ✗= No, ✓= Yes.
† = extension, not integrated; ⋆ = advanced, carefully calibrated feature.

Simulation models are diverse in their approach and implementation. In general, we distinguish between three characteristics of simulation models [90]:

1. *Static* models represent systems at a single point in time, while *dynamic* models simulate systems over time.
2. *Continuous* models change state continuously over time, while *discrete* models advance in discrete steps in time.
3. *Stochastic* models include randomness with their outputs described by probability distributions, while *deterministic* models always perform the same given the same initial parameters.

In this work, we consider only a particular method of simulation, *discrete-event simulation* [17], where the operation of a system is represented as a sequence of events over time, with the assumption that no changes occur in-between events. This allows direct progression between events, in contrast to continuous models. Almost all efforts to model cloud and datacenter operations employ discrete-event simulation, due to the sheer scale and complexity of datacenters and long-running nature of experiments; OpenDC does the same.

2.3.2. Datacenter Simulators in the Field

In this section, we survey work related to OpenDC, for which we summarize the comparison in Table 2.1. The community has already built many high-quality simulators that provide a rich set of features to build upon [15, 33]. We select here the simulators closest in nature to OpenDC. The others offer typically a single feature, or are very general and thus require repeating the work we propose here for each specific model.

OpenDC (introduced in Section 1.4, but whose design we detail only in Section 3.3.2) proposes unique modeling advances, such as (i) the serverless model that is the first to detail the reference architecture proposed by SPEC [54] into an operational model, (ii) a detailed model for machine learning workloads based on TensorFlow, and (iii) prefabricated components for sharing designs. The integrated nature and the convenience features of OpenDC allow it to be deployed in practice quickly, even without deep expertise.

Closest alternative for research and development processes: OpenDC is closest in nature to the CloudSim ecosystem. CloudSim includes a high-quality simulator [34], which focuses on simulating cloud system components including virtual machines, data centers, and resource provisioning policies. It also includes numerous other single-feature simulators, such as iFogSim [70], WorkflowSim [42], and CloudAnalyst [158]. However, the single-feature simulators extend CloudSim each in their direction and cannot be combined without extensive engineering. In contrast, OpenDC offers an integrated approach, and specific modeling advances for its main features.

Similar is the SimGrid framework [36], which serves as the foundation of many simulators, such as SimGrid VM [76], Schlouder [108] and WRENCH [37]. In contrast to OpenDC, it is more general purpose (supporting not only clouds, but also P2P networks) and runs at a much finer granularity, which in turn enables

emulation of specialized applications (e.g., MPI). However, SimGrid and its ecosystem do not provide the advances described in the overall part of this section.

Alternatives for training and education: The use of simulation in education is generally not documented, but we assume it to be widespread. The notable emergence of WRENCH [38] shows the potential of simulation for teaching complex subjects in the computer science curriculum. OpenDC provided already extensive engagement with many categories of students and lengths and complexities of projects, which we have described in [107].

2.3.3. On the Benefits and Drawbacks of Simulation

Simulation might not be the most appropriate tool in all cases. We discuss in this section several aspects of experiments, evaluate the appropriateness of simulation, and compare it against other approaches.

Compared to Mathematical Analysis

One alternative to simulation is the use of mathematical analysis. Analytic models provide a fast and high level mathematical approach to predicting performance, but may encounter limited accuracy due to relying on pre-existing data from which the models are derived. Since analytical models operate at such a high-level, it is difficult to capture the scale and complex interplay (e.g., operational phenomena) of the distributed systems present in datacenters. Detailed analytical models can overcome some of the limitations of high-level analytical models, but the "curse of dimensionality" means investigating real-scale problems with them can be very time-consuming and even impractical.

Nonetheless, it is common to use a combination of analytic models and simulation, where subsystems of particular interest can be simulated to achieve a higher accuracy while other subsystems that are of less importance can be mathematically modeled.

Compared to Real-world Experimentation

Another alternative is to use physical infrastructure to perform real-world experiments. For instance, scientific projects, such as the DAS supercomputer [14] in the Netherlands, provide restricted but real environments for experimentation. Clouds offer the alternative of cheap infrastructure, but temporary and not as controlled. Such an approach provides results closest to real-world operating conditions. Nevertheless, this approach has several problems.

First, experiments on physical infrastructure are difficult to reproduce. Operational phenomena [147] affect system performance in non-trivial ways and in turn influence measurements of identical experiments.

Second, experiments on physical infrastructure are difficult to adapt or reconfigure. Such steps often require procurement and installation of new resources, which could be expensive. By employing simulation instead, we can quickly evaluate alternative scenarios with few costs.

Third and last, experiments on physical infrastructure are time-consuming, expensive, notwithstanding the environmental impact of such experiments. This impact can become unacceptable for even moderately sized infrastructure, as we find in Section 4.7, where we estimate the costs of running the experiments conducted in this thesis on physical infrastructure.

2.4. A Primer on Risk Management for Datacenters

We explain in this section what risk management is, describe how it is used in the context of datacenters, and survey the available literature to understand the current state-of-the-art in risk management.

2.4.1. What is Risk Management?

Risk management is the systematic (and often iterative) process of identifying, analyzing, and controlling risks. Risk, in this context, is defined as the "effect of uncertainty on objectives" in ISO 31000 [2], and can be positive, negative, or even both. Risk originates from a variety of sources, such as from legal liabilities, financial uncertainty, threats to IT infrastructure, accidents, and climate change.

Risk management is a fundamental part of any organization's strategy, and enables organizations to set out future objectives, to recognize potential risks that it could face, and to adequately control potential risks. Every organization is confronted daily with events that represent opportunities for benefit or threats to success. While a natural response for an organization is often to try to avoid all risk, such an approach is very difficult or outright impossible, and accepting or controlling some risks can actually lead to substantially more benefits for the organization. For instance, the flexibility and cost reduction obtained by workload consolidation in datacenters outweighs risks of security [92, 102] and performance [147] by a large margin.

Several generic standards and frameworks have been developed for risk management [2, 3, 124]. However, the actual definitions, methods, and objectives used for risk management vary widely across the different domains that employ it, including project management, engineering, security, and public health.

The first part of risk management is *risk identification*, which aims to recognize and describe risks that impact the future objectives of an organization. To do so successfully, it is critical that the organization has the appropriate and latest information, and that it also recognizes risks that are not under its control.

The second, key component of risk management is *risk analysis*, which concerns the comprehension of the nature of risk and its potential impact [2]. Depending on the focus of the analysis, the availability and quality of information, and the available resources, risk analysis can be conducted in varying degrees of detail and complexity. There exist multiple techniques for risk analysis, and choosing the appropriate technique often depends on the circumstances and intended use. *Qualitative risk analysis* uses a subjective assessment of the risk probability and impact, relying on the knowledge and interpretation of the assessor. On the other hand, *quantitative risk analysis* [155] is a systematic approach that tries to quantify both the likelihood and impact of risks numerically, relying on accurate, measurable data to produce insights. *Semi-quantitative risk analysis* uses a combination of both qualitative and quantitative methods to analyze risk, and generally provides a greater insight when it is difficult to quantify highly uncertain events (with possibly severe consequences). The risk analysis process provides insight into questions on whether certain risks need to be managed, and into selecting the most appropriate response to risk.

The final component of risk management is *risk response*, which is responsible for managing threats (uncertainties with negative consequences). Typical strategies for managing risks include (i) eliminating the risk all together; (ii) reducing the probability of the risk, the impact, or both (e.g., through redundancy and testing); (iii) transferring some or all of the risk to another party (e.g., through insurance); or even (iv) accepting the potential outcomes of a risk, for instance, when the cost of control is significantly higher than the consequences. A similar approach can be used to respond to opportunities (uncertainties with benefits).

2.4.2. Risk Management across Domains

Risk management is an integral organizational process employed in many different areas of industry and science, which include finance, information technology, project management, and healthcare.

Financial risk management [44] focuses on monitoring and controlling financial (or operational) risks, which involve the financial accounts of organizations. An example of financial risk management is the Basel III framework, which was introduced to reform financial regulations and mitigate risk in the international banking sector, following the financial crisis of 2007–2008. The Basel framework divides the concept of risk into market risk (price risk), credit risk, operational risk, and liquidity risk, while also establishing methods for determining the capital requirements for each of these risk types. Traditional measures for risk include value at risk (VaR), profit at risk (PaR), valuation ratio, and the Sharpe ratio.

Risk management is also present in engineering projects. In such projects, mistakes or other unexpected events can have severe consequences, from substantial economic losses to physical damage or injury. One example is aerospace engineering, where a simple mistake could render a multi-million-dollar satellite useless, or worse, lead to the loss of human life. NASA uses extensive, continuous risk management processes to ensure the safety, performance, feasibility, and punctuality of projects [49]. These processes employ qualitative assessments as well as quantitative analysis, for instance through high-fidelity simulations [123].

In datacenter facilities, risk management is used to improve reliability of the power distribution systems [157], to balance operational risk against electricity costs when participating in smart grid environments [164, 165], to safely apply emergency power upgrades [43], and to recognize the importance of appropriate airflow and heat transfer [130]. The datacenter industry also offers various services, tools, and methods for risk assessment, such as Uptime Institute's Data Center Risk Assessment [97].

2.4.3. Risk Management for Cloud Computing

Risk management is a critical process for ensuring high quality of cloud services. Although risk management for cloud computing has a shorter history than the approaches discussed previously, it is certainly not a new subject in cloud research.

The community has proposed several general methodologies and systems for risk analysis in the context of cloud computing, which we have summarized in Table 2.2. The AssessGrid project [50] introduces the notion of risk assessment in grids as a decision paradigm, using high-level mathematical models to analyze the impact of SLA violations. Similarly, the OPTIMIS [52, 57] project also employs mathematical models to quantify risks, taking into account historical SLA violations and estimates of the reliability of the environment.

Table 2.2: Comparison of selected publications on risk management for cloud computing.

Publication	Type	Stakeholders	Focus	GUI
AssessGrid [50]	Quantitative (M)	Research	SLA negotiation for Grids	✗
OPTIMIS [52, 57]	Quantitative (M)	Research	Cloud Brokering	✓
Yeo and Buyya [163]	Quantitative (M)	Research	Utility Computing	✗
Albakri et al. [6]	Qualitative	Research	Security	✗
SEBCRA [59]	Semi-quantitative	Research	Business Objectives	✗
Janus [8]	Quantitative (S)	Industry (Google)	Network Planning	?
RSS [162]	Quantitative (S)	Industry (Facebook)	Network Risk Identification	?
Radice (this work)	Quantitative (S)	Research, Education	Infrastructure Optimization, Sustainability	✓

Techniques: S = Simulation, M = Mathematical modeling; **Support**: ✗= No, ✓= Yes, **?** = Unknown.

Yeo and Buyya describe four objectives for balancing risk in grids, and develop two evaluation methods to validate the effectiveness of resource management policies in attaining these objectives [163]. Albakri et al. propose a qualitative method for security in cloud computing environments, which actively involves cloud customers in the evaluation of security risk factors [6]. SEBCRA [59] is a semi-quantitative cloud risk assessment model that focuses on evaluating the impact of cloud-specific risks on the business-level objectives of organizations, but requires knowledge from experts for establishing probabilities and impacts.

Furthermore, the community has developed several methods for cloud brokering that integrate risk assessment. Research in this area includes supporting SLA requirements [87], and more specifically, Quality of Service (QoS) requirements [104] in customer requests; negotiation of SLAs [25]; selecting clouds that minimize SLA violations [50, 51]; and utilizing spot instances with minimal risk [132].

Cloud operators have also incorporated risk management in their resource management and scheduling systems. Current approaches include using portfolio scheduling to dynamically select the scheduling algorithm based on operational and disaster-recovery risks [150], measuring the impact of QoS requirements on SLA violation rate [136], identifying the risk of resource oversubscription [142, 153], considering the security implications of workload consolidation [9], guaranteeing scalability of the infrastructure [31], and maximizing energy efficiency with minimal SLA violations [167].

Risk management is also used for computer networks. Janus [8] is a system for planning network changes used by Google, which uses flow-level Monte-Carlo simulations to evaluate the impact of various risk scenarios, based on operator specified risks and probabilities. A similar approach is used for planning the capacity of Google's network backbone [16]. The Risk Simulation System (RSS) [162] from Facebook identifies possible issues in the company's backbone and quantifies their potential impact using network simulation and a set of network risk metrics. The design of Radice follows an approach similar to these systems, but extends this model with support for compute infrastructure and sustainability metrics, whereas Janus and RSS focus solely on network infrastructure.

Herbst et al. propose a novel metric for quantifying operational risk of cloud services, which takes into account elasticity of the system and performance interference of services [74]. Other relevant work in the field has focused on incorporating green energy in SLAs [73], defining service level agreements involving security (also named secSLAs) [143], using the actor model for SLA lifecycle management [105], and monitoring SLAs using external sources (e.g., Twitter) [116].

3

Design of Radice

In this chapter, we address the first two research questions (RQ1 and RQ2) by presenting the design of a system for data-driven risk analysis of sustainable cloud infrastructure.

3.1. Overview

We design Radice, *the first instrument to facilitate data-driven risk analysis of sustainable cloud infrastructure,* employing discrete event simulation at its foundation. Our contribution in this chapter is four-fold:

1. We analyze the requirements for Radice (Section 3.2).
2. We then propose a high-level design for Radice's architecture (Section 3.3).
3. We describe how Radice facilitates the modeling and exploration of operational scenarios (Section 3.4).
4. We present the detailed design of Radice (Section 3.5).

We summarize our contributions of this chapter and discuss threats to validity in Section 3.6.

3.2. Requirements Analysis

We determine in this section the requirements that should be addressed by Radice. This matches stage (1) of AtLarge Design Process [81]. We begin by identifying the stakeholders of such a system and describing relevant use-cases. We then synthesize the functional and non-functional requirements for Radice.

3.2.1. Stakeholders

We identify five relevant stakeholders of the system:

(**S1**) **Customers** and **end-users** do not interact directly with the system, but have a substantial influence on the definition of risk for a cloud provider. Customers may employ various services used by the end-users and hosted by datacenter infrastructure of the cloud provider. They expect high quality of service and near-infinite scalability, but at low cost and high availability. Failure to offer the desired quality of service may be disastrous for the customer and in turn may carry hefty penalties for the cloud provider as stipulated by SLAs.

(**S2**) **Datacenter operators** manage the daily operation of the datacenter infrastructure of the cloud provider. They are responsible for ensuring the demands and requirements of the customers are met, maintaining efficient operation, but also monitoring the actual risk of the cloud provider, in order to prevent disruption of datacenter operation.

(**S3**) **Datacenter architects** design the datacenter infrastructure of the cloud provider. They must trade off various factors during the design of a new datacenter, such as the scale of the infrastructure, the hardware being used, and the cost of the design, while also taking into account how these factors impact the risk of the cloud provider.

Table 3.1: Mapping of use-cases to requirements

	FR1	FR2	FR3	FR4	FR5	FR6	NFR1	NFR2	NFR3	NFR4
Risk Monitoring (UC1)	✓	✓	✓	✓	✓	✓	✓	✓		✓
Capacity Planning (UC2)	✓	✓	✓	✓	✓	✓	✓			✓
Change Management (UC3)			✓	✓		✓	✓			✓
Compliance (UC4)				✓				✓		✓
Research (UC5)	✓	✓	✓	✓		✓	✓	✓	✓	✓
Education (UC6)			✓	✓					✓	✓

(**S4**) **Scientists** research and develop new techniques for designing, operating, and optimizing datacenter infrastructure. To foster adoption of new techniques, it is essential that stakeholders properly understand the risks associated with deploying these new techniques.

(**S5**) **Students** should be educated on datacenter infrastructure and large-scale computer systems. This group represents the future generation of engineers, and given the serious shortage of skilled personnel in the industry, must be supported in exploring their curiosity in computer systems and developing their talents.

3.2.2. Use-cases

Now, based on the stakeholders that we have identified, we devise six use-cases where Radice could be useful:

(**UC1**) **Risk Monitoring**
Datacenter operators can utilize the system to monitor in near real-time the current risk of the datacenter infrastructure and make informed decisions on how risk is being managed.

(**UC2**) **Capacity Planning**
The system can support datacenter architects during the capacity planning processes for cloud infrastructure, where smart decisions could lead to significant service improvements, cost savings, and environmental sustainability [19]. We have previously proposed *Capelin*, a data-driven, scenario-based capacity planning system for cloud datacenters [12]. By integrating Capelin, the system can provide capacity planners with risk-related insights to make better informed decisions.

(**UC3**) **Change Management**
Datacenter operators can utilize the system to evaluate how changes to the datacenter infrastructure (e.g., deploying new workloads or changing the scheduling policy) affect the risks they face, such as increased probability of failure events or degraded performance.

(**UC4**) **Compliance and Auditability**
The system can aid datacenter managers in showing their efforts towards reducing operational risk to (potential) customers and in demonstrating compliance with standards, contracts, and regulations.

(**UC5**) **Research and Development**
Researchers and scientists can employ the system to understand how risk affects and is affected by the various components of a datacenter, as well as evaluating new techniques for designing, operating, and optimizing datacenter infrastructure.

(**UC6**) **Education**
Educators can utilize the system to educate students about datacenter infrastructure, the numerous risk trade-offs that datacenter operators face, and the key decisions that influence risk.

3.2.3. Functional Requirements

Based on the use-cases we have envisioned, we now synthesize the functional requirements for Radice. In Table 3.1, we map each requirement to the presented use-cases.

(**FR1**) **Model cloud datacenter environments.**
The system should enable users to model cloud environments, supporting diverse resources, workloads, and policies, within the scope introduced in Section 2.2. Without FR1, Radice cannot support common risk scenarios (FR3) or identify risks applicable to cloud datacenters (FR4).

(**FR2**) **Enable expression of risk factors.**
The system should enable the user to express factors that quantify some aspect of operational risk in datacenters. As is currently the practice in commercial settings (e.g., in cloud or datacenter operations), these risk factors are formulated as quantitative objectives (e.g., SLOs), defining thresholds or ranges of acceptable values for a selection of metrics reported by the system, along with the impact of violating these thresholds, for instance through a monetary penalty. Without FR2, Radice cannot distinguish high-risk scenarios from low-risk scenarios (FR4).

(**FR3**) **Support common risk scenarios.**
The system should support risk scenarios that are common in cloud datacenters, such as failure events or performance degradation. Without FR3, Radice does not offer a realistic or accurate representation of the risks in a cloud datacenter (FR4).

(**FR4**) **Identify and explain IT-related risks in cloud datacenters, focusing on sustainability.**
The system should identify and report the diverse IT-related risks that surface in cloud datacenters. It should also present a brief explanation of the identified risks, supported by appropriate visualizations. Without FR4, Radice cannot provide useful insights to users about the risks in a datacenter.

(**FR5**) **Suggest optimizations that reduce risk, including sustainability risks.**
The system should explore possible optimizations of the topology and operational settings that reduce risk, and suggest appropriate changes. This significantly reduces the effort necessary from the users to identify applicable scenarios. Without FR5, users need to be aware of all possible risk-affecting scenarios and evaluate them manually, possibly overlooking key scenarios.

(**FR6**) **Integrate with the existing ecosystems for monitoring, decision-making, and sharing.**
The system should integrate with the existing ecosystem of systems, standards, and formats for datacenter infrastructure used by the industry and academia. Minimal effort should be necessary to incorporate Radice into users' existing workflows. Without FR6, we hinder the adoption of Radice.

3.2.4. Non-Functional Requirements

In addition to the functional requirements, we define four non-functional requirements for Radice:

(**NFR1**) **Provide in-meeting, near-interactive, albeit coarse estimates and precise same-day estimates.**
Cloud infrastructure currently operates at an unprecedented scale. The system should operate efficiently to support large-scale cloud environments and workloads. Without NFR1, Radice can not reasonably be used in interactive settings or for large-scale infrastructure.

(**NFR2**) **Facilitate reproducible science and experimentation.**
The results produced by Radice should be fully reproducible. Furthermore, the system should facilitate users in their reproducibility efforts. Without NFR2, we weaken the confidence of results produced by Radice and hinder the comparison of different scenarios.

(**NFR3**) **Adhere to modern software development standards.**
The system should not only be useful for this work, but should evolve and adapt to future work. To this end, the system should be engineered to professional, modern software development standards. Without NFR3, it is difficult to sustain the development of Radice or support extensions to the system.

(**NFR4**) **Provide insights at varying degrees of detail.**
The risk analysis process should respect inclusive design and support an incremental workflow. Users should be able to gain high-level insights with minimal effort and prerequisite knowledge of the system. Expert users should be able to obtain more in-depth results (e.g., raw data) and extend their focus on particular scenarios or components of interest. Without NFR4, Radice is unable to be used for the proposed use-cases without significant upfront investment from the stakeholders.

3.3. Overview of Radice

At the core of our approach for Radice is discrete-event simulation, coupled with a detailed modeling of energy operations, sustainability, and related risk metrics, supplemented with risk-analysis processes powered by this simulator. We depict in Figure 3.1 an overview of the Radice architecture. Radice builds upon OpenDC,

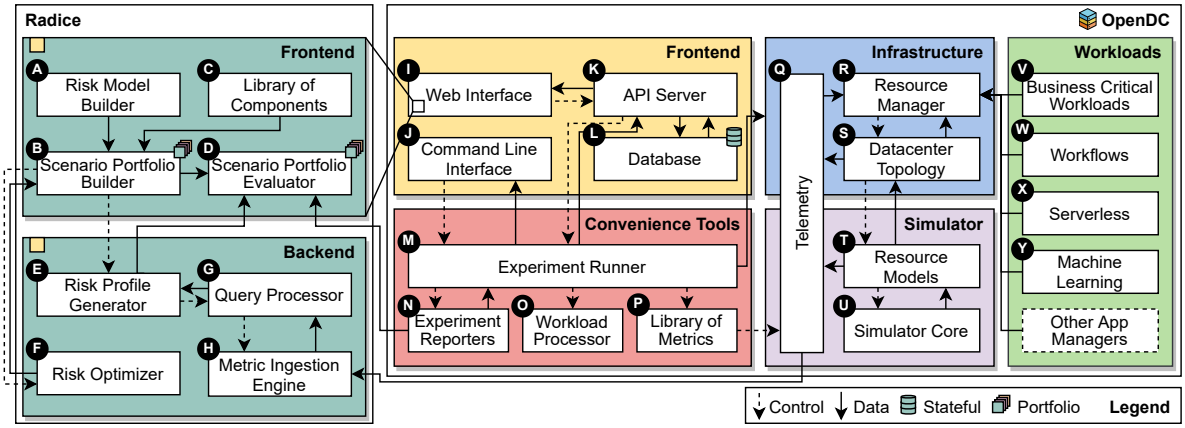


Figure 3.1: An overview of the high-level architecture of Radice and OpenDC.

a free and open-source platform for cloud datacenter simulation built through 4+ years of development and operation [107]. As we show in Sections 3.3.2 and 3.4.1, our work has contributed significantly to the design, development, and release of OpenDC 2.0. Following the AtLarge Design Process [81], we construct the Radice architecture iteratively. This process begins with bootstrapping the creative process (stage 3), after which we focus on the high-level and low-level design (stage 4).

We now discuss each main component of the Radice architecture. We elaborate on several low-level design decisions in Radice in Section 3.5.

3.3.1. The Radice Process

Radice integrates directly into the OpenDC platform. This enables Radice to leverage OpenDC's capabilities for datacenter modeling and allows other users of OpenDC to benefit from Radice's functionalities.

Users interact with Radice through OpenDC's user interface, where it starts with the *Risk Model Builder* (component Ⓐ in Figure 3.1). Through this component, the user can construct the risk model applicable for their datacenter design, defining risk factors of interest and the impact of violating these risk factors (FR2). Risk factors are expressed as predicates over (possibly) aggregated metrics for some time interval. This information is then used by Radice to estimate the risk of the datacenter.

Next, through the *Scenario Portfolio Builder* component (Ⓑ), users can visually construct scenarios to explore how they affect the risk of the datacenter, addressing FR3. These scenarios may take into consideration the workload, topology, scheduler, and operational phenomena. This component originates from Capelin [12], a system for scenario-driven capacity planning system that we recently proposed, and builds upon the notion of *portfolios of scenarios*, enabling users to experiment with datacenters in a structured, iterative way. In a portfolio, the first specified scenario is considered to be the *base scenario*, forming the baseline for all other scenarios in the portfolio. We elaborate further on our integration with Capelin in Section 3.4.2.

Scenarios can be composed using pre-built components from the *Library of Components* (Ⓒ). This library contains workload, topology, and operational building blocks, facilitating fast and intuitive composition of scenarios. This library is pre-populated by the system with a set of industry-standard components (for example, using the Open Compute Project[1] as starting point), but can be augmented by the user with platform-specific components. Allowing user-populated components benefits ease-of-use: If users are to include this in their frequent practice, any obstacles to quick interactive querying should be minimized.

Once built in the builder, users can explore and evaluate the estimated risk of scenarios in a portfolio via the *Scenario Portfolio Evaluator* (Ⓓ), addressing FR4. This component provides graphical overviews of the estimated risk, highlighting key selected metrics across scenarios, and giving users quick access to the outcomes of the simulation of built scenarios. Possible automated curation of the full results, showing the most relevant or interesting results first, could further reduce the time needed for users to gather the desired insights from their overview (NFR4).

The *Risk Profile Generator* (Ⓔ) converts the risk model specified by the user into a set of queries understood by the *Query Processor*. This component will collect the results of the query and compute the impact of risk factor violations. In turn, it generates a risk profile of each scenario, estimating the risk of each scenario.

[1] `https://www.opencompute.org/`

The *Query Processor* (G) aggregates metrics received from the simulator based on the active queries. This component is responsible for instructing the *Metric Ingestion Engine* what metrics to collect from the simulator. Queries processed by this component are evaluated on the fly during simulation, to enable dynamic decision-making based on the near real-time estimated risk, but also prevent huge volumes of data being generated. These queries enable the expression of risk factors, and this component therefore addresses FR2.

The *Metric Ingestion Engine* (H) is responsible for collecting the metrics emitted by the telemetry system in OpenDC during simulations. This component uses a pull-based model that collects metrics from the relevant systems in an on-demand fashion.

The *Risk Optimizer* (F) component generates alternative scenarios and optimizes them based on their risk profile. In Section 3.4.3, we describe the optimization process from the user's perspective, and in Section 3.5.4, we explain our exploration algorithm in detail.

3.3.2. The OpenDC Simulator

Radice employs OpenDC for cloud datacenter simulations, enabling stakeholders to model cloud datacenter environments (FR1). At the highest level, the OpenDC architecture is composed of three main components: (i) a web and textual frontend, (ii) a model-driven discrete-event simulator, and (iii) a set of tools to assist with simulation. In turn, the datacenter model, which we have described in Section 2.2, is a layered architecture representing the various abstraction levels offered by clouds, e.g., Infrastructure as a Service (IaaS), Platform as a Service (PaaS), building upon a small simulator core. We now discuss, in turn, each layer and sub-layer.

Frontend

In Figure 3.1, the *Web Interface* (I) serves as the user-portal, through which stakeholders can interactively construct, share, and re-use datacenter designs. With these designs, users can configure and conduct experiments. At any time, users can explore the automated plots and visual summaries generated by OpenDC, from single setups to comparative experiments. The *API Server* (K) responds to web requests, acting as intermediary and business-logic between the web frontend on the one side, and database and simulator on the other side. The *Database* (L) manages the state of the simulation platform, including topology models, historical data, simulation configurations, and simulation results.

Users may also deploy the simulator as a standalone package and utilize its *Command Line Interface* (J). Although this does not provide the same usability and accessibility as the web interface, it is useful for conducting experiments in headless environments and simplifies reproducibility efforts.

Simulator

The foundation of OpenDC is the simulator. The *Simulator Core* (T) provides a small set of primitives to enable simulated components using discrete-event simulation, such as an event queue. The *Resource Models* (U) model generic resource-sharing semantics, which are used to represent the behavior of datacenter resources and their scheduling policies. The simulator coordinates with cloud-level operational models, IaaS and PaaS, which we describe in the following, in turn.

Infrastructure

The *Resource Manager* (R) component models a typical IaaS platform, such as AWS EC2, from where users can lease compute resources on-demand. Internally, the *Resource Topology* (S) represents the resources available in the simulated datacenter, ranging from physical cluster nodes to VMs to containers (see Section 2.2.2). The *Telemetry* (Q) component is responsible for monitoring the datacenter resources.

Workloads

This level supports the execution of many platform-level operations, programmatically. This enables a variety of application types, as described in Section 2.2.1. *Business-critical workloads* (V) run directly in VMs procured from the IaaS platform. App managers provide an additional layer of abstraction to facilitate workload execution. For instance, the *Workflows* (W) component implements a generic workflow orchestration engine. The *Serverless* (X) component models a *Function-as-a-Service* (FaaS) platform, such as AWS Lambda, while the *Machine Learning* (Y) component models the TensorFlow machine learning framework in the cloud.

Convenience Tools

Supporting the simulation process is a collection of tools. The *Experiment Runner* (M) automates the orchestration of experiments using OpenDC, enabling users to write declarative experiment specifications, and

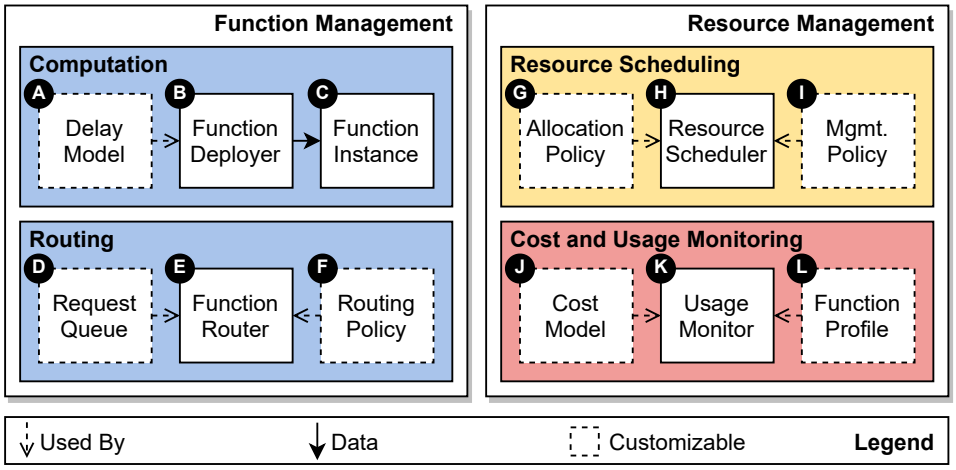


Figure 3.2: Architecture of the serverless model in OpenDC.

perform automated experiment design and optimization (e.g., evolutionary optimization). The *Environment Processor* (N) is responsible for processing the datacenter designs of users and adds mechanisms to assemble, provision, and configure the simulated infrastructure. The *Workload Processor* (O) facilitates reading, writing, and processing of workload traces in many of the formats used by the community. Currently, our tooling supports parsing, processing, and converting workload traces from the Grid Workload Archive [78], the Workflow Trace Archive [154], WorkflowHub [47], the Parallel Workloads Archive [55], and several internal and ad-hoc formats. The *Library of Metrics* (P) consists of a vast and diverse set of metrics utilized by the community, ranging from resource-level metrics (such as CPU usage or energy consumption) to application-level metrics (such as workflow makespan), provided by OpenDC and ready to be used by the user.

3.4. Modeling and Exploration of Operational Scenarios

We describe in this section how Radice facilitates the modeling and exploration of operational scenarios. In Section 3.4.1, we present the advances in OpenDC to support diverse short-term operational scenarios. Then, in Section 3.4.2, we demonstrate how Radice integrates with Capelin to support modeling of long-term operational scenarios. Finally, we describe in Section 3.4.3 how datacenter operators can combine these approaches to effectively manage and optimize risk in cloud datacenters.

3.4.1. Precise Modeling of Diverse Short-term Operational Scenarios (FR3)

Simulators need to *support frequent and substantial innovation* in cloud datacenters. The diversity of users and the rapid emergence of new technologies means requirements can change drastically over both short and long periods of time, as observed in grids [101] and clouds [127]. In addressing this challenge, OpenDC is the first simulator to integrate serverless computing and machine learning workloads, both emerging technologies already offered by all major cloud providers. This functionality has been developed in collaboration with two students from the Vrije Universiteit Amsterdam [107].

Simulation of Serverless Workloads

Serverless computing encompasses cloud services that abstract operational concerns, such as resource provisioning and load-balancing, away from the user. They provide an event-driven interface and charge users at a much finer granularity than the traditional cloud computing services.

Motivated by the promise held by serverless computing, the SPEC Research Group has investigated the properties of tens of real-world serverless platforms and proposed the SPEC RG Reference Architecture for Function-as-a-Service (FaaS) [54]. This reference architecture is one of the first to provide a systematic approach to designing serverless platforms for which serverless computing is charged as a cloud service. OpenDC takes this high-level descriptive model, and adds to it a detailed, fine-grained, operational model.

Figure 3.2 depicts the OpenDC detailed design for FaaS operations. At the core of this design are new components for function computation, function routing, and usage monitoring, which we discuss in the remainder of this section, in turn.

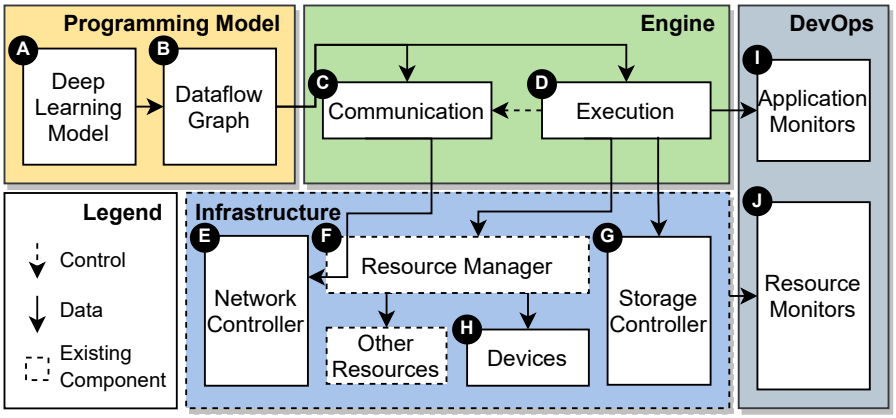


Figure 3.3: Architecture of the TensorFlow model in OpenDC.

The *Computation* component provides all the logic surrounding the *Function Instance* (C), our model of a FaaS execution container. The *Function Deployer* (B) decides *how* the function instance is deployed on the available resources, on invocation. A special component, the *Delay Model* (A), simulates delays that typically occur during deployment, such as cold starts or lookup delays; we envision this will evolve as the community develops more advanced serverless platforms.

The *Routing* component is responsible for routing invocation-requests to available instances of their respective function. The *Function Router* (E) is the brain of this component. It uses a customizable *Request Queue* (D) to enqueue invocation requests, and a configurable *Routing Policy* (F) to select an available function instance to route a request to. OpenDC provides many classic policies, such as selecting a random available instance or the instance with the least cumulative idle time, which the community can complement.

The *Resource Scheduling* (H) component manages the datacenter resources on which the function-instances run. Users can configure it through two scheduling policies: the *Resource Management Policy* (I), which governs the lifetimes of function instances, and the *Allocation Policy* (G), which decides on the appropriate VM for each containerized instance.

To monitor individual functions, the *Usage Monitor* (K) works with *Function Profiles* (L). Each profile contains a selection of metrics, data structures, and other characteristic elements. The usage monitor employs a *Cost Model* (J) to determine the cost of computations using a variety of customizable cost functions.

Simulation of TensorFlow

Machine learning (ML) and deep learning have gained much recent attention due to their great potential in numerous areas [28], including speech recognition, medical image analysis, and product recommendation. Current ML applications can have large data and computational requirements, which makes cloud datacenters natural environments to execute them.

Among the many approaches developed to enable ML-use in complex applications, TensorFlow [4] is one of the most prominent and representative ML frameworks. Yet, this raises new challenges, such as data management. To explore and improve the operation of TensorFlow in datacenters, we extend OpenDC with a model for the TensorFlow ecosystem. Our detailed, fine-grained model captures TensorFlow workload execution and communication.

Figure 3.3 depicts the architecture of the TensorFlow model in OpenDC. The *Resource Manger* (F) allocates and deallocates resources using various policies. We use simple models for networking and storage: The *Network Controller* (E) is a simple network model to control the data-flow between machines, considering bandwidth, but not more complex network features. The *Storage Controller* (G) models persistent storage used during execution. We also provide an extension point, *Devices* (H), for heterogeneous resources.

Our TensorFlow model considers application, execution, and communication aspects. In OpenDC, to ensure generality beyond TensorFlow, an ML application can be modeled as a high-level *Deep Learning Model* (A) or a detailed *Data-flow Graph* (B). The *Execution* (D) component uses different strategies to orchestrate jobs across machines for distributed training (such as the parameter server strategy), and executes two types of operations (for mathematical computation and communication). The *Communication* (C) and Execution components collaborate to support different communication methods (e.g., asynchronous communication). The *Application Monitors* (I) record application-level metrics for TensorFlow users. Similarly, *Resource Monitors* (J) keep cluster-level metrics for sysadmins.

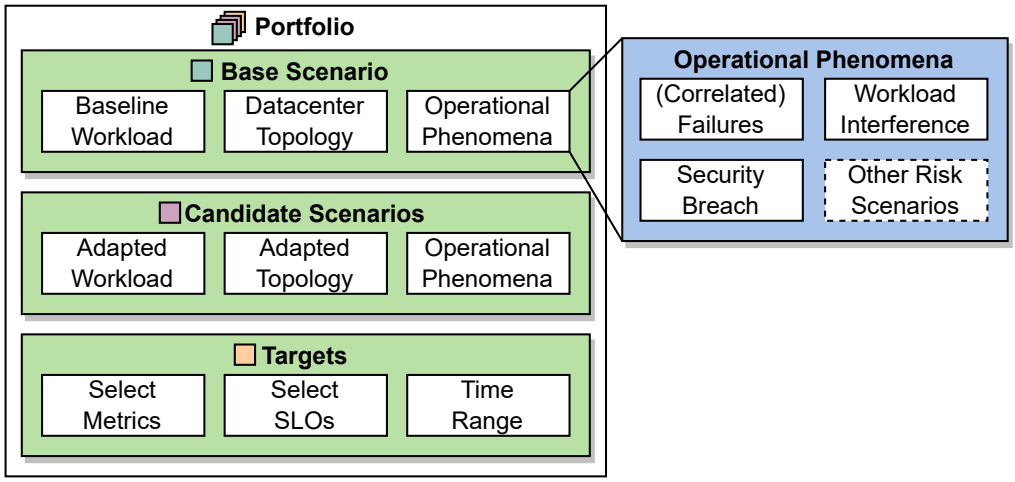


Figure 3.4: Abstraction of capacity planning portfolio used by Capelin [12], consisting of a base scenario, a number of candidate scenarios, and comparison targets. Radice re-uses this abstraction from Capelin and introduces it in the risk analysis process for datacenters.

3.4.2. Precise Modeling of Long-term Operational Scenarios with Capelin (FR3)

Over the lifetime of a datacenter, its designers and operators encounter complex decisions with long-lasting consequences, which require insight into long-term operational scenarios. An example of this is *long-term capacity planning*, which is the process of procuring machines that form the datacenter capacity. Although many approaches to the long-term capacity planning problem have been published [12, 35, 166], much of the industry still relies on rule-of-thumb reasoning for procurement decisions.

Recently, we proposed Capelin, a scenario-based capacity planning system that helps practitioners understand the impact of alternatives [12]. Capelin introduces a new abstraction of *portfolios of capacity planning scenarios*, by organizing multiple scenarios into a portfolio (see Figure 3.4). Each portfolio includes a base scenario, a set of candidate scenarios given by the user and/or suggested by Capelin, and a set of targets to compare scenarios. In contrast, most capacity planning approaches in published literature are tailored towards a *single* scenario—a single potential hardware expansion, a single workload type, one type of service quality metrics. We believe that this approach does not adequately cover the complexities that capacity planners face in 2022. The multi-disciplinary and multi-dimensional nature of capacity planning call for a novel approach to capacity planning, based on *multiple* scenarios.

A scenario represents a point in the capacity planning (datacenter design) space to explore. It consists of a combination of workload, topology, and a set of *operational phenomena*. Phenomena can include correlated failures, workload interference, security breaches, etc., allowing the scenarios to more accurately capture the real-world operations. Such phenomena are often hard to predict intuitively during capacity planning, due to emergent behavior that can arise at scale.

A portfolio also includes a set of targets that prescribe on what grounds the different scenarios should be compared. Targets include the metrics that the practitioner is interested in and their desired granularity, along with relevant SLOs [111]. Following the taxonomy defined by the performance organization SPEC [74], Capelin supports both system-provider metrics (such as energy efficiency and resource utilization) and organization metrics (such as performance variability and throughput rates). The targets also include a time range over which these metrics should be recorded and compared.

Radice builds upon the abstractions introduced by Capelin, extending it with functionality for risk analysis of datacenters. The risk modeling approach of Radice enables full coverage of SPEC's taxonomy [74], quantifying risks, as well as costs of cloud infrastructure, as we depict in Figure 3.5

3.4.3. New Capabilities for Risk Analysis and Exploration (FR4, FR5)

Combining the modeling capabilities for short-term operations in OpenDC, with the support for long-term operational scenarios through integration with Capelin, enables Radice to effectively manage and explore risks that emerge frequently in cloud datacenters. In this section, we present a systematic process for approaching risks in datacenters using Radice, for which we show a summary in Figure 3.6.

Before Radice can be used to manage and explore risks, it needs to be configured, along with the underlying OpenDC platform, by connecting these tools to the target datacenter infrastructure. This consists

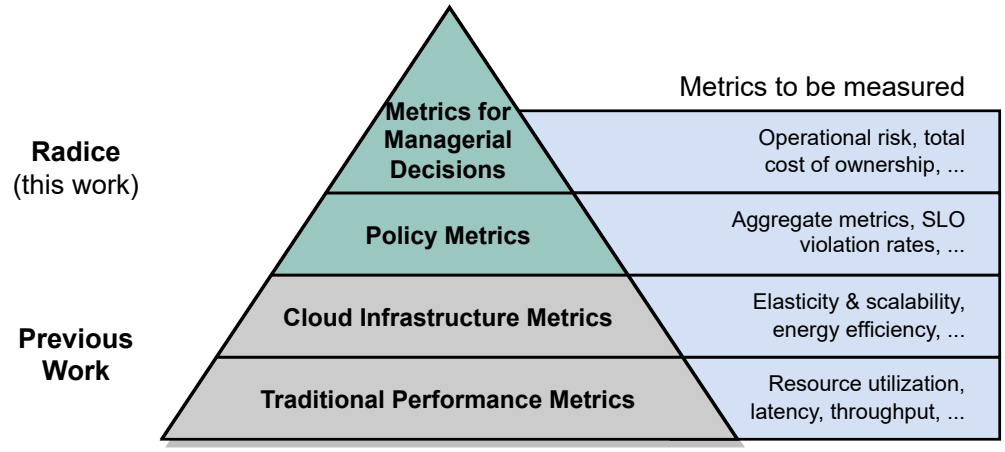


Figure 3.5: Hierarchical taxonomy of cloud metrics proposed by the performance organization SPEC [74]. Radice extends Capelin with support for higher-order metrics, such as service costs or operational risk, to enable full coverage of SPEC's taxonomy.

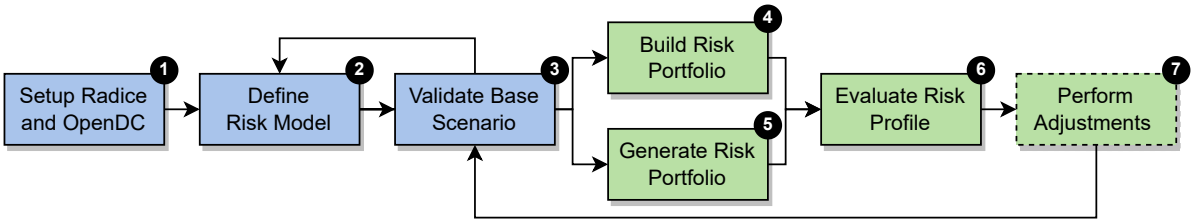


Figure 3.6: The process for optimizing risk in cloud infrastructure using Radice.

of (1) constructing a model of the datacenter infrastructure in OpenDC, by defining the physical and logical topology of the hardware and services in the datacenter; (2) providing operational data of the datacenter that characterize its workload to OpenDC, in the form of a trace or through a direct integration with monitoring systems; and (3) configuring the services to be simulated by OpenDC, for instance, by selecting the allocation policy used by resource scheduler.

The next step in the process is to define a risk model for the datacenter. A risk model describes the risk factors that constitute the overall risk in a datacenter, specifies how these factors are quantified, and assigns an impact to each factor. In Section 3.5.1, we describe in detail how such a risk model is constructed. After defining an initial risk model, it is important to validate the model using the results reported by Radice, for instance, using historical measurements, and to (potentially) (re-)calibrate the model using the results from the validation step.

Once the risk model is established, Radice can be used to explore portfolios of scenarios that show that the current infrastructure is at risk or that reduce overall risk. Users can build such portfolios manually, for example, to explore "what-if" questions about the workload, topology, or operational phenomena. Alternatively, Radice can automatically propose portfolios of scenarios that reduce risk in the datacenter, by employing the risk optimization algorithm described in Section 3.5.4.

Radice evaluates the portfolios of scenarios constructed by the user or generated automatically, and produces corresponding risk profiles, which highlight, for each of the scenarios, the risks that threaten the datacenter. These risk profiles assist datacenter operators in deciding how to manage different classes of risk. Certain risks might be accepted due to their low impact or probability, while other risks may be mitigated by making adjustments to the datacenter infrastructure. After the risk profiles are evaluated, and adjustments are potentially implemented, we jump back to the third step to again validate the base scenario and repeat the remainder of the process.

3.5. Detailed Design of Radice

We present in this section the detailed design of Radice.

3.5.1. Model for Quantifying Risks in Cloud Datacenters (FR2)

In this section, we present the design of our model for quantifying risks in cloud datacenters We represent in this work risk as a portfolio of *risk factors* that each model an aspect of the operational risk of a datacenter, quantifying it in a specific dimension. These risk factors are composed of possibly several SLOs (predicates over metrics), an aggregation time period (e.g., monthly or hourly), and an impact function. Every aggregation period, the active SLOs are evaluated, and if in violation, the impact is computed using the impact function. Combined, the impact of all SLO violations sum to a single value, representing the overall risk.

We distinguish between three categories of risk: customer-facing risks, company-facing risks, and society-facing risks. *Customer-facing risks* are risk factors that directly affect the customer, which include the availability of resources, quality of service, or security. Risk factors that affect the sustainability of the company, but do not directly affect customers, are categorized as *company-facing risks*. Operational efficiency is an example of such a risk. Finally, *society-facing risks* are risks that affect society as a whole, but do not directly affect the company or whose impact for the company is diminutive, such as environmental sustainability.

To estimate the impact of risk factors, Radice assumes financial impact as opposed to some abstract unit of risk. By expressing risk in terms of monetary value, we ensure a fair comparison between different scenarios, since our model can take into account the cost of other risk responses, such as risk mitigation, elimination, or transfer, as well. This might entail the costs of upgrading the datacenter topology, or the costs for purchasing insurance. Moreover, this approach enables users at different layers of the organization, from technicians to managers, to grasp the impact of both short-term and long-term decisions in datacenters.

Although the impact of some risk factors might be difficult to express in terms of financial cost, we argue that the existing risk management processes in organizations should already yield a rudimentary overview of the impact of potential risks that the organization faces, albeit qualitatively. The estimation does not need to be very precise, but needs to match the magnitude of the impact.

The strategy for specifying the impact function mostly depends on the type of risk. The impact of customer-facing risks can often be derived from the penalties for SLA violations, as stipulated by the contracts between company and customer. Company-facing risks can be quantified based on the operational expenses of the organization. These include the expenses for electricity usage, but also increased engineering costs as a result of problem troubleshooting. An impact function for society-facing risks is often more difficult to define and requires estimations from scientific literature if available.

3.5.2. Compute Scheduler (FR1)

The scheduler is a key component of datacenter infrastructure management and is responsible for deciding the placement of new VMs onto the available physical hosts. OpenDC's compute scheduler is designed after the Filter Scheduler[2] from OpenStack, a popular open-source cloud computing platform.

To decide on the placement of new VMs, the scheduler uses a two-step process that consists of *filtering* and *weighing*. During the filtering phase, the scheduler filters the available hosts based on a set of user-configured policies (e.g., based on the number of available vCPUs, the remaining RAM, or affinity rules). In the weighing phase, the scheduler uses a selection of policies to assign weights to the hosts that survived the filtering phase. From a subset of the highest ranking hosts, the destination of the VM is then selected randomly, in order to prevent overloading a single host when multiple candidates share the same rank. How the weights are determined is configurable by the user, but by default the scheduler will attempt to spread VMs across all hosts evenly based on the available RAM. This policy is similar to the default behavior of OpenStack's scheduler[3] and is actively used in production datacenters [149].

Kubernetes employs almost exactly the same process[4], but is configured by default to use more extensive weighing policies, ensuring workloads are balanced over the hosts, while also taking into account dynamic information, such as resource utilization. A key difference with OpenDC is that Kubernetes does not consider the memory requirements of workloads when weighing the hosts. VMWare vSphere offers the Distributed Resource Scheduler (DRS) to schedule VMs in a cluster and automatically balance workloads across hosts in the cluster based on memory requirements of the workloads.

In Section 4.2.3, we describe the implementation of the filters and weighers available in OpenDC, and explain how we can use these primitives to construct traditional scheduling policies used by the community.

[2] `https://docs.openstack.org/nova/latest/user/filter-scheduler.html`
[3] `https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html#id18`
[4] `https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/#kube-scheduler-implementation`

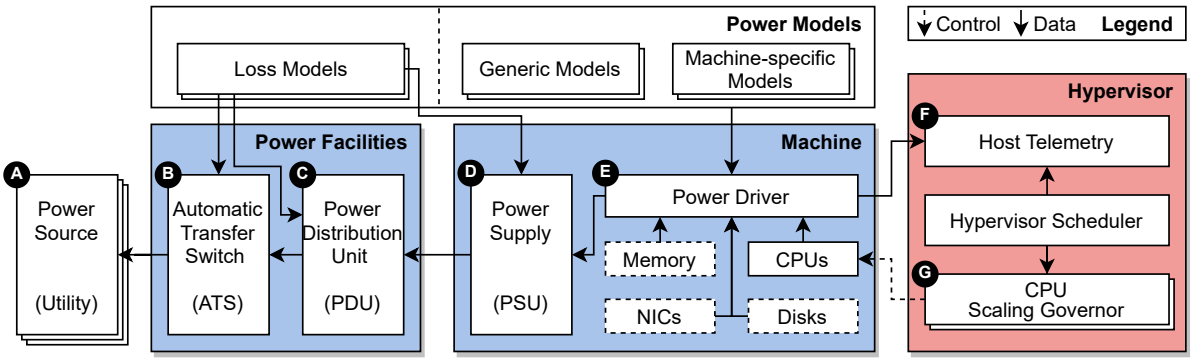


Figure 3.7: Architectural overview of the advanced power modeling subsystem in OpenDC.

3.5.3. Advanced Power Modeling Subsystem (FR1)

Power distribution systems play a vital role in datacenters. They ensure that the IT infrastructure in the datacenter has reliable and efficient access to electricity, even in the presence of grid failures.

The author of this thesis has supervised an Honors B.Sc. student tasked with designing an advanced system for energy modeling and power management in OpenDC. Having access to such a model allows OpenDC to capture the complex interplay between IT infrastructure and power distribution in datacenters, and support fine-grained metering of electricity in datacenters from source (e.g., the grid) to consumer (e.g., servers).

Figure 3.7 depicts the architecture of the power distribution system in OpenDC. Electricity enters the datacenter from the utility, backup generators, batteries, or any other generic power source (Ⓐ). The automatic transfer switch (Ⓑ) provides fail-safe power redundancy, in the event one of the datacenter's power sources fail. The electricity is then distributed by power distribution units (Ⓒ) and supplied to the servers via their power supply units (Ⓓ). For each of these power distribution devices, our design considers energy losses that occur in these devices, for instance, using the models proposed by iCanCloud [39, 117].

The *power driver* is responsible for estimating the power consumption of the machine (Ⓔ). Its design is extensible and supports simple models that derive power consumption from CPU utilization, as well as more complex models that support dynamic frequency and voltage scaling (DVFS) (e.g., through P-states) or integrate power consumption of off-chip components. To allow the hypervisor to query the power consumption of the machine, the power driver exposes a programmatic interface similar to the RAPL [48] technology available in Linux. DVFS functionality in the hypervisor is modeled after the design of the `CPUFreq` subsystem in Linux [32] (Ⓖ). Linux uses hardware-independent policies called *governors* to take decisions to adjust CPU frequency. The scaling rules therein are based on an estimation of the required CPU capacity (reported by the process scheduler). OpenDC follows the same approach.

3.5.4. Genetic Risk Optimization Algorithm (FR5)

To optimize a datacenter for risk using Radice, we employ in this work an evolutionary approach inspired by the process of natural selection. Using this approach, each point in the design space is encoded as a set of chromosomes that can be altered using genetic operators (such as mutation or crossover) [110]. Note that our work is not focused on designing the best possible optimization algorithm for Radice. Our intention is to demonstrate that the model of Radice can be adapted to support risk exploration. Future work could investigate more efficient methods for exploring the risks in cloud datacenters using Radice.

On the Necessity of Smart Exploration Algorithms

Risk exploration can be considered a form of design space exploration, which selects candidate solutions based on their risk profile. Various methods for design space exploration exist. Often, the most straightforward approach is to perform a brute-force (exhaustive) search of the design space, enumerating every possible candidate. While brute-force search is effective for smaller design spaces, it quickly becomes infeasible as the size of the design space grows larger, resulting from a phenomenon called *combinatorial explosion*.

In this section, we show that the problem of datacenter optimization suffers from combinatorial explosion, and thus the need for smart exploration algorithms in this area. We use a three-point estimation to understand the feasibility of an optimistic, pessimistic, and most likely scenario. In our estimation, we assume that simulating a single point in the design space requires exactly 5 seconds, our hardware is capable of 64 simulations in parallel, and that we perform 128 replications for statistical confidence.

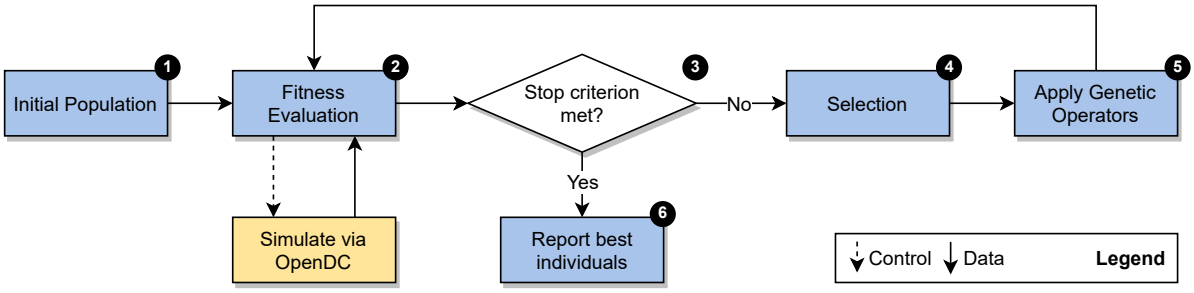


Figure 3.8: Exploration of the design space through genetic search.

Best-case scenario: Suppose we want to find the best combination of datacenter topology and scheduling policy for a datacenter running a single workload. We consider only the pre-built configurations offered OpenDC, which consist of 20 topologies and 11 scheduling policies. In this situation, our design space contains at least $11 \times 20 \times 4 = 220$ points. Based on our assumptions, an exhaustive search of this design space would require approximately 35 minutes, and would therefore be a feasible approach.

Average-case scenario: Instead of examining only pre-built configuration, suppose we consider custom scheduler and topology constructions, but limit exploration to only 10 values per parameter. With the very modest assumption of four parameters to construct the scheduler and four parameters to construct the topology, we arrive at a design space with a magnitude of $10^4 \times 10^4 = 10^8$ (100 million points). Exhaustive search of this design space would require over 30 years to complete given our assumptions, and thus makes such a brute-force approach infeasible.

Worst-case scenario: In reality, the performance of a datacenter relies on many factors, including software, hardware, network design, cooling, and power distribution, If we were to consider all these factors along with all possible values of their parameters, the resulting design space would be astronomically large, making a brute-force approach *intractable*.

Genetic Optimization Process

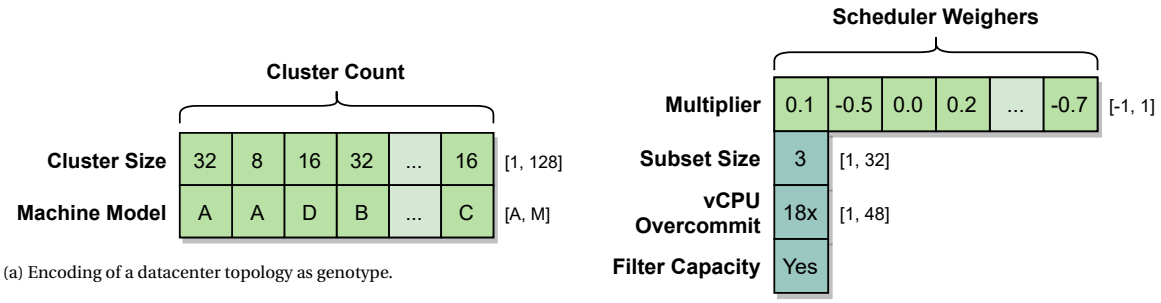
Below, we outline the optimization process used by Radice, of which we show a depiction in Figure 3.8. A description of the implementation as well as the algorithm parameters is given in Section 4.4.6.

1. A population of random scenarios (individuals) is generated based on the population size and seed specified by the user. In Section 3.5.4.3, we describe the encoding of scenarios into chromosomes.
2. The fitness of every scenario in the population is computed. We employ OpenDC to evaluate every scenario and estimate the risk, each consisting of multiple simulation runs to take into account variability.
3. The stop criteria are evaluated. Criteria might include limiting the total number of generations or waiting until the algorithm has converged to some solution. Once one of the stop criteria is satisfied, we jump directly to step 6 to complete the algorithm execution.
4. A selection of the fittest individuals is made from the current population.
5. Genetic operators are applied to the selection. These operators are used to either converge or diverge the solution by altering the chromosomes of individuals in the population, which helps explore the design space. We repeat the process and jump to step 2 to evaluate the new population.
6. After the genetic algorithm completes, Radice reports the best-performing scenarios to the user.

Encoding of Cloud Datacenters

Essential for the operation of a genetic algorithm is the encoding of the problem domain into chromosomes, given that it greatly influences the design of the operators that act on it (especially crossover). It is important to use an appropriate genetic representation for genetic exploration, since the particular choice of encoding scheme can significantly impact the performance of the algorithm [7].

We construct in this work a simple representation of a cloud datacenter for the genetic algorithm. Our representation considers two core elements of cloud datacenters, the datacenter topology and the scheduler. An overview of the encoding schemes used in the work is shown in Figure 3.9.



(a) Encoding of a datacenter topology as genotype.

(b) Encoding of a scheduler configuration as genotype.

Figure 3.9: Encoding schemes used for modeling cloud datacenters in our genetic optimization algorithm.

We represent the datacenter topology as a collection of clusters, where all machines in the cluster are identical (*homogenous*), but different clusters may use different hardware. In our encoding, there are two genes for each cluster representing the number of hosts in the cluster and the hardware model of the machines in the cluster, respectively.

Our representation of the datacenter scheduler is based on the Filter Scheduler architecture (described in Section 3.5.2). Each weigher configured in the scheduler is assigned a gene representing a multiplier for that weigher. This allows the genetic algorithm to explore different orderings of hosts. For example, a negative multiplier reverses the ordering produced by a weigher, while a very small multiplier effectively disables the weigher. In addition, the representation also includes genes for other parameters of the scheduler (e.g., vCPU overcommit ratio), to allow the genetic algorithm to explore the impact of these parameters as well.

3.5.5. Design for Reproducibility (NFR2)

Although reproducibility is a key principle of scientific research, the "replication crisis" [103, 121] highlights a growing problem in science, in which many articles are difficult or impossible to reproduce. The field of computer science faces the same challenges of reproducibility as encountered in other areas of science, such as psychology or medicine [45]. For instance, artifical intelligence (AI) research has been criticized for lack of transparency and reproducibility [77], with many articles not sharing source code, results depending heavily on the chosen hyperparameters, and models becoming prohibitively expensive to (re)construct (e.g., the well-known GPT-3 model is estimated to have cost $10–$12 million [131]). In cloud computing, Uta et al. have shown that cloud variability is often disregarded by the community, finding that many articles underspecify cloud-based experiments or conduct too few replications [147].

The design Radice and OpenDC emphasizes reproducibility and aims to assist researchers with this challenge (addressing NFR2). To guide our design, we consider the methodological principles for reproducible performance evaluation proposed by Papadopoulos et al. [120].

Principle 1 of [120] states the importance of repeated experiments to ensure results are not due to chance. Uta et al. show how variability of measurements can significantly impact findings in cloud research [147]. To fulfill this principle, OpenDC includes built-in functionality for running multiple repetitions of experiments. In our experiments, we were able to run 4,096 repetitions per scenario (see Section 4.4.1). More importantly, the high performance of OpenDC (as demonstrated in Section 4.5) allows researchers to actually run a large number of repetitions without requiring prohibitively large or expensive compute infrastructure.

Furthermore, although good experiment design necessitates representative coverage of the design space, authors often do not cover the design space sufficiently, and instead select parameters (e.g., workloads) based on ease of use [120]. To address this issue, OpenDC facilitates wide coverage of configurations and workloads through its diverse and detailed simulation models, which support emerging technologies, such as serverless computing and machine learning workloads, as described in Section 3.4.1 (fulfilling **Principle 2**).

Another common issue is articles providing insufficient information about the environment in which the experiments were carried out [120]. Experiments can inadvertently rely on environmental parameters, which may lead to different results if reproduced in a different environment. Addressing this, OpenDC includes in the produced results, a description of the hardware and software environment used to conduct the experiments, as well as other relevant environmental parameters (fulfilling **Principle 3**). Even so, the use of discrete-event simulation and seeded randomness in OpenDC, allow it to have a deterministic simulation process, and in turn produce reproducible results, independent of the execution environment.

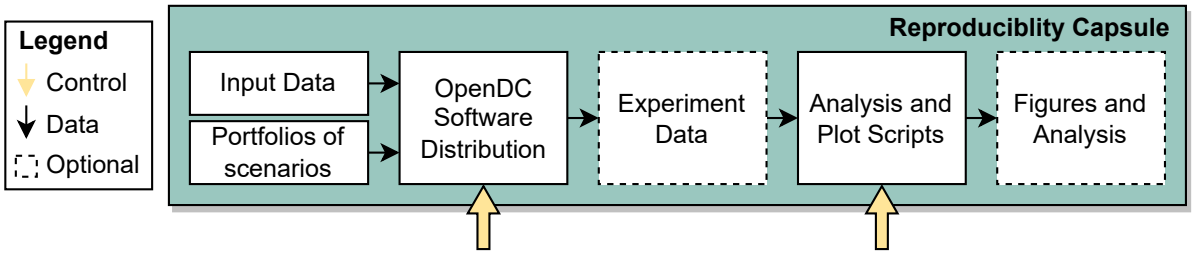


Figure 3.10: Conceptual format of the reproducibility capsule generated by OpenDC. The yellow arrows indicate interaction between the user and the capsule.

OpenDC can automatically package the research material used to conduct the experiments into a shareable *open access artifact* by means of a *reproducibility capsule* (fulfilling **Principle 4**). We depict a conceptual overview of the design in Figure 3.10. The capsule is self-contained; it includes all input artifacts (e.g., workload traces, topologies, parameters) for the experiments, a software distribution of OpenDC, as well as the scripts necessary to reproduce the raw data and conduct analysis of a set of experiments. Optionally, the capsule can include the raw data, so researchers do not have to re-run the experiments to perform additional analysis. In Section 4.2.5, we describe our implementation of the reproducibility capsule in Radice.

Furthermore, **Principle 5** and **Principle 6** of [120] stress the importance of proper statistical analysis of experimental data, including characterizations of the empirical distribution of the measurements. OpenDC assists researchers in achieving these principles, by offering aggregation of metrics, statistical summaries, and automated visualizations of the experimental data.

Finally, through the use of discrete-event simulation, we ensure that experiments with Radice remain affordable for researchers all over the world. We demonstrate in Section 4.7 the costs of our experiments in simulation and compare it to the financial and environmental costs of experiments using real-world, large-scale infrastructure.

3.6. Discussion

We now summarize the contributions of this chapter and discuss potential threats to their validity.

3.6.1. Summary

We propose in this chapter Radice, an instrument for simulation-based risk analysis of sustainable cloud infrastructure, addressing RQ1 and RQ2. Radice is build upon the OpenDC platform, leveraging its existing feature set and extending it in key areas. The explorative capabilities of Radice, combined with support for both long-term and short-term modeling of risk scenarios, enable integral risk analysis of datacenters.

3.6.2. Threats to Validity

We discuss potential threats to internal validity, construct validity, and external validity.

Internal Validity

Aligned with the guidelines for reproducible experiments in cloud computing [120], we design and use simple tests for the validity of our models, primarily focusing on precision (accuracy is guaranteed by the math libraries used in the OpenDC project). However, the precision of our models still could be threatened by the impossibility of comparing directly with large-scale infrastructure. To do so would cost, as we evaluate in Section 4.7, over 190,000,000x more energy (and corresponding monetary value and climate impact).

Construct Validity

The use of simulation in the design of Radice, as opposed to mathematical models or real-world experimentation, might pose a threat to the construct validity. We compare simulation to other approaches and address this threat in depth in Section 2.3.3.

External Validity

Successful application of this system design for risk analysis of cloud datacenters represents the main threat to external validity of the design. In Chapter 4, we show a prototype of this design and demonstrate its risk analysis capabilities. We believe this shows evidence for broader, external applicability of the design.

4

Evaluation of Radice

In this chapter, we address the third research question (RQ3) through a comprehensive evaluation of Radice.

4.1. Overview

We conduct a comprehensive evaluation of Radice, developing a working software prototype and applying four different evaluation methodologies. Overall, our contribution is six-fold:

1. We develop a working software prototype of Radice (Section 4.2), realizing key features of the design. Radice extends the existing OpenDC platform with capabilities for data-driven risk analysis.

2. We engineer Radice and OpenDC to support high-performance simulation capabilities (Section 4.3).

3. We evaluate our prototype of Radice experimentally (Section 4.4), using long-term operational traces collected from private and public cloud datacenter. Our experiments investigate key trade-offs faced commonly by datacenter operators, exploring datacenter sustainability, emerging operational phenomena, alternative topologies, diverse workloads, and scheduler configurations. This work is *the first study to analyze the impact of the recent price surges for electricity and carbon on datacenter operators.*

4. We demonstrate the high-performance simulation capabilities of Radice (Section 4.5), showing that it can provide risk estimates to practitioners in matters of seconds, even in realistic, complex situations, thus enabling the use of this instrument in live discussions.

5. We successfully validate Radice (Section 4.6). Our approach is multi-faceted, employing both manual inspection and comparison against a popular simulator in the field, as well as mathematical analysis.

6. We reflect on the environmental impact of our experiments using Radice (Section 4.7), and contrast it with the impact of equivalent experiments using real-world infrastructure.

We summarize our contributions and discuss their validity in Section 4.8.

4.2. Implementation of a Software Prototype

We describe in this section the implementation of a working software prototype of Radice. First, we explain the software engineering process used to develop the prototype (§4.2.1). We then describe the extensions to OpenDC's simulation model that have been developed to support risk analysis, consisting of a new system for telemetry based on OpenTelemetry (§4.2.2), a new compute scheduler (§4.2.3), and an advanced power modeling subsystem (§4.2.4). Finally, we present the implementation of the reproducibility capsule (§4.2.5) and the implementation of the risk analysis functionality of Radice (§4.2.6). A full overview of the implementation efforts as a result of this thesis is depicted in Figure 4.1.

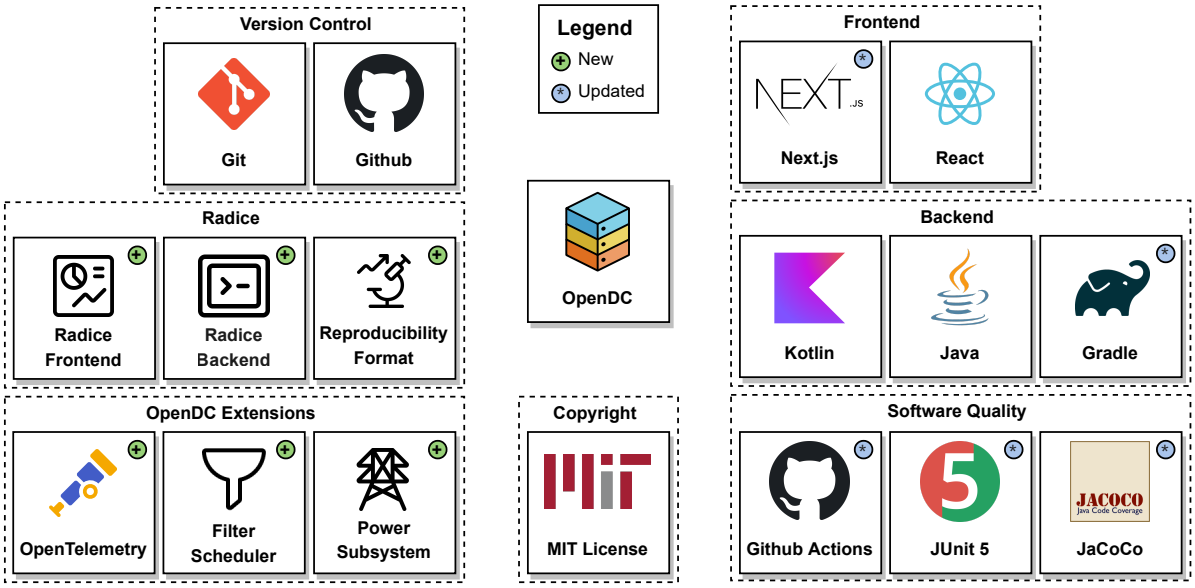


Figure 4.1: Implementation efforts as a result of this thesis.

4.2.1. Software Engineering Process of Radice

We employ industry-best software development practices to build Radice and OpenDC (addressing NFR3). The main codebase is written in Kotlin, a modern and fast-growing programming language that is already adopted by large companies including Google [85]. Kotlin is designed to be fully interoperable with Java, and as such, benefits from integration with the vast Java ecosystem.

We enforce through continuous integration (CI) [53, 61] adherence to high development standards, running for each change an automated test suite to catch regressions, and static code analysis tools (e.g., linting) to spot common mistakes. In addition, we employ a manual-review policy in our version control system, requiring an independent code review before changes to the project can be integrated into the main codebase.

Similar to the Linux project [96], we encourage and oversee the integration of OpenDC extensions into the main codebase. Although this approach increases the overall burden of maintenance, it also ensures high quality and compatibility across all components, as OpenDC evolves.

4.2.2. Telemetry and Metrics

Observability of the system processes is critical for understanding the operation and risks in datacenters. Radice relies extensively on telemetry exposed by OpenDC to assess the risk of a datacenter. The same holds true for operators of cloud datacenters monitoring their internal systems [27, 112].

As part of this work, we adopt OpenTelemetry in OpenDC. OpenTelemetry is an industry initiative supported by the Cloud Native Compute Foundation and drives the development of open standards and tools for instrumenting, collecting, and generating telemetry data (metrics, logs, and traces)[1]. By using OpenTelemetry, we ensure interoperability of OpenDC with the wider ecosystem for observability, and benefit automatically from future community developments (addressing FR6). OpenTelemetry is already adopted by many cloud vendors, and offers extensive tooling for processing the telemetry data produced by its SDKs. By not adopting OpenTelemetry, we risk having to re-implement many of these tools or having to develop separate integrations for each cloud vendor.

Thus, we update all services of OpenDC to expose telemetry data via the OpenTelemetry SDKs. Combined, this amounts to 40+ different metrics that are available by default in OpenDC. In Table 4.1, we list the metrics relevant for this work. Our selection of metrics is representative [125], as demonstrated by the wide range of existing monitoring tools reporting comparable metrics[2], and covers the entire hierarchy of metrics proposed in the taxonomy from SPEC [74], from traditional, low-level performance metrics, to high-level metrics for supporting managerial decisions.

[1] `https://opentelemetry.io`

[2] `https://docs.vmware.com/en/vRealize-Operations/8.6/com.vmware.vcom.metrics.doc/GUID-ACF48F67-B877-45DB-910C-4BFADC86F794.html`

Table 4.1: Metrics exposed by Radice and OpenDC that are relevant to this work.

Name	Unit	Description
`scheduler.attempts`	-	Number of VM scheduling attempts, further subdivided into success, failure, and error categories.
`scheduler.servers[state=pending]`	-	Number of VMs pending to be scheduled by the scheduler.
`scheduler.servers[state=active]`	-	Number of VMs currently active in the system.
`scheduler.hosts[state=up]`	-	Number of physical hosts available to the scheduler.
`scheduler.hosts[state=down]`	-	Number of physical hosts unavailable to the scheduler.
`scheduler.latency`	ms	End-to-end latency for a VM to be scheduled (in multiple attempts).
`system.guests`	-	Number of VMs active on the host.
`system.cpu.limit`	MHz	Total CPU capacity available to a host or VM.
`system.cpu.demand`	MHz	CPU capacity of the host requested to be utilized.
`system.cpu.usage`	MHz	CPU capacity of the host actually utilized.
`system.cpu.utilization`	%	CPU utilization relative to the capacity of a single CPU.
`system.cpu.time[state=active]`	s	Accumulated CPU time spent in a running state.
`system.cpu.time[state=idle]`	s	Accumulated CPU time spent in an idle state.
`system.cpu.time[state=steal]`	s	Accumulated CPU time requested by a VM, but not provided due to CPU contention.
`system.cpu.time[state=lost]`	s	Accumulated CPU time requested by a VM, but not provided due to hypervisor overhead.
`system.time[state=up]`	s	Uptime of the host or VM.
`system.time[state=down]`	s	Downtime of the host or VM.
`system.time.boot`	Epoch (ms)	Boot time of the host or VM.
`system.power.usage`	W	Active power usage of the host.
`system.power.total`	J	Accumulated energy usage of the host.
`customer.availability`	%	Uptime percentage of each VM per aggregation period.
`customer.latency`	s	Starting latency after a VM has been scheduled for start.
`customer.cpu.contention`	%	Average VM CPU contention per aggregation period.
`customer.cpu.interference`	%	Average VM CPU interference per aggregation period.
`company.availability`	%	Uptime percentage of each host per aggregation period.
`company.cpu.saturation`	%	n^{th} percentile of the host CPU utilization per aggregation period.
`company.cpu.imbalance`	-	Standard deviation of the host CPU utilization per agg. period.
`company.power`	kWh	Electricity usage of the datacenter per aggregation period.
`company.co2`	kg	CO_2 emissions of the datacenter per aggregation period.

Table 4.2: Compute filters currently available in OpenDC.

Filter	Predicate
Host Availability	Host is online and ready to accept new VMs.
VM Count	Host contains less VMs than a user-specified threshold.
Available RAM	Host has enough remaining RAM for the VM.
Available vCPUs	Host has enough remaining vCPUs for the VM (considering CPU overcommitment).
VCpu Capacity	Host has enough CPU capacity (in terms of clock frequency) for the VM.

Table 4.3: Compute weighers currently available in OpenDC.

Weigher	Ranking Property
VM Count	Number of VMs.
Available RAM	Available host memory.
Available RAM (per pCPU)	Available host memory divided by the number of physical CPUs.
Available vCPUs	Available vCPUs (considering CPU overcommitment).
VCpu Capacity	Available CPU capacity.

4.2.3. Compute Scheduler

The scheduler is a critical part of live infrastructure management and is responsible for placing VMs onto physical machines. We implement in OpenDC the filter scheduler as described in Section 3.5.2. Our implementation supports many of the filters and weighers already available in the OpenStack project[3]. Tables 4.2 and 4.3 list the filters and weighers currently implemented in OpenDC.

Although our approach differs from the scheduling techniques common in literature, through the combination of different filters and weighers, we can re-create many of the traditional VM allocation policies described in literature [149]. For instance, round-robing scheduling can be implemented by ordering hosts based on the number of VMs allocated to them. Implementing a random scheduling policy is achieved by disabling all weighers and setting the subset size to infinite (see Section 3.5.2 for more detail).

4.2.4. Advanced Power Modeling Subsystem

During the development of Radice, the author of this thesis has supervised an external project by an Honors B.Sc. student to extend OpenDC with an advanced system for energy modeling and power management.

To be able to model the complex power distribution systems that exist in cloud datacenters, the student implemented in OpenDC various power devices that are commonly present in datacenters, such as automatic transfer switches and power distribution units. The implementation also models electrical losses as a result of the circuitry in these devices. Furthermore, the student added support for eight different CPU power models commonly used by the community, dynamic frequency and voltage scaling (DVFS) through P-states, and four configurable scaling governors in the hypervisor derived from the Linux kernel. We describe in detail the design of the energy modeling and power management system in Section 3.5.3.

To obtain accurate power consumption estimations, we adopt in this work an interpolation-based CPU power model implemented by the student. This model interpolates power consumption based on real-world results from the *SPECpower_ssj2008* benchmark published by hardware vendors[4]. These results have been validated by SPEC and include measurements of the active power of systems for various system loads. In our experiments, we have mapped machines in the topology to representative hardware models for which benchmark results have been published by hardware vendors.

4.2.5. Reproducibility Capsule

We implement in our prototype the *reproducibility capsule* as described in Section 3.5.5, addressing NFR2. Our tooling can automatically generate a compressed archive (in ZIP or TGZ format) that includes the input traces, portfolios of scenarios, a software distribution of OpenDC, and the scripts for generating the raw experiment data and for analysis of the results. By automating this process, we ensure that the archive is correct and that we can reproduce the archive for arbitrary versions. The Jupyter notebooks included in the reproducibility capsule combine documentation and scripts, and provide straightforward analysis of results.

4.2.6. Radice Functionality

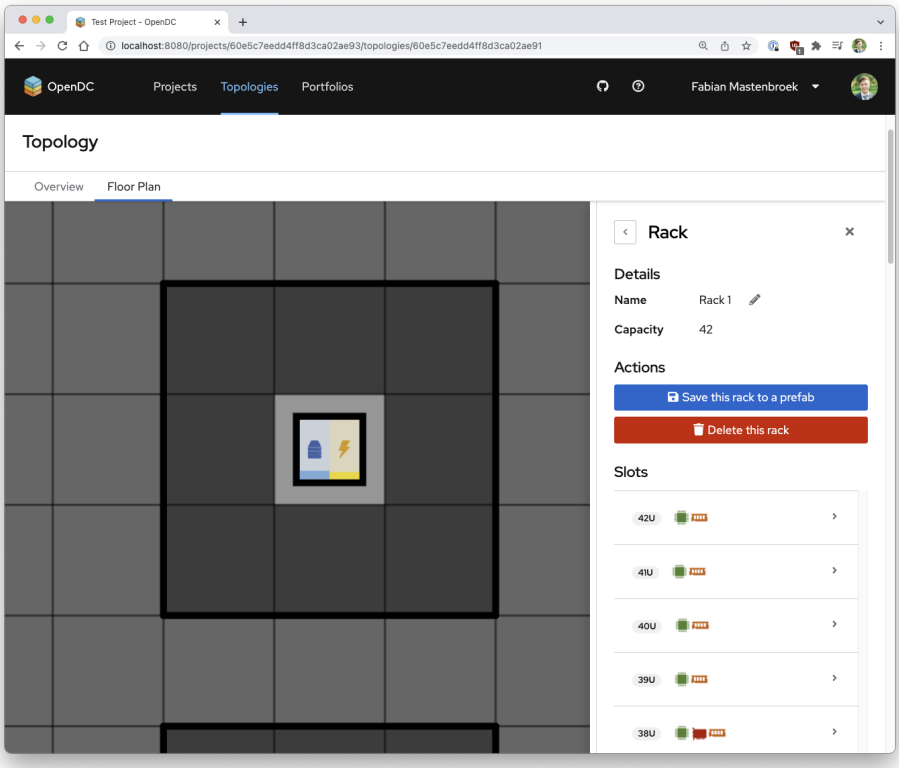
We describe in this section the extensions to OpenDC specifically built to support the functionality of Radice.

Extending the Backend

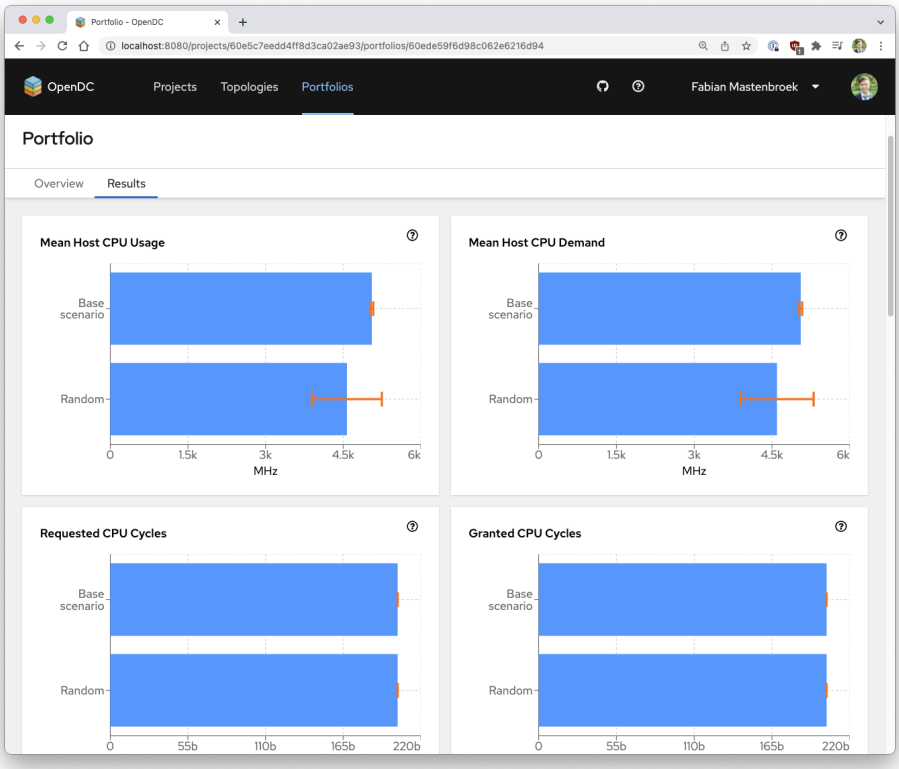
We update the communication protocol used by the *experiment runners*. With previous versions of OpenDC, experiment runners have to connect directly to the database to retrieve scenarios to simulate. Such an approach is not secure (since it grants access to all user-data) and makes it difficult to support user-provided experiment runners. We therefore extend the Python web server with a new communication API for experiment runners. This API allows experiment runners to periodically poll for new scenarios, launching simulations as they arrive, and reporting the results back to the API server. These efforts also include updating the OpenAPI[5] specification of our API, which formally specifies the key communication protocols used in OpenDC.

Furthermore, we redesign the results processing pipeline used by OpenDC. The original prototype of Capelin [12] emits its periodic measurements to Parquet files and uses a Spark big data pipeline to aggregate results into a per-scenario overview. Our new implementation greatly simplifies the results processing pipeline, by utilizing Radice to aggregate measurements of active experiments in memory.

[3] `https://docs.openstack.org/nova/latest/admin/scheduling.html`
[4] `https://www.spec.org/power_ssj2008/results/power_ssj2008.html`
[5] `https://swagger.io/specification/`



(a) Users of OpenDC can interactively design a datacenter.



(b) OpenDC automatically generates visual summaries of the experiment results.

Figure 4.2: Impression of the web interface of OpenDC including the extensions contributed in this work.

Extending the Frontend
We improve the user interface of OpenDC by adopting the PatternFly v4 design framework[6]. By using an existing design framework, we can re-use high-quality components designed by experienced engineers, and increase accessibility of the interface. This framework is used in products such as OpenShift and oVirt, which closely match the requirements of OpenDC. In Figure 4.2, we depict an impression of the new web interface provided by OpenDC.

Furthermore, we decouple the authentication system in OpenDC from Google. Previous versions of OpenDC required users to have a Google account to be able to sign in into the system. Experiences with using OpenDC in an educative setting showed that this requirement frequently caused problems, with students not having a Google account, or not able to create one. We address this issue by updating the authentication system in OpenDC to use OpenID Connect[7], which is an open standard for authentication that integrates with existing identity providers, including Google and Microsoft.

Adding a Library of Components and Experiments
In addition to the supervision efforts in Section 4.2.4, the author of this thesis has also supervised an external project to develop a web store for OpenDC where users can exchange datacenter designs and prefabricated components, as well as sharing experimental results with other users.

4.3. Performance Engineering

Performance is a key requirement of Radice (NFR1). We employ various profiling tools offered by the Java ecosystem (such as VisualVM and the Java Microbenchmark Harness) to identify bottlenecks and analyze potential optimizations. The efforts as part of this work have led to the identification of three major system bottlenecks, and, in turn, to significant performance improvements to OpenDC as a result of addressing them. In Section 4.5, we demonstrate the impact of these changes on the performance of OpenDC.

Our initial analysis revealed that the low-level resource models were the main limiting factor of performance in OpenDC. These resource models enable workloads to characterize how they utilize resources such as CPU, memory, or network. In particular, the *hypervisor scheduler*, which ensures that the limited capacity of CPUs is shared between multiple workloads, has a significant impact on performance. We find that the old implementation required multiple memory allocations in the hot path, as well as unnecessarily repeating the same computations. Since the identified issues are mostly inherent to the architecture of the original implementation, we design a new module in OpenDC called the *flow engine*. The flow engine underpins all resource models in OpenDC (e.g., compute, memory, or network) and models the interaction between resources and resource consumers as a flow network. In a flow network, sources (e.g., workloads) attempt to push flow at a certain rate to sinks (e.g., CPUs) based on demand and capacity. The flow rate is generic and could represent the current speed of the CPU, the utilization of a disk, or the throughput of a network interface card. A flow network, in contrast to our previous approach, only needs to be re-computed if the flow rate changes. Our implementation of the flow engine minimizes computation and eliminates all memory allocations in the hot path. During every update, the flow engine will only visit nodes in the network whose state is invalidated. We use the new flow engine to implement an efficient hypervisor scheduler, which employs a new max-min fair sharing algorithm to divide the available CPU resources fairly across workloads.

Furthermore, we discovered during experimentation that the original design of the workload interference algorithm had a non-negligible impact on memory consumption of the simulator, severely restricting the number of simulations we could run in parallel. This algorithm needs to track whether two or more VMs are located on the same host and determine whether they interfere. However, the previous implementation essentially cloned the model data for every physical host in the simulation, amounting to gigabytes of memory in total. We have redesigned the algorithm to share model data across physical hosts and reduce the amount of per-host state necessary. Our new implementation requires slightly more computation than before. However, according to our benchmarks, this does not significantly affect the overall runtime performance.

Finally, we have focussed on optimizing the workload trace library of OpenDC. This library is responsible for parsing, processing, and exporting workload traces in a wide variety of workload formats. Our initial implementation suffered from a lot of unnecessary memory allocations, caused by Java's auto-boxing functionality and simplistic parser implementations. We have updated the API exposed by the library, as well as the underlying implementations, to use streaming parsers and eliminate unnecessary memory allocations.

[6] `https://www.patternfly.org/v4/`
[7] `https://openid.net/connect`

Table 4.4: Experiment configurations explored in this work.

Experiment	Focus of experiment	Factors	Optimization
§4.4.7	Datacenter sustainability	Energy and CO_2 prices, PUE	✓
§4.4.8	Operational phenomena	Availability target	✗
§4.4.9	Datacenter topology	Topology	✓
§4.4.10	Workload	Workload trace	✗
§4.4.11	Datacenter scheduler	Allocation policy	✓

4.4. Experiments with Radice

How to conduct a comprehensive evaluation of a system for risk analysis of datacenters? There exist in the field already various methodologies for assessing system design. In this work, we use in-depth experimental analysis of a prototype to demonstrate Radice's capabilities. The design of such an evaluation consists of multiple aspects; a typical experiment in the field must consider at least the workload, the environment, and the key performance indicators to measure [120]. We explain each of these aspects in Chapters 1 and 2.

In our experiment design, which is summarized in Table 4.4, we cover these aspects in detail, addressing key questions such as: What workload is modeled (§4.4.2)? What is the datacenter topology (§4.4.3)? How is the scheduler configured (§4.2.3)? What operational phenomena are taken into account (§4.4.4)? Which risk factors are considered (§4.4.5)? Our experiments explore a selection of risk scenarios that appear frequently in datacenters, highlighting the key trade-offs between different risk factors that datacenter operators face.

4.4.1. Execution and Evaluation

Reproducibility is a key principle of the scientific process. This principle is embedded in core parts of Radice's system design (NFR2), as we explain in Section 3.5.5. Reproducibility of experiments requires a detailed description of the methods, such that externals are able to reproduce the results, ideally with exactly the same outcomes. The results produced by Radice are fully reproducible, regardless of the underlying platform on which the experiments are run. Radice and OpenDC have been designed explicitly so that each source of non-determinism is seeded by the current repetition, therefore ensuring reproducibility of our measurements.

Reproducible experimentation also necessitates consideration for variability of measurements [120]. Despite this, there is no accepted threshold for confidence intervals in the field; worse, most studies do not even compute or report them [120]. Instead, many studies repeat experiments a number of times; recent work shows that a few tens of repetitions are desirable for datacenters [147]. In our work, we run a much larger number of repetitions, to improve statistical confidence and precision of the results. We picked a round number of this magnitude, 4,096, given that with the performance improvements in OpenDC, it is possible to run so many repetitions without considerable delay. The individuals discovered by the risk optimization algorithm are evaluated 32 times, with the best performing individuals again being simulated with 4,096 repetitions. We have inspected the results of our experiments and found that the error observed in our results is small and does not affect our conclusions.

It is feasible for other scientists to reproduce our experiments without significant computational requirements. A single experiment run, e.g., simulating three months of datacenter operation, takes approximately three seconds on modern hardware. The full set of experiments is conveniently parallel and requires around five hours to complete on a "beefy" but standard 64-core machine (costing in total €4–€20 in the cloud); parallelization across multiple machines could reduce this to a mere hour or even minutes. We elaborate on the performance of Radice and OpenDC in detail in Section 4.5.

4.4.2. Workload

Key to proper experiment design in this field is the selection of workloads [120]. We experiment in this work with four different long-term, real-world workload traces, originating from both public and private cloud environments. These traces characterize the operation of several workloads through a set of VM-level metrics aggregated over 5-minute-intervals. A summary of these workload traces is provided in Table 4.5.

We focus in this work on a business-critical workload trace from Solvinity, a *private cloud provider* [150] operating mainly in the Dutch ICT market, which we refer to as *baseline*. Such a workload is representative for the Dutch economy, which directly depends for 33% on digital services hosted in datacenters, the majority of which are business-critical [122]. An anonymized version of this trace has been published in the Grid

Table 4.5: Characteristics of the workloads used in this thesis.

Workload	VM submissions/h		VM duration [days]		CPU Usage [MHz]		Memory Usage [GB]	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ
Baseline	0.919	24.286	44.702	21.171	542.068	2,295.732	7.552	10.840
Bitbrains	1.857	44.677	28.703	5.164	1,811.749	5,080.531	11.755	32.602
Materna	0.337	11.970	91.734	13.704	307.442	724.981	13.624	14.742
Azure	2.544	3.958	1.753	6.247	486.625	848.522	5.804	10.169

Workloads Archive [78]. The full trace spans a period of three months of datacenter operation and consists of 1,800 VMs running business-critical applications, collectively consuming 3,063 PFLOPs (*exascale*), with a mean CPU utilization of 5.6% on the original topology. This low utilization is in line with industry, where utilization levels below 15% are common [89, 134, 151], as to ensure the datacenter has enough spare capacity in the presence of failures, and to reduce the risk of workload interference.

We experiment with different workloads, in Section 4.4.10, to analyze the impact of workload on risk in datacenters. We select two other business-critical traces published in the Grid Workloads Archive [78], Bitbrains [135] and Materna [94, 95], which are both similar in scale and structure. Bitbrains is an earlier workload trace published by Solvinity spanning one month of operation across 1,250 VMs, while the trace of the German service provider Materna consists of 547 VMs stretching over a period of three months.

Furthermore, we also consider a public cloud trace from Azure [46], to contrast it against private cloud workloads. This is a recent operational trace published by the prominent cloud provider Microsoft only a few years ago and increasingly used in the community. We use the most recent release of the trace. Unlike the other traces, the Azure trace does not express CPU usage in terms of frequency (MHz), instead reporting it as a utilization metric ranging from 0 to 1, due to anonymity. For this work, we assume that the original workload ran on Dv4-series instances, with a base clock of 2.5 GHz, and we scale each utilization measurement by this value accordingly. Moreover, the complete Azure trace contains over 2 million VMs. To provide a realistic comparison, we randomly sample approximately 1,800 VMs using the trace sampling tools offered by OpenDC, matching the size of the baseline workload. We do not scale the other two traces, as they are similarly sized to the baseline workload.

4.4.3. Datacenter topology

For all experiments, we consider the topology that ran Solvinity's original workload as the *baseline* topology. This topology is very common in the industry and represents a subset of Solvinity's complete infrastructure when the full trace (see Section 4.4.2) was collected. It consists of 12 compute clusters of approximately 200 physical hosts in total, spread over three datacenters. These datacenters are connected through a fiber optic ring network, while the servers in the clusters communicate via a low-latency InfiniBand network.

There are two sets of clusters in the topology: *standard* clusters and *bigmem* clusters. Standard clusters contain 16 servers, each fitted with two 8-core CPUs and 128 GB of memory, while bigmem clusters contain 6 servers equipped each with four 8-core CPUs and 768 GB of memory. The servers do not carry local persistent storage and are instead attached to a reliable storage area network (SAN), consisting of six devices of 10 TB each, providing up to 60 TB of storage. Due to confidentiality of operational details, we cannot share more detailed information about the topology, such as the exact hardware models used in the datacenter.

The datacenters in this topology are assumed to have a PUE of 1.57. This value is equivalent to the global average in 2021 [29]. Although hyperscale datacenters report substantially lower PUE values (ranging 1.1–1.4), we focus in this work mainly on *mid-tier providers* of cloud infrastructure where such values are not yet common. The CO_2 emission factor (carbon intensity) for these datacenters, which describes the amount of CO_2 emissions produced per unit of electricity consumed, is assumed to be 556 kg CO_2 per MWh of electricity consumed. This assumption is based on the estimated CO_2 emissions in the entire supply chain for the consumption of gray energy sources in the Netherlands [138, 159]. Although datacenters are increasingly purchasing electricity from renewable sources, due to the intermittent nature of many of these renewable energy sources [115], electricity usage in these datacenters might still result in additional CO_2 emissions.

We also explore in Section 4.4.9 variations of this topology stemming from manual modification and automated optimization. We consider two types of alterations in particular: (1) *scaling* the topology, by increasing or decreasing the number of servers in each of the clusters; and (2) *upgrading* the topology, by replacing the hardware in the clusters with newer, faster, or different hardware models.

Table 4.6: Parameters for the lognormal failure model we use in our experiments. We use the normal logarithm of each value.

Parameter [Unit]	Scale	Shape
Inter-arrival time [hour]	24×7	2.801
Duration [minute]	60	60×8
Group size [machine-count]	2	1

4.4.4. Operational Phenomena

Operational phenomena appear frequently in large datacenters (see Section 2.2.3). We consider in our experiment design two particular types of operational phenomena: (1) performance variability due to workload interference and (2) correlated cluster failures.

We assume a common model for workload interference [93, 150], where a set of collocated workloads is assigned a *score* from 0 to 1, with 0 indicating full interference between VMs contending for the same physical CPU, and 1 indicating non-interfering VMs, at a given CPU load level. The performance score is derived from *CPU Ready* (also called *CPU Steal* time) values of VMs, which represent the percentage of time a VM is ready to run, but has to wait while the hypervisor services other VMs. We construct our model using the placement data of all VMs in the baseline workload running on the baseline topology, and collect the set of collocated workloads along with their mean performance score, defined as the mean CPU ready time fraction subtracted from 1, conditioned by the CPU utilization of the host at that time, rounded to one decimal. During simulation, this score is then activated if a VM is collocated with at least one other in the recorded set and the system utilization is at least equal to the recorded utilization. The score is then applied randomly to each collocated VM with probability $\frac{1}{N}$, where N is the number of VMs in the collocation set, by multiplying its CPU demand with the score, thus granting it this (potentially lower) amount of CPU time.

The second phenomenon we model are cluster failures. We assume for this a common model for space-correlated failures [62], in which a failure might trigger more failures within a short time span; these failures constitute a *group*. We limit this phenomenon to hardware failures that crash machines (full-stop failures), with subsequent recovery after some duration. Other types of failures, such as software failures, network failures, or Byzantine failures, are not considered for this work. Space-correlated failures in large-scale distributed systems have been extensively studied in literature; Gallet et al. report common values and derive fitting statistical distributions [62, 83]. Therefore, in this work, we use a lognormal model with parameters for failure inter-arrival time, group size, and duration, as listed in Table 4.6. The choice of parameter values is inspired by GRID'5000 [62] (public trace also available [83]) and Microsoft Philly [84], scaled to the size of the baseline topology. The failure duration is restricted by a minimum of 15 minutes, since faster recoveries and reboots at the physical level are rare. Note that these are pessimistic estimates of failure frequency and time to recovery. Actual parameters will depend on the choice of hardware and configuration of the infrastructure.

4.4.5. Risk Model

In Section 3.5.1, we describe our approach for quantifying risk factors in datacenters. We consider for our experiments eight different risk factors present in datacenters and summarize them in Table 4.7. Our selection covers risks that impact customers, risks related to operational sustainability, and risks facing society.

Table 4.7: Risk factors considered in this work.

Risk Factor	Objective	Aggregation Period	Impact
Availability (Uptime)	≥ 99.5%	1 month	See Table 4.8a
Scalability (Scheduling Latency)	< 1 hour	1 month	
Quality of Service (VM Interference)	≥ 97.5%	1 day	See Table 4.8b
Electricity	-	1 month	€300 per MWh (see Figure 4.3)
CO_2 Emissions (Company)	-	1 month	€64 per tCO_2 (see Figure 4.3)
Resource Saturation	< 75%	1 day	€125 per incident
Resource Imbalance	< 0.2	1 day	€125 per incident
CO_2 Emissions (Society)	-	1 month	€360 per tCO_2 [129]

Table 4.8: Refund policy based on SLAs common in the industry. We assume customers are charged $0.046 per vCPU-hour (€0.040 as of November 2021), based on the cost of a *m6g.medium* instance in the Frankfurt region of AWS.

(a) Availability SLA

Monthly Uptime [%]	Refund
< 95%	100%
< 99%	30%
< 99.5%	10%

(b) Quality of Service SLA

Quality of Service [%]	Refund
< 90%	100%
< 95%	30%
< 97.5%	10%

Customer-facing risks

The key performance indicator customers of datacenter operators are concerned with is availability. We configure Radice to ensure a monthly uptime target of 99.5% for individual VMs. Customers with VMs that fail to reach this uptime target are refunded according to refund policy listed in Table 4.8a. This kind of SLA is common for cloud providers, such as Amazon Web Services[8], Google Cloud[9], and Microsoft Azure[10].

Furthermore, we ensure that customer requests are served promptly (*scalability*). If a VM is not started within an hour of its submission time, the customer is refunded all expenses for that VM while it was pending to be scheduled. It is important that we add such a constraint, as otherwise, not scheduling a VM at all incurs a lower risk than scheduling the VM in the presence of failures.

Finally, we also consider Quality of Service (QoS) as a customer-facing risk. It is rare for cloud providers to offer performance guarantees to customers [111], with Oracle Cloud being the only major cloud vendor to provide performance commitments to customers[11], given that they depend very much on customer behavior and that of other tenants (performance variability, which is well-known in clouds [147]). Nevertheless, datacenter operators monitor performance metrics carefully, since low quality of service can lead to reputational damage or loss of customers. To quantify the impact of performance variability due to workload interference, we monitor CPU contention and interference metrics as a proxy for QoS, and refund customers with VMs with average CPU interference values above 2.5% per day (equivalent to a QoS value lower than 97.5%), according to the refund policy in Table 4.8b. This policy is derived from existing cloud SLAs and uses commonly accepted thresholds in the industry.

Company-facing risks

To model the operational sustainability of the datacenter operator, we consider resource utilization metrics for hosts. Usually, a high resource utilization indicates an issue with the infrastructure and demands an investigation from an engineer (*resource saturation*). A large imbalance in utilization between the hosts might also indicate an issue with the scheduler and can trigger an investigation as well. We configure Radice to ensure the 95th percentile of the host utilization does not exceed 75%, and that the resource imbalance (defined as the standard deviation of the host utilization) does not exceed 0.2. Such values are commonly used in datacenter monitoring systems to alert operators for issues. We assign an impact of €125 to an investigation. This is an optimistic estimation of the costs of an engineer needing to investigate an issue with the infrastructure and taking a mere two hours to resolve the issue.

Furthermore, we consider the expenses for electricity usage and CO_2 emissions incurred by the datacenter operator. We assume the cost of electricity is €300 per MWh, based on public data from APX for October 2021 (depicted in Figure 4.3). The cost of CO_2 emissions is assumed to be €64 per tonne of CO_2, based on public data from ICE for October 2021 (also depicted in Figure 4.3).

Society-facing risks

We include in our model also risks that society is confronted with as a consequence of datacenter operation. We use the social cost of carbon (SCC) to model the risk of CO_2 emissions as a result of electricity usage by datacenters. Ricke et al. estimate the global social cost of CO_2 to be $417 per tonne of CO_2 [129] (€360 per tCO_2 as of October 2021).

[8] `https://aws.amazon.com/compute/sla/`
[9] `https://cloud.google.com/compute/sla`
[10] `https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/`
[11] `https://www.oracle.com/en/cloud/sla/`

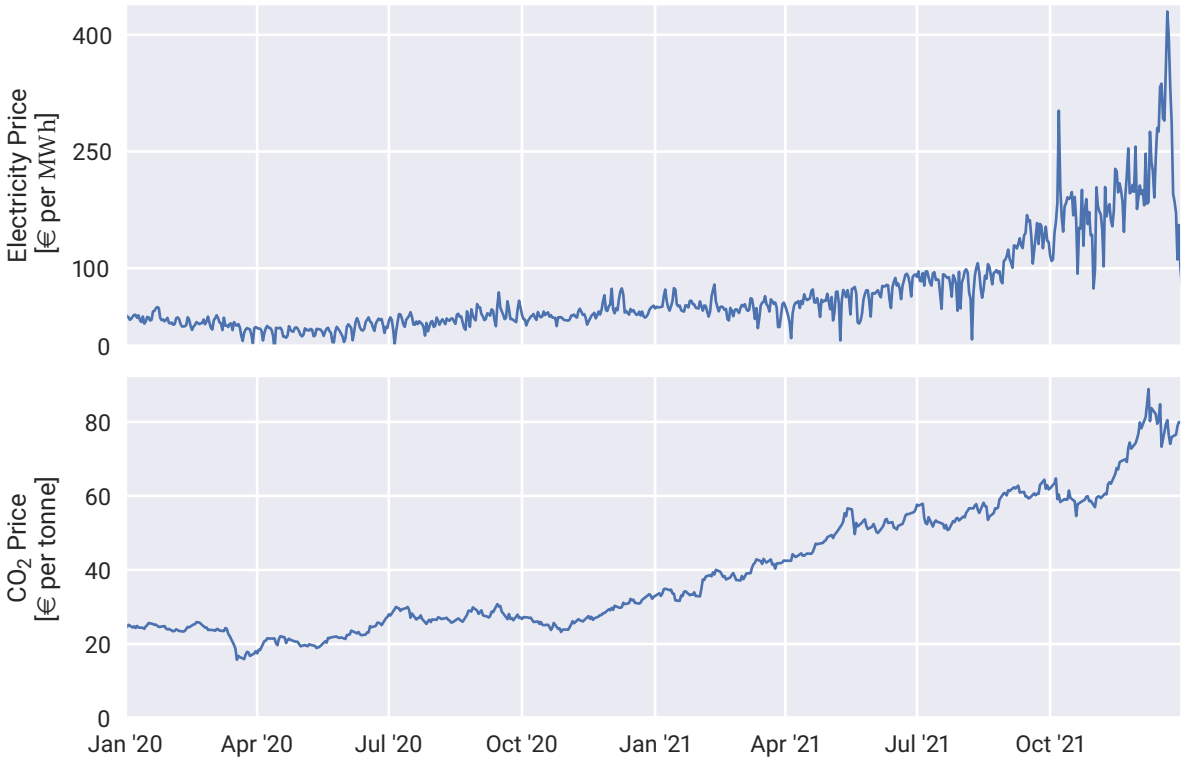


Figure 4.3: Day-ahead electricity prices in the APX Power Spot Exchange from 2020 to 2021 in € per MWh (Source: APX) and daily close prices for European Union Emissions Trading System (ETS) Dec21 Futures from 2020 to 2021 in € per tonne CO_2 (Source: ICE).

Table 4.9: Configuration of the genetic algorithm used by Radice.

Parameter	Value
Population Size	30 individuals
Stop Criteria	10 steady generations or at most 50 generations
Selection	Tournament selection [109]
Genetic Operators	Uniform crossover (probability 0.2), uniform mutation (probability 0.15), Gaussian mutation (probability 0.10)

4.4.6. Genetic Algorithm

We describe in Section 3.5.4 the design of a risk optimization algorithm in Radice using genetic search. We implement this genetic search algorithm using Jenetics [160], a Java library for constructing genetic algorithms. In Table 4.9, we summarize the configuration parameters used for the genetic search algorithm.

Radice limits the maximum number of generations to 100 by default. Furthermore, it also terminates the evolution process when the average fitness of the last 10 generations differs at most 0.01% from the average fitness of the last 30 generations, in which case we consider the fitness to be converged.

We use tournament selection [109] to select individuals for the next generation, which chooses the best individual from a random sample of five individuals (drawn with replacement) from the population. It is a good choice due to its lack of stochastic noise and independence to scaling of the fitness function [30].

We employ three types of genetic operators in our algorithm. *Uniform crossover* is used to combine the genetic information of two parents into a new offspring, by swapping genes at index i of two chromosomes with probability 0.2. Empirical studies show that this approach leads to better exploration of the design space while maintaining the exchange of good information [41]. *Uniform mutation* is used to maintain genetic diversity in the population, and changes a gene's value to a uniform random value with probability 0.15. Finally, we use *Gaussian mutation* to mutate a gene with probability 0.10 to a new value is picked based on a Gaussian distribution around the current value of the parameter, allowing for exploration of better solutions near the current individual.

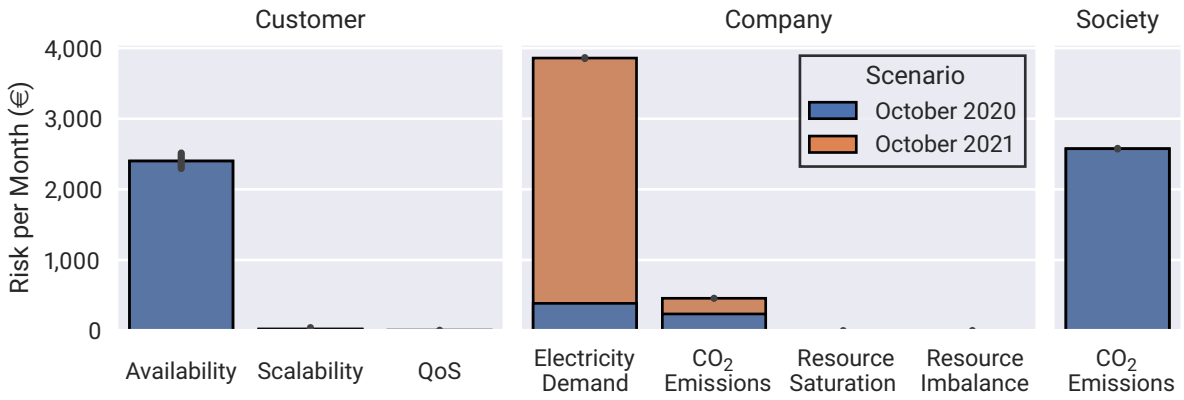


Figure 4.4: Risk profile of October 2021 and October 2021 for the *baseline* workload generated by Radice.

4.4.7. Analyzing the Sustainability of Datacenters

Our main findings from this experiment are:

MF1 Radice enables data-driven risk analysis of cloud infrastructure.

MF2 Electricity expenses have become the primary risk in 2021 due to increasing electricity prices. Hyperscale infrastructure operators could face billions of dollars in yearly electricity expenses.

MF3 Energy-inefficient facilities are impacted by price increases by a factor 1.8x–2.2x more compared to hyperscale infrastructure operators.

MF4 The societal impact of the CO_2 emissions is one of the highest risks in 2021, yet only a fraction of that risk is currently carried directly by the company or customer.

This experiment investigates the sustainability of datacenters. We consider two different perspectives, that of the company having to sustain operation (*operational sustainability*), and that of the environment having to cope with datacenter operation (*environmental sustainability*). These perspectives represent an increasingly relevant trade-off for infrastructure operators, due to the large environmental footprint of datacenters combined with the increased efforts by governments and legislators to penalize polluters.

We focus first on the price spikes for electricity and CO_2 bonds that occurred in 2021 (see Figure 4.3). Although datacenter operators often enter long-term power purchase agreements (PPAs) to ensure sufficient power capacity at a stable price, new datacenter projects or operators looking to expand existing capacity face increased operational expenses, which may threaten their financial viability. In Figure 4.4, we depict the risk profile reported by Radice for October 2021, compared to the risk profile of the year before (October 2020). We observe that in 2020, the primary factors contributing to the risk profile of the datacenter are availability, covering the costs of potential failures in the datacenter, and CO_2 emissions for society. We investigate the former risk factor in Section 4.4.8, while the latter risk factor is covered later in this section. By contrast, in 2021, the prices for electricity and CO_2 emission bonds have increased significantly, leading to electricity expenses becoming the primary risk factor for datacenters in Figure 4.4. The volume or price of CO_2 emissions caused by datacenters is still too low to have a significant impact on the risk profile of datacenters, though. Whereas the risk of availability is highly variable and has a much lower median cost (as we show in Section 4.4.8), the electricity demand and CO_2 emissions of datacenters is relatively stable, leading almost certainly to higher expenses and making this such a serious problem.

These price increases can have even more devastating effects for operators of energy-inefficiency datacenters. In Figure 4.5, we illustrate increased monthly costs that datacenter operators face compared to 2020. We take into account multiple price-surge scenarios, as well as the energy-efficiency of the datacenter, in terms of PUE. On average, datacenters operate at a PUE of 1.57, while older, less energy-efficient facilities may reach a PUE of 2.5 (equivalent to the average PUE in 2007) or worse. By contrast, hyperscale datacenters, such as those from Google and Facebook, manage to reach considerably lower PUE values of 1.1 or below[12].

[12] `https://www.google.com/about/datacenters/efficiency`

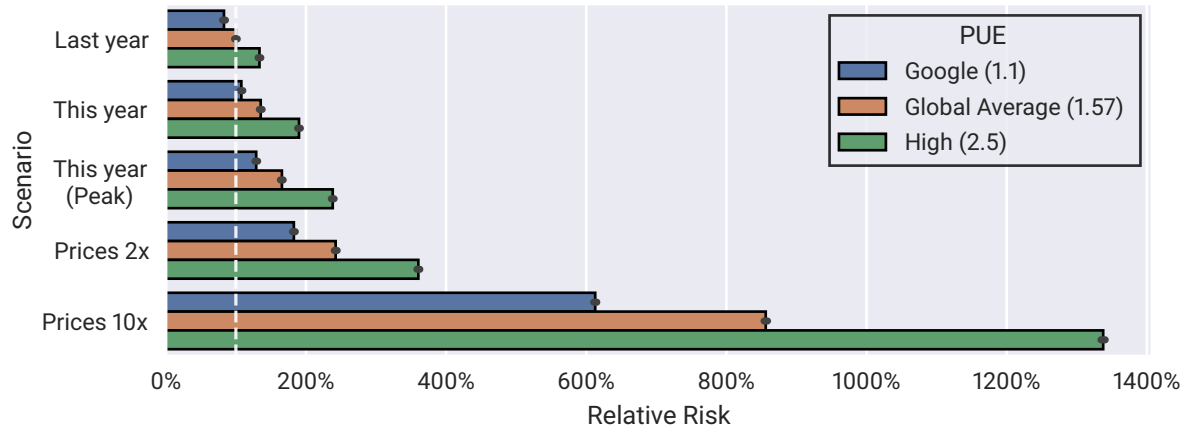


Figure 4.5: Relative risk for the baseline workload for different price-surge scenarios, compared to 2020. The dashed vertical line represents 100% or the situation of 2020 for a datacenter with an average energy-efficiency.

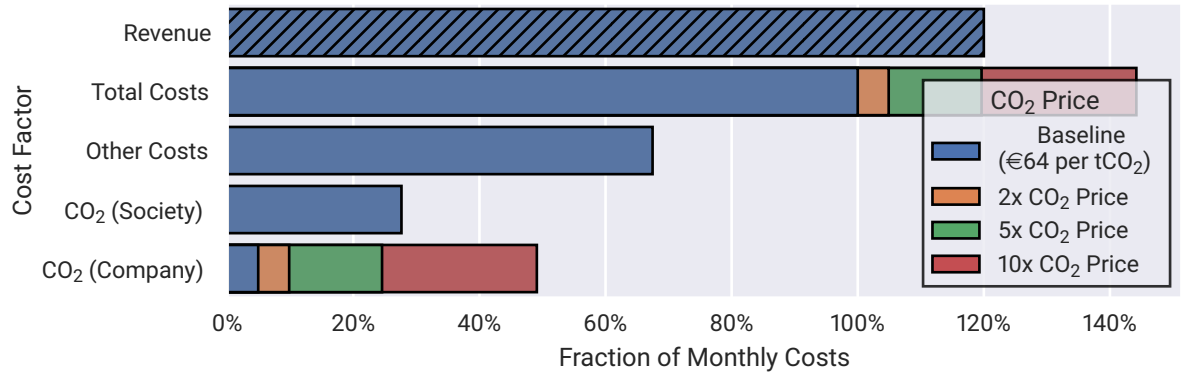


Figure 4.6: CO_2 costs relative to the other costs incurred by the datacenter operator per month.

Unsurprisingly, we find that datacenters with a high PUE are affected the most by the increasing prices, by a factor 1.8x–2.2x higher compared to hyperscale facilities. An optimistic view of this is that these datacenters still have plenty of opportunities left to improve their energy-efficiency, whereas hyperscale facilities will likely have caught all the low-hanging fruit already. Nevertheless, the price increases for electricity and CO_2 bonds pose a real threat to datacenters. Were the prices to increase by another factor 10x as seen in 2021, monthly expenses of datacenter operators could rise by a factor 60x–140x. In that scenario, operators of hyperscale infrastructure (assuming 2+ million machines) face a bill of over 30 billion dollars per year just for electricity, twice the total revenue of Google Cloud (!)[13].

Finally, we investigate the discrepancy between the price paid by datacenters operators for emitting CO_2, compared to the overall costs incurred by society due to CO_2 emissions, also called the *social cost of carbon*. The European Union Emissions Trading System (ETS) was the first large-scale trading scheme for greenhouse gas emissions and requires installations to obtain bonds (also called allowances) to cover their emissions, with each bond permitting the emission of 1 tonne of CO_2. Although prices for these bonds reached new heights in 2021, as shown in Figure 4.3, the baseline price of €64 is just a fraction of the overall costs incurred by society. This social cost is estimated to be substantially higher, with a median of $417 per tCO_2 and 66% confidence interval of $177–$805 per tCO_2 [129]. In Figure 4.6, we depict the effect of adjusting the cost of CO_2 on monthly costs incurred by a datacenter operator, where we assume that the risk factors reported by Radice encompass all costs of the operator, and that the operator runs at an operating margin of 20%. Surprisingly, we find that increasing the CO_2 prices by 5x, to roughly match the social cost of CO_2, consumes the entirety of the datacenter operator's operating margin of 20%. Thus, further price increases of CO_2 bonds could threaten the operational sustainability of datacenters, especially when legislators decide to narrow down the discrepancy between the bond costs and the societal costs.

[13] `https://abc.xyz/investor/static/pdf/2020Q4_alphabet_earnings_release.pdf`

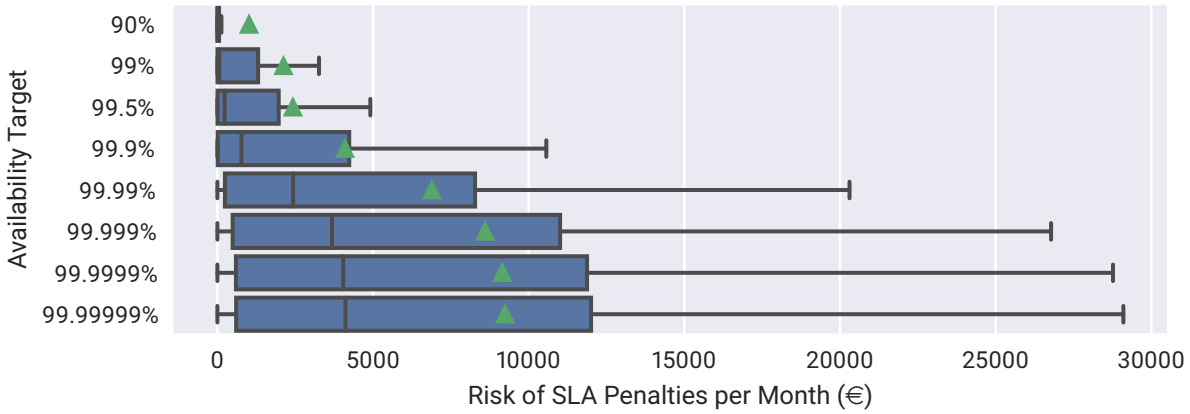


Figure 4.7: Impact of availability target on incurred SLA penalties. 99.5% is the baseline availability target.

4.4.8. Impact of Operational Phenomena

Our main findings from this experiment are:

MF5 The risk of violating the availability SLA has a high variability. In most cases, the monthly incurred cost is less than €250, whereas in the worst case, the operator could incur €100,000 in penalties as a result of SLA violations.

MF6 *Free Nines* — The risk of SLA violation stops increasing as the guaranteed number of nines is increased, given that almost any failure event will cause availability to dip below the target.

MF7 CPU interference is a low risk in our experiments, due to low resource utilization.

This experiment investigates the impact of operational phenomena on the risks faced by datacenter operators. We explore the impact of host failures and the strictness of customer SLAs on the incurred SLA penalties. Figure 4.7 depicts the SLA penalties incurred by the datacenter operator per month, for different VM availability targets. For the baseline scenario, we find that the availability risk is highly variable, since depending on how long the downtime is and which machine is affected, the cost can vary significantly. To illustrate, the average cost for availability SLA violations is €2,400 per month, while in most cases the company incurs a cost of less than €250 per month, yet in the worst case, the company faces €100,000 in penalties.

Furthermore, we find that the cost of going from our baseline availability target of 99.5% to 99.9% is significantly higher than going from 99.0% to 99.5%. The difference going from 99.9% to 99.99% is even more pronounced. This suggests that many unavailability events fall outside the 99.5% target, but cause SLA violations for higher availability targets. In turn, it also highlights the difficulty and risk of guaranteeing higher availability of individual VMs to customers. This is also what we observe in the industry: cloud providers provide higher guarantees (99.99%+) for services using redundant infrastructure, but are only willing to guarantee an availability of 99.5% for individual VMs.

Furthermore, we observe that as the number of nines is increased further, the incurred SLA penalty stops increasing. This behavior is especially noticeable with the last two availability targets, where the difference in SLA penalty between both targets has become negligible. However, this is expected because for these strict availability targets, almost any failure event will cause the SLA to be violated. For reference, six nines (99.9999%) permits just 2.63 seconds of downtime each month.

We also investigate the effects of operational phenomena on quality of service or overall performance. Figure 4.8 depicts the contended CPU time of all VMs in the baseline workload in the presence of different operational phenomena. The contented CPU time metric represents the amount of time that a VM was ready to run, but was waiting while other workloads were serviced by the hypervisor, and gives an impression of the quality of service experienced by VMs. We observe that workload interference has a significant impact on CPU contention metrics, representing 87% of the recorded CPU contention with and without the presence of failures. Furthermore, we find that the presence of failures increases, on average, the recorded CPU contention, but on the other hand causes a much higher variation in recorded values. This is not unexpected, since failures may affect scheduling decisions or mask periods of high interference.

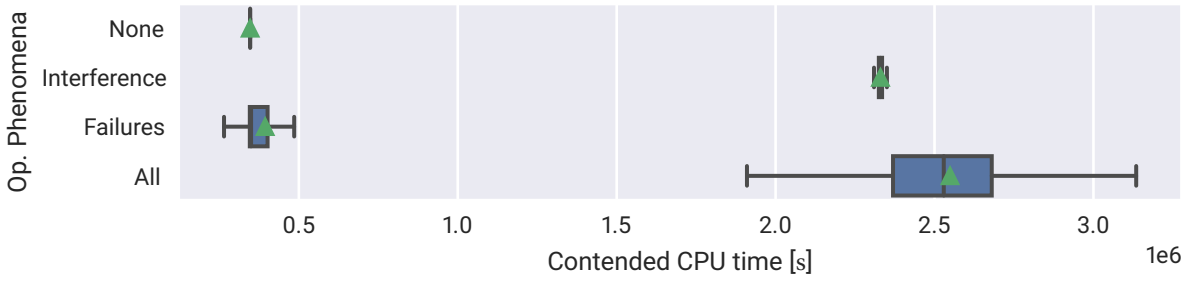


Figure 4.8: Total contended CPU time (in s) experienced by VMs for different operational phenomena over a timespan of three months.

Although workload interference represents a substantial portion of the recorded CPU contention values, it has a negligible impact on risk in our baseline scenario, as shown by Figure 4.4. Since resource utilization in the baseline workload is relatively low (see Table 4.5), there are fewer opportunities where interference could happen. Our experiments show that further increasing the vCPU overcommit ratio and thus placing more VMs on the same host eventually leads to a non-negligible QoS risk for the datacenter. Workloads that demand full performance, such as HPC, suffer more rapidly from workload interference, especially when such workloads are improperly scheduled. We explore the impact of different workloads in Section 4.4.10.

4.4.9. Impact of Datacenter Topology

Our main findings from this experiment are:

MF8 Amid the soaring electricity prices, datacenter operators are forced to compromise between topology scale and SLA compliance.

MF9 Radice's risk optimization algorithm reduces average risk in datacenters by a factor 0.65x–0.7x by optimizing the datacenter topology.

This experiment investigates the impact of the datacenter topology on risk in datacenters. We examine the impact of downscaling a datacenter on its risk, and also employ the built-in risk optimization algorithm of Radice to explore new hardware configurations and cluster sizes, optimizing the datacenter design for risk.

Scaling down datacenter infrastructure, and in turn, operating at a higher resource utilization, could enable cost reductions for electricity expenses. There is no free lunch however, because to satisfy customer SLAs, datacenters need to maintain enough capacity to serve customers, even in the presence of failures. Too little capacity and a datacenter will not be able to absorb the full workload as a result of a crash in another datacenter. We highlight this trade-off in Figure 4.9, where we compare the per-month risk of the baseline workload for different topology scales. Interestingly, the total risk of running the baseline workload at half the size of the original datacenter is almost equivalent to our baseline scenario. In this case, risk has become a trade-off between electricity expenses and SLA penalties due to lowered quality of service.

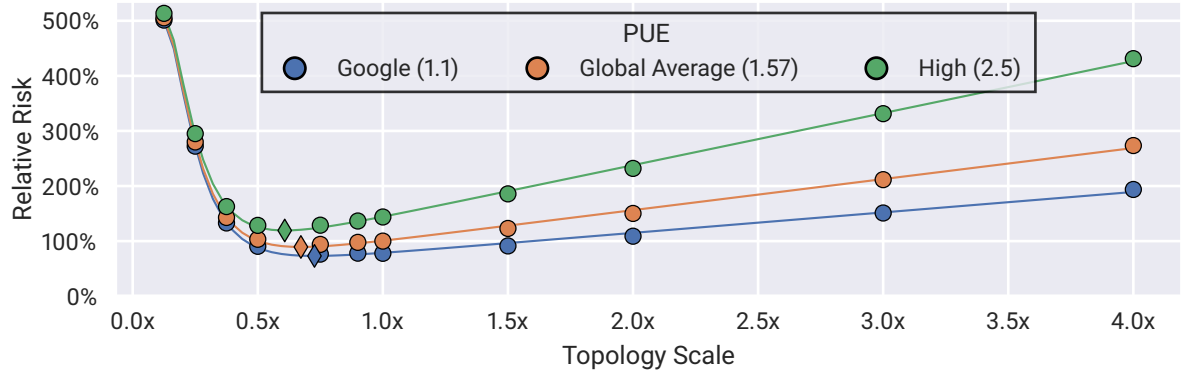


Figure 4.9: Regression of the average monthly risk on the topology scale (the number of machines) for the *baseline* workload and for different PUE values. The diamond markers indicate the minimum risk for each category.

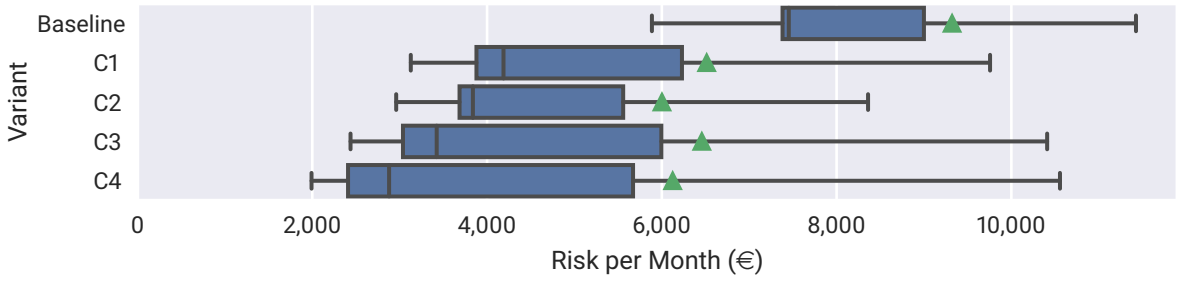


Figure 4.10: Per-month risk estimation of optimized topology candidates discovered by the risk exploration algorithm in Radice.

The figure also shows that a 0.75x scale leads to the lowest per-month risk of all explored scenarios, with a regression indicating the optimal scale lying between 0.5x–0.75x, depending on the energy efficiency of the datacenter, with energy-inefficient, high-PUE datacenters benefitting more from downscaling the topology. Note that these results do not take into account the costs of losing customers or reputational damage due to low-quality service. Furthermore, the figure exhibits a hockey stick type curve, where risk slowly increases as the scale of the datacenters is increased (due to higher electricity expenses), while the risk rises sharply as the datacenter is downscaled (as a consequence of SLA penalties).

So far, we have only considered the scale of the datacenter topology (the number of machines). In reality, there are countless other factors that influence the results of our experiments, such as the type of hardware used, the combination of hardware, or the age of the hardware. As we demonstrate in Section 3.5.4, considering each of these factors manually is infeasible, and instead, an intelligent approach for systematic exploration of the design space is necessary. The risk exploration algorithm in Radice (described in Section 3.5.4) is capable of exploring new hardware configurations and cluster sizes. In Figure 4.10, we depict the risk estimation for the top-4 best performing topologies discovered by our exploration algorithm. We observe that on average, the optimized topologies only incur a monthly risk of 0.65x–0.7x compared to the baseline topology, while for the median case only 0.4x–0.6x. The solution set contains two notable characteristics: (1) machines are primarily vertically scaled (e.g., increased CPU count), and (2) hardware models are more recent. Presumably, this is a result of the algorithm optimizing for the energy-efficiency of the topology.

Focusing now on the individual topologies, we find that, despite the best-case scenario for C4 incurring the lowest costs of all topologies, its worst-case scenario leads to the highest costs of all optimized topologies. This topology contains less than half the capacity of C1 (in terms of CPU count), which explains the high variability of this solution, since such a topology leaves little room for spare capacity in the event of a failure. By contrast, while C2 has a higher median risk, its variability is lower compared to other solutions. This solution stands out because it uses relatively many low-power commodity servers.

4.4.10. Impact of Workload

Our main findings from this experiment are:

MF10 Radice supports risk analysis for different types of workloads that appear in cloud datacenters.

MF11 Changing workloads can significantly affect the risk profile of a cloud datacenter.

This experiment investigates the impact of introducing new workloads for datacenter operators. Figure 4.11 depicts the risk estimation of running different workloads using our *baseline* topology and highlights significant differences between different workloads. The Materna workload carries a lower overall risk, but has a similar variability to the *baseline* workload, while the Azure workload has both a lower risk and variability compared to the *baseline* workload. In contrast, the Bitbrains workload carries a significantly higher risk and variability compared to the other workloads.

The trade-off between SLA penalties and operating costs (such as electricity and CO_2 emissions) that we observed in the baseline workload, is also evident for the Materna workload in Figure 4.12. Interestingly, while the risk profile is similar to the risk profile of the baseline workload, the Materna workload is less than one third the size of the baseline workload in terms of VM count. However, as Table 4.5 shows, VMs in the Materna workload have on average double the runtime compared to the baseline workload, meaning the number of active VMs in the system is higher.

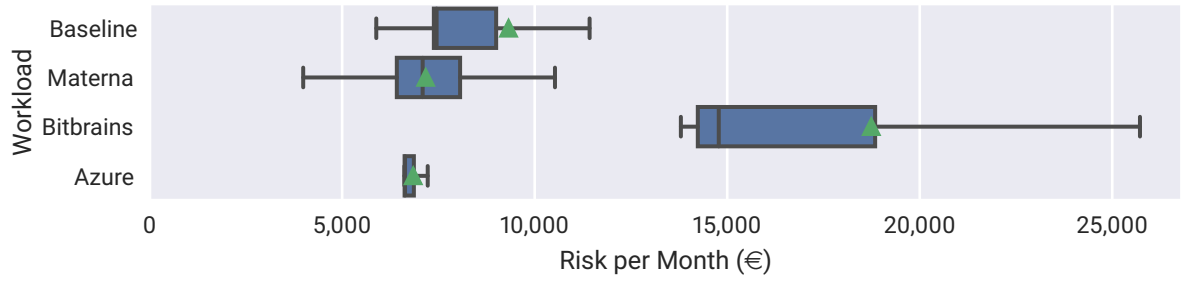


Figure 4.11: Per-month risk estimation of different workloads running on the *baseline* topology.

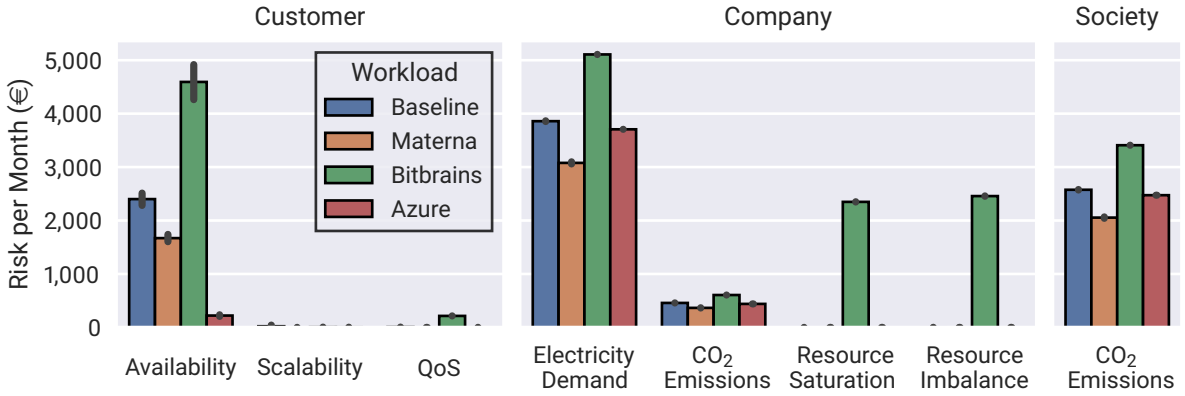


Figure 4.12: Risk profile for different workloads using the *baseline* topology.

Although the baseline and Bitbrains workloads are similarly sized, we observe relatively high host saturation and host imbalance in Figure 4.12, where we have depicted the risk profile of the Bitbrains workload running on the *baseline* topology. A high host saturation indicates that utilization of the physical hosts is too high on average, while a high host imbalance indicates that there is a large difference in utilization between physical hosts. This means either the datacenter is underprovisioned for our workload or the scheduler is not doing a good job distributing the load over the physical hosts. Further investigation shows that the workload contains a few HPC VMs that cause significant CPU usage, and in turn, high risk. Ideally, OpenDC's scheduler should place such VMs in a separate cluster and on separate nodes.

Finally, focusing on the Azure workload, we observe that for this workload the electricity costs and CO_2 emissions dominate the risk profile. In contrast, its availability risk is significantly lower than that of the other workloads. Usually, this indicates that the datacenter running the workload is over-provisioned. This could suggest that a public cloud workload (such as the Azure workload) requires less infrastructure compared to private cloud workloads. The short runtime of VMs in the Azure workload compared to the other workloads already provides evidence for this (see Table 4.5). To confirm our hypothesis, we run the Azure workload using scaled-down versions of our *baseline* topology and depict the risk estimations in Figure 4.13. We find that the Azure workload can run at a significantly lower cost in a datacenter a quarter the size of the *baseline* topology. As we scale down the topology, we observe the variability of the risk increasing; this is the same trade-off between SLA penalties and electricity costs that we observed in Section 4.4.7.

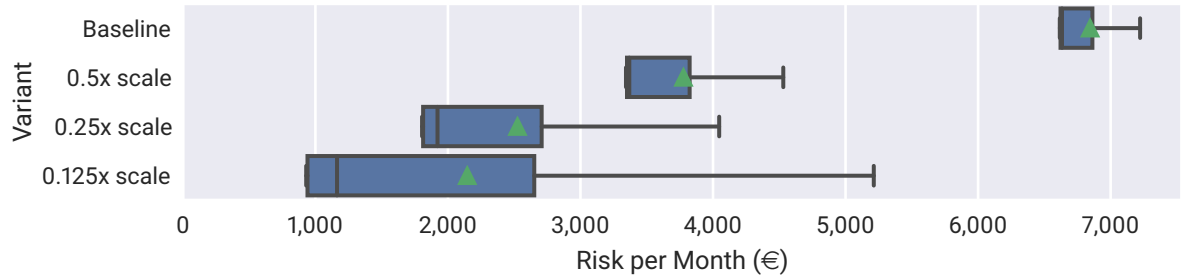


Figure 4.13: Per-month risk estimation of the Azure workload running on down-scaled topologies.

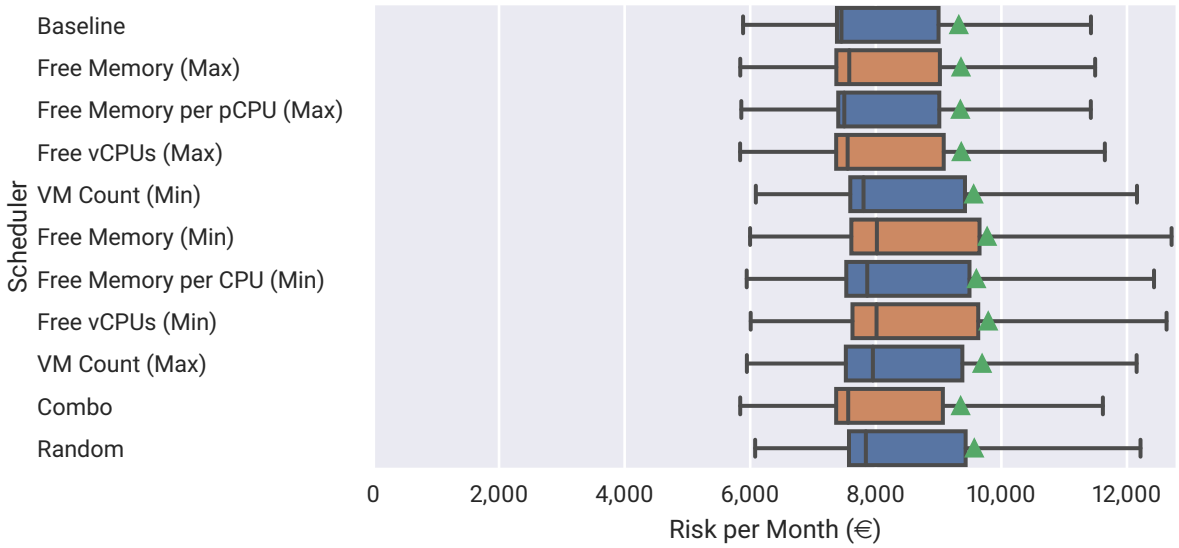


Figure 4.14: Per-month risk estimation of using different scheduler (traditional) configurations for the *baseline* workload.

4.4.11. Impact of Datacenter Scheduler

Our main findings from this experiment are:

MF12 Traditional VM placement policies cause approximately 5% of the per-month risk.

MF13 Radice's risk optimization algorithm finds scheduler configurations more efficient than traditional VM placement policies, with on average 7.5% higher performance.

MF14 A misconfigured scheduler can, in the worst case, heighten the risk by up to a factor 55x.

This experiment investigates the impact of different scheduler allocation policies on the risk profile of datacenters. Figure 4.14 shows how different *traditional* VM allocation policies affect the per-month risk reported by Radice. We observe that the difference between the selected VM allocation policies is on average only €420 or approximately 5% of the total per-month risk, with a difference of €250 at the 25th percentile and €570 at the 75th percentile. This suggests that switching between any of the traditional VM allocation policies carries a relatively small risk. On the other hand, it might also mean that switching allocation policies is not worth the risk, since the improvement is relatively small. Furthermore, we find that our baseline allocation policy is one of the best performing policies. It carries the lowest risk for the mean, median, and 75th percentile case compared to the other, *traditional* policies.

We also compare our baseline allocation policy to optimized scheduler configurations discovered by our optimization algorithm (as described in Section 3.5.4). Radice can automatically optimize the scheduler configuration for a datacenter topology and workload based on the estimated risk. We show in Figure 4.15 the top-4 best performing scheduler configurations and find that the optimized configurations provide moderate improvements in terms of per-month risk, on average a 2.5% lower than the *baseline* policy.

Again, the differences with the traditional scheduler configurations are relatively small, which might suggest that the traditional scheduler configurations are already well-optimized for our workload. More importantly, we expect more advanced techniques, such as live workload migration, to have a larger impact on the risk profile, whereas OpenDC's scheduler currently does not move VMs after initial placements.

Whereas both the traditional and optimized scheduler configurations cause only small improvements in terms of risk, we find that accidentally misconfiguring the scheduler carries a very high risk as we show in Figure 4.16, where we highlight the three worst-performing configurations discovered by our scheduler optimization algorithm. Such accidents carry on average a risk of over €60,000 in monthly incurred costs; an increase of a factor 6x. Further investigation shows that the CPU allocation ratio is the primary factor contributing to poor performance of these scheduler configurations. This ratio describes how many vCPUs a physical CPU can host and is usually set to 16x or greater, yet it was set to 1x–1.5x for these scheduler configurations, vastly lowering the effective cluster capacity and, in turn, server utilization.

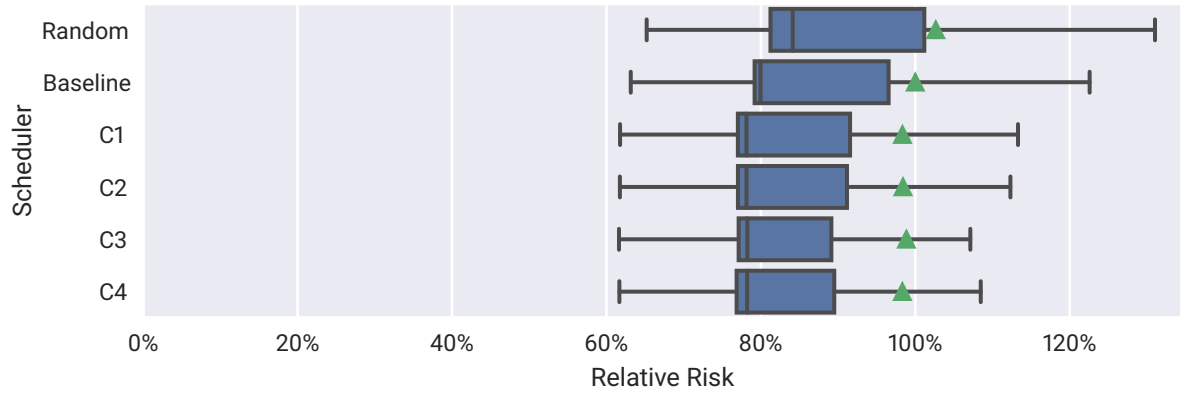


Figure 4.15: Risk of optimized scheduler configurations discovered by the risk optimization algorithm relative to the *baseline* scheduler.

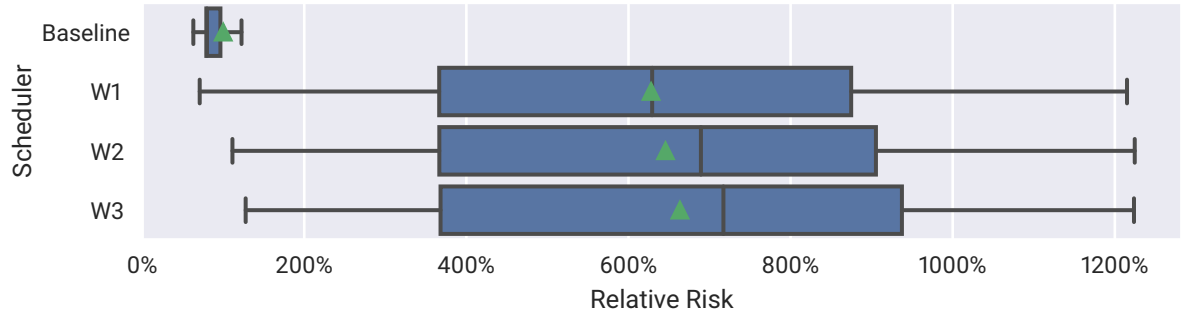


Figure 4.16: Risk of the worst-performing schedulers reported by the risk optimization algorithm relative to the *baseline* scheduler.

4.5. Performance Analysis of Radice

In this section, we analyze the performance of Radice (and OpenDC [107], the simulator underpinning it). Modern datacenter infrastructure operates at unprecedented scale [27]. For Radice to be useful for risk analysis of cloud datacenters, it must support the immense and growing scale of modern datacenters (addressing NFR1). To demonstrate the feasibility of using Radice for risk analysis of large-scale datacenter infrastructure, we benchmark the performance of Radice against another cloud simulator widely used in the community.

Overall, the main findings from the performance analysis are:

MF15 OpenDC is 70x–300x faster than a widely-used cloud simulator in the field, for the workloads considered in this thesis.

MF16 Radice can simulate three months of datacenter operation in a matter of seconds, enabling interactive risk exploration for practitioners.

MF17 OpenDC has similar memory usage to a cloud simulator in the field, for the workloads considered in this thesis.

4.5.1. Experiment Setup

We compare the performance of Radice against CloudSim Plus [58] (v7.1.1, released on 20 October 2021), a distribution of the well-known cloud simulation framework CloudSim [34]. We have selected CloudSim Plus since it is well-maintained, incorporates various bug-fixes and performance improvements, and is compatible with Java 17. In contrast, the original CloudSim repository has not had an official release since 2016.

Radice uses the latest development version of OpenDC[14], which already incorporates all performance improvements that have been developed as part of this work (see Section 4.3). These improvements, along with the Radice extensions will become available to all OpenDC users in the next release v2.1.

[14] `https://github.com/atlarge-research/opendc`

Table 4.10: Mean simulation runtime and peak memory usage (along with the standard deviation) of both Radice + OpenDC (v2.1) and CloudSim Plus (v7.1.1) when simulating various workloads.

Workload		Runtime [s]		Peak Memory Usage [MB]	
		Radice	CloudSim Plus	Radice	CloudSim Plus
Baseline	(5%)	0.11 ± 0.01	7.53 ± 0.02	59.01 ± 0.05	59.73 ± 0.05
Baseline	(10%)	0.16 ± 0.01	28.16 ± 0.08	100.96 ± 0.05	100.39 ± 0.05
Baseline	(25%)	0.66 ± 0.01	159.78 ± 1.08	246.37 ± 0.05	246.32 ± 0.10
Baseline	(50%)	1.69 ± 0.01	396.89 ± 0.92	484.69 ± 0.05	483.42 ± 0.10
Baseline	(100%)	3.29 ± 0.03	$1{,}099.83 \pm 3.01$	939.32 ± 0.05	942.05 ± 0.10

For both simulators, we construct a scenario that simulates the *baseline* workload (see Section 4.4.2) running in a datacenter using the *baseline* topology as described in Section 4.4.3, and which measures the total energy consumed by the machines in the datacenter. To take into consideration the scalability of both systems, we consider multiple scaled-down variants of this scenario for scale factors 50%, 25%, and 10%. We scale both the workload and topology in terms of size (virtual machines and physical machines respectively) according to the scale factor.

Since CloudSim Plus has no support for either the workload trace format or topology description format used by our experiments, we needed roughly 300 lines of code to set up the scenario and add support for these formats, compared to 90 lines for Radice. The modular design of OpenDC allowed us to re-use most of the existing code to support the aforementioned formats and integrate it into CloudSim Plus.

We measure both the *runtime* (in ms) and *peak memory usage* (in MB) during simulation, using performance counters exposed by the Java runtime. Our measurements do not include the experiment setup (e.g., reading the workload traces or parsing the topology description) or experiment tear-down.

All scenarios are replicated 32 times to ensure statistical correctness of the results. After every replication, we force the Java runtime to perform a complete garbage collection cycle, to prevent it from influencing subsequent replications. Each scenario is preceded by 4 (discarded) warm-up iterations, to allow the HotSpot JIT compiler to optimize the application, and to warm up the platform caches. This number of replications leads to a sufficiently small error in our measurements, supporting our observations about the overall performance of Radice (and OpenDC) with high confidence.

The experiments are executed using OpenJDK 17 (release 2021-09-14, Oracle distribution) on Arch Linux (Linux kernel 5.13.19), running on a machine fitted with an AMD Ryzen ThreadRipper 3990x 64-core chip, 128 GB of DDR4 RAM, and 8 TB of fast NVMe storage.

4.5.2. Results and Analysis

Table 4.10 lists the measurements of runtime (in s) and peak memory usage (in MB) for both simulators running the selected workloads. Since the runtime measurements differ significantly between both simulators, we also depict the runtime speedup of Radice and OpenDC compared to CloudSim Plus in Figure 4.17. We find that Radice is able to simulate the selected workloads orders of magnitudes faster than CloudSim Plus, ranging from a factor 70x in runtime for smaller workloads, to a factor 330x for the largest workload. In terms of absolute values, Radice is able to analyze three months of datacenter operation (of 1,800 virtual machines) in a matter of seconds, whereas CloudSim Plus requires *over 18 minutes* to evaluate the same scenario.

These results highlight the impact of the performance engineering that was done as part of this work (see Section 4.3) and shows that Radice addresses NFR1. The runtime performance of Radice opens possibilities for using the tool interactively while discussing operational changes, by providing datacenter operators with coarse estimates of risk for various "what-if" scenarios. For higher degrees of confidence, Radice can be used to evaluate thousands of risk scenarios overnight.

Whereas the runtime performance of Radice and CloudSim Plus differs significantly, the peak memory usage of both simulators is nearly identical, as is evident from Table 4.10. Further inspection shows that the main contributor to the memory usage for both simulators is the workload trace, which is loaded entirely into memory before simulation for performance reasons. This also demonstrated in Figure 4.18, where we depict the scaling behavior of the peak memory usage (and runtime) of Radice and OpenDC against the scale of the workload and observe that the memory usage grows linearly with the scale of the workload (that is, the number of virtual machines to simulate). OpenDC is able to amortize this cost across parallel simulation runs.

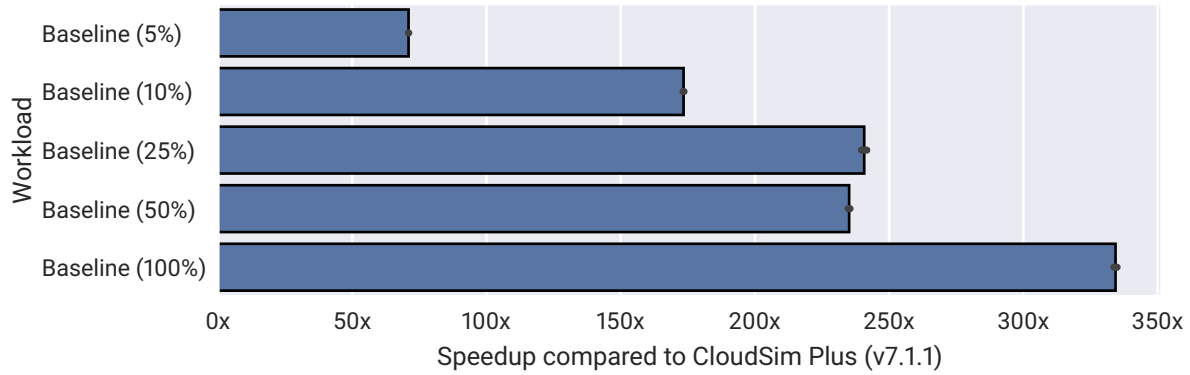


Figure 4.17: Runtime speedup of Radice and OpenDC (v2.1) compared to CloudSim Plus (v7.1.1) when simulating equivalent scenarios.

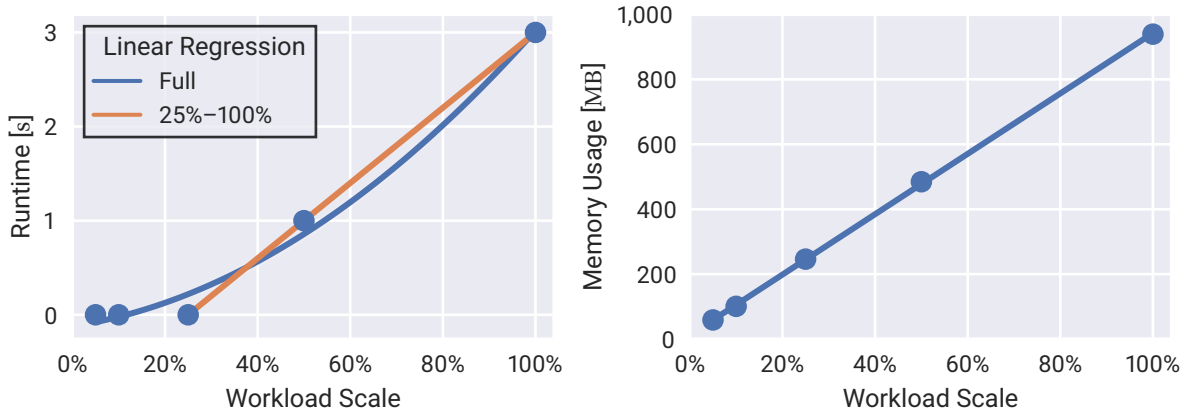


Figure 4.18: Scaling behavior of simulation runtime and peak memory usage of Radice and OpenDC (v2.1).

Based on observations while running the experiments, OpenDC consumes roughly 20 GB of memory for 80 parallel simulation runs. Although such scaling behavior might pose a problem for substantially large workloads that might not fit entirely into memory (for instance, Google Borg's [152] public cluster data from 2019 is roughly 2.6 TB compressed), this limitation can be addressed without much difficulty in a future version of OpenDC by reading the workload trace in batches.

In Figure 4.18, we also depict the scaling behavior of the simulation runtime and peak memory usage of Radice and OpenDC as the scale of the workload increases. We observe that the growth in runtime is not fully linear. However, we find that it becomes linear from a scale of 25% as the orange regression line shows. We believe this behavior is due to the small (milliseconds) runtime of the 5%- and 10%-scaled workloads, which causes the constant overhead to become apparent in the measurements. By contrast, the scaling behavior of the peak memory usage appears to be fully linear with respect to the workload scale.

4.6. Validation of Radice

We discuss in this section the validity of the results produced by Radice and (the extensions to) OpenDC. Radice facilitates data-driven risk analysis of datacenter infrastructure through trace-based discrete event simulation. Real-world experimentation might provide more accurate results, but using physical infrastructure to evaluate the vast amount of scenarios generated by Radice is difficult to reproduce, prohibitively expensive, and unable to capture the scale of modern datacenter infrastructure, as we argue in Section 2.3.3. We also analyze in Section 4.7 the environmental impact of our experiments.

Nonetheless, the ability to use Radice for risk analysis successfully heavily depends on the validity of the results produced by it. For this reason, we employ a systematic process to ensure validity of Radice and OpenDC, which consists of the following three aspects: (1) validation using manual inspection, (2) validation using another simulator, and (3) validation using queuing theory. Below, we discuss for each of these aspects our approach and results. Finally, we also discuss how we ensure the results produced by the simulator remain valid after the addition or modification of functionality in subsequent versions of the simulator.

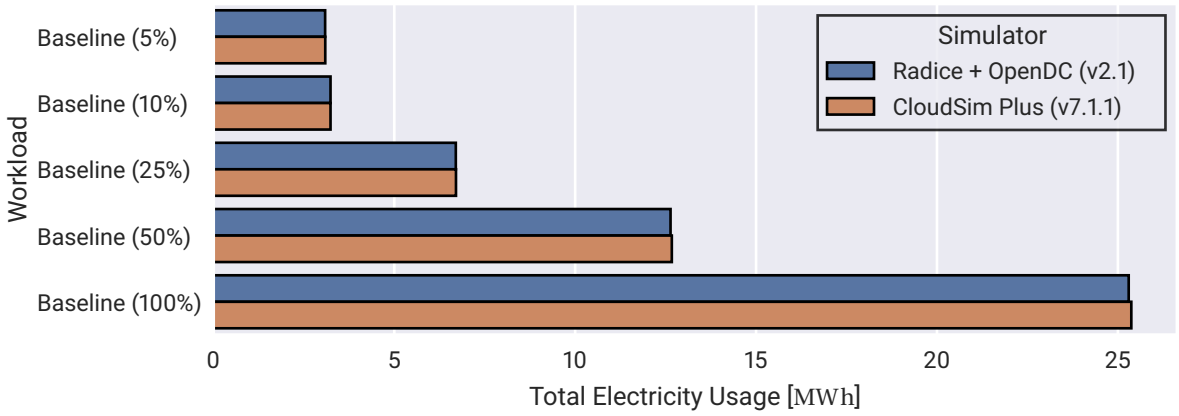


Figure 4.19: Electricity usage of the datacenter (in MWh) reported by Radice and OpenDC (v2.1) compared to CloudSim Plus (v7.1.1) when simulating equivalent scenarios.

4.6.1. Validation using Manual Inspection

We ensure validity of simulator results by tracking a wide variety of metrics during the execution of simulations in order to validate the behavior of the system. This selection consists of metrics of interest which we analyze in our experiments, but also fail-safe metrics (e.g., total CPU time) that we can verify against known values. We combine this with step-by-step inspection using the various tools offered by the Java ecosystem (e.g., Java Debugger, Java Flight Recorder, and VisualVM) to verify the state of individual components on a per-cycle basis.

Furthermore, we have had several meetings with both industry and domain experts to discuss the simulator results in depth, validate our models and assumptions, and spot inconsistencies. Our communication with experts is proactive when possible issues with the simulator arise during development, such as unclear observations.

4.6.2. Validation using another Simulator

Our second approach to validation involves a comparison against another simulator. We re-use the experiment setup from the Performance Analysis (see Section 4.5.1), which compares Radice and OpenDC (v2.1) to CloudSim Plus (v7.1.1), and collects the electricity usage of the datacenter reported by both simulators. Only a single repetition of these experiments is necessary, since both simulators are deterministic and have been verified to report the same electricity usage for identical parameters across simulation runs.

Figure 4.19 shows the electricity usage of the datacenter reported by Radice compared to CloudSim Plus when simulating equivalent scenarios. We observe that the electricity usage reported by both simulators is nearly identical. Only for the larger workload scales (50% and 100%), a small difference is visible in Figure 4.19. However, we believe the small difference is due to lack of support for performance interference, as well as incomplete support for CPU overcommitment in CloudSim Plus' VM scheduler, which causes small differences in the reported host CPU usage. In fact, the developers of CloudSim Plus have recently removed support for CPU overcommitment altogether due to reliability issues. In contrast, OpenDC' VM scheduler fully supports overcommitment of CPU resources and has been validated thoroughly during development.

Furthermore, manual inspection of the results shows that OpenDC is able to generate an identical VM allocation schedule as produced by the default allocation policy of CloudSim Plus (worst-fit). This demonstrates that while OpenDC's scheduler implementation (see Section 4.2.3) is vastly different from the scheduler of CloudSim Plus, it is still able to produce the same results.

4.6.3. Validation using Queuing Theory

We attempt to validate Radice against existing queuing theoretic models. Our goal is to reproduce the results reported by Radice using analysis of the selected queuing model. We model with Radice and OpenDC an M/M/c queuing system. In this model, arriving virtual machines form a single queue and there are c servers that host the virtual machines. Servers can only host a single virtual machine at any point in time. Virtual machine requests arrive in the system at a rate λ according to a Poisson process, and have a runtime that is exponentially distributed at a rate of μ. We list a full description of the system parameters and outputs in Table 4.11 and show the derivation of the outputs based on the system parameters in Equation 4.20 [18].

Table 4.11: Mathematical symbols used in queuing theory, along with their definition.

Symbol	Definition
λ	*Arrival rate* – Number of VMs that are submitted per hour
μ	*Service rate* – Number of VMs that complete per hour
c	Number of hosts in the system
L	Average number of VMs in the system
L_Q	Average number of VMs in the queue
W	Average time spent (in hours) by a VM in the system (from submission to finish)
W_Q	Average time spent (in hours) by a VM in the queue
ρ	Average host utilization

$$\rho = \frac{\lambda}{c \cdot \mu}$$

VM arrival rate (per hour)

VM service rate (per hour)

Host count

$$P_0 = \left(\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!} \cdot \frac{1}{1-(\lambda/(c\mu))} \right)^{-1}$$

$$L_Q = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2} P_0 \qquad L = L_Q - \frac{\lambda}{\mu}$$

$$W = \frac{L}{\lambda} \qquad W_Q = W - \frac{1}{\mu}$$

Figure 4.20: Derivations of queuing theoretic properties of a M/M/c queuing system [18].

We test multiple combinations of system parameters. For each combination, we run the simulation until at least 5 million VMs have been processed by the system. Each experiment run takes 5 minutes with the setup described in Section 4.5.1.

In Table 4.12, we list the measurements reported by Radice for various combinations of system parameters, along with the expected theoretic values derived from mathematical analysis (see Equation 4.20). We find that almost all values reported by our simplified model of Radice match the theoretical properties of an M/M/c queuing system. There are just a few cases where the empirical measurements deviate from the theoretical value, but in those cases, the difference is negligible.

4.6.4. Safeguarding against Regressions

Although we may at one point trust the simulator to produce correct outputs, the addition or modification of functionality in subsequent versions of the simulator may inadvertently affect the output compared to previous versions.

Table 4.12: Properties of various M/M/c queuing systems derived from queuing theory and from empirical measurements reported by Radice. See Table 4.11 for the definitions of the symbols used in this table.

λ	μ	c	L		L_Q		W		W_Q		ρ	
			T (Theory)	R (Radice)	T	R	T	R	T	R	T	R
4	1	5	6.22	6.22	2.22	2.21	1.55	1.55	0.55	0.55	0.80	0.80
4	1	6	4.57	4.57	0.57	0.57	1.14	1.14	0.14	0.14	0.67	0.67
4	1	10	4.01	4.01	0.01	0.01	1.00	1.00	0.00	0.00	0.40	0.40
28	3	10	20.13	20.10	10.80	10.74	0.72	0.72	0.39	0.38	0.93	0.93
28	3	12	10.46	10.46	1.12	1.72	0.37	0.37	0.04	0.04	0.78	0.78
16	0.75	22	48.20	47.87	26.87	26.52	3.01	3.00	1.68	1.66	0.97	0.97
16	0.75	24	25.13	25.12	3.79	3.76	1.57	1.57	0.24	0.24	0.89	0.89

We safeguard against such issues by means of snapshot testing. With snapshot testing, we capture a snapshot of the system outputs and compare it against the outputs produced by subsequent simulator versions. For these tests, we consider a downsized variant of the experiments run in this work, which captures the same metrics. These tests execute after every change and ensure that the validity of the simulator outputs is not affected. In case output changes are intentional, the test failures serve as a double check.

In addition, we utilize assertions in various parts of the simulator to validate internal assumptions. This includes checking that simulation events are not delivered out-of-order and verifying that simulated machines do not enter invalid states.

Finally, as we discuss in Section 4.2.1, we employ industry-standard development practices. Every change to the simulator or its extensions requires an independent code review before inclusion in the main code base. Through CI, we ensure that all changes are automatically analyzed using static code analysis tools (e.g., linting) in order to identify common mistakes.

4.7. Environmental Impact of Radice

State-of-the-art research increasingly requires considerable computational resources [77], resulting in large financial costs [131], substantial energy usage, and in turn significant greenhouse gas emissions [139]. This leads to higher inequality among the community (due to financial requirements) and unsustainable environmental impact (due to resulting greenhouse gas emissions).

In this section, we analyze the environmental impact of our experiments, determining their electricity requirements, and deriving the resulting financial costs and CO_2 emissions. We then compare these results against the hypothetical situation of replicating our experiments on real-world infrastructure.

Overall, the main findings from this analysis are:

MF18 Our experiments consume approximately 4.5 kWh of electricity, roughly equivalent to powering the fridge for a whole day.

MF19 The electricity demand for replicating our experiments on real-world infrastructure is equivalent to powering 3,400 hospitals for a whole year or 2 billion fridges per day.

As part of this work, we have simulated more than 50,000 years of datacenter operation resulting from over 250,000 simulation runs. In total, our experiments consumed approximately 4.5 kWh of electricity (measured at the wall socket) at a combined cost of €1.4, roughly equivalent to powering the fridge for a whole day[15]. This has likely resulted in 2.5 kg of CO_2 emissions, based on the energy profile of the Netherlands [138, 159].

In contrast, replicating our experiments on real-world infrastructure has an extremely severe environmental impact. Combined, our experiments consumed over 8.5 TWh of electricity in simulation, equivalent to powering 3,400 hospitals or 10,000+ schools for a whole year[16], also 2 billion fridges for one day. This scenario would result in an electricity bill of more than €2 billion and over 4.5 million tonnes of CO_2 emissions.

4.8. Discussion

We now summarize the contributions of this chapter and discuss possible threats to their validity.

4.8.1. Summary

We present a working prototype of Radice based on the design and requirements outlined in Chapter 3. Our experiments demonstrate that Radice can assist datacenter operators faced with complex trade-offs involving risk. We highlight the impact of the increasing electricity and CO_2 emission prices for datacenter operators, with it becoming a substantial contributor to the risk profile of datacenters. We also show that Radice is able to evaluate scenarios by a factor 70x–330x faster compared to a widely-used cloud simulator. Finally, we have validated Radice systematically using manual inspection, using another simulator, and using queuing theory.

4.8.2. Threats to Validity

We discuss potential threats to validity in terms of internal validity, construct validity, and external validity.

[15] https://www.siliconvalleypower.com/residents/save-energy/appliance-energy-use-chart
[16] https://illumination.duke-energy.com/articles/how-you-helped-save-1-billion-kilowatt-hours

Internal Validity
The lack of *full details about the workload trace and datacenter topology* is a threat to the internal validity of this work, since our findings may not actually be reflected by the full, confidential data artifacts. We are prevented from releasing the full artifacts and details, due to confidentiality of the workload trace and datacenter topology that we use in our experiments. To address this threat, we release an anonymized and restricted version of our experiments, which can be used to reproduce the findings in this work. Furthermore, we also experiment with three other workload traces that are available in public trace archives [78].

Construct Validity
The *validity of the results* produced by Radice and OpenDC could pose a threat to the construct validity of the experiments. This threat also partially extends to external validity, since it affects both the quality and reliability of measurements in our study, as well as the extent to which findings can be generalized. However, we believe this threat has been addressed extensively in Section 4.6.

External Validity
The lack of *different resource types* in our experiments is a threat to external validity. We consider in this work only the impact of computing resources on risk. Although OpenDC recently received support for memory, networking, and storage, we decided to leave out these resources in this work due to the immaturity and missing validation of these models in OpenDC, as well as the relative lack of comprehensive workload data to effectively model the operation of these resources in our experiments.

5

Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis and envision areas of future work that may emerge from it.

5.1. Conclusion

We investigate in this work three main research questions concerning the analysis of operational risk in sustainable cloud datacenters. We first describe the problem of risk analysis in datacenters in Chapter 1 and provide more background in Chapter 2. In Chapters 3 and 4, we present the main contributions of this thesis. We now answer each of the main research questions in turn:

RQ1 How to model IT-related risks in sustainable cloud infrastructure?
We have constructed a model for quantifying IT-related operational risks that emerge in sustainable cloud infrastructure. Our model incorporates various insights from state-of-the-art risk analysis in closely-related fields, as well as exploiting the vast amount of operational data collected in datacenters, including environmental data. A risk profile of a cloud datacenter is constructed from a set of risk factors, each defined as an objective over some metric and an associated cost function for violating that objective. Our model also accounts for the sustainability of datacenters (and the associated risks), by incorporating electricity usage and CO_2 emissions of datacenters.

RQ2 How to design a system for simulation-based analysis of IT-related risks in cloud infrastructure?
We have conducted a requirement analysis for a risk assessment system for sustainable cloud infrastructure, identifying potential stakeholders, documenting use-cases, and in turn synthesizing ten requirements. Based on these requirements, we have designed Radice. Radice incorporates discrete-event simulation to model datacenter infrastructure and diverse workloads accurately and timely. It features the ability to define a risk model, to explore what-if risk scenarios (such as outages or interference), to evaluate risk profiles, and to propose changes to the datacenter that reduce risk. Radice embodies reproducible science, by ensuring all simulation results are reproducible and enabling users to share and replicate experiments through a reproducibility capsule.

RQ3 How to evaluate and validate a system for risk analysis of cloud infrastructure?
We have implemented a working software prototype of Radice, realizing key features of the design. Our implementation extends the OpenDC codebase with support for new resources, workloads, and metrics, and with significant performance improvements. To demonstrate Radice's capabilities, we have conducted extensive experiments using real-world workload traces from both private and public clouds. Our findings show that Radice is able to take into account diverse factors that impact the risk of cloud datacenters. We find evidence that increasing electricity and CO_2 prices may significantly impact the financial sustainability of datacenters. We also show that Radice is a factor 70x–330x faster compared to a widely-used cloud simulator.

We have released Radice as free and open-source software (FOSS) for the community to use. Radice is engineered using modern software engineering practices and produces reproducible results.

5.2. Future Work

We envision five areas of future work, building upon the contributions presented in this work:

1. We plan to *broaden our model of IT-related operational risk in sustainable cloud infrastructure*. This includes extending the model with risk factors related to networking, security, and other environmental factors (e.g., water consumption). By supporting more risk factors in Radice, we increase the accuracy of the results.

2. We intend to *keep improving Radice* to become a production-ready system for risk analysis of cloud infrastructure. We are continuously working to improve and expand the capabilities of the underlying OpenDC simulation platform. This includes support for new emerging workloads, metrics, and resource management techniques. Any improvements in this category directly benefit users of Radice as well. Furthermore, we plan to further engineer Radice and implement its full feature set. Our goal is to make Radice widely adoptable for diverse stakeholders, by improving support for dynamic, interactive design of datacenters and risk scenarios, and including an extensive library of prefab topologies, workloads, and experiments. We also plan to integrate Radice with existing operational systems in datacenters, such as the software from VMWare or the open-source monitoring project Grafana. Such functionality could enable online and near-real-time risk analysis of datacenter infrastructure with little involvement from the datacenter operator.

3. We are *investigating the use of more efficient Artificial Intelligence search techniques for risk exploration*. Automatic design space exploration assists users by evaluating many different points in the design space and presenting key trade-offs. Although we demonstrate in this work that Radice supports automatic design space exploration, we have not focused on selecting the most efficient method of exploration in this work. Research into design space exploration techniques for datacenters could help improve the quality of the solutions proposed by Radice and reduce the time to answer.

4. We see possibilities for *utilizing Radice for real-time decision-making in datacenters based on the current risk*. Radice can assist datacenter operators in managing incidents, by highlighting issues, proposing solutions, or even applying changes to the datacenter automatically. This is similar to the concept of portfolio scheduling, but can be applied more broadly.

5. We plan to *develop educative material around Radice and OpenDC*. We envision a series of interactive courses integrated directly into OpenDC and Radice that educates students on the operation of datacenters, the design process, and the impact of design decisions in datacenters. Since OpenDC is directly available in the web browser and runs simulations in the cloud, we are able to engage in large-scale computer systems a diverse group of students, including people that are currently under-represented.

Acronyms

AI artifical intelligence. 29

CI continuous integration. 32, 54

DVFS dynamic frequency and voltage scaling. 27, 34

FOSS free and open-source software. 5, 6, 57

HPC high performance computing. 9, 45, 47

ML machine learning. 23

PPA power purchase agreement. 42

PUE power usage effectiveness. 2, 3, 38, 42, 43, 46

QoS Quality of Service. 15, 39, 40, 45

SAN storage area network. 38

SLA service level agreement. 11, 15, 17, 26, 40, 44–47

SLI service level indicator. 11

SLO service level objective. 11, 19, 24, 26

SRE site reliability engineering. 2

VM virtual machine. 2, 9, 11, 21, 23, 26, 33, 34, 36–40, 44–48

Bibliography

[1] ISO/IEC 30134-2:2016. Information technology – Data centres – Key performance indicators — Part 2: Power usage effectiveness (PUE). Standard, International Organization for Standardization, Geneva, CH, Apr 2016.

[2] ISO 31000:2018. Risk management – Guidelines. Standard, International Organization for Standardization, Geneva, CH, Feb 2018.

[3] ISO/IEC 73:2009. Risk management – Vocabulary. Standard, International Organization for Standardization, Geneva, CH, Nov 2009.

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, pages 265–283. USENIX Association, 2016.

[5] David Adam. Special report: The simulations driving the world's response to covid-19. *Nature*, 580 (7802):316–319, 2020.

[6] Sameer Hasan Albakri, Bharanidharan Shanmugam, Ganthan Narayana Samy, Norbik Bashah Idris, and Azuan Ahmed. Security risk assessment framework for cloud computing environments. *Security and Communication Networks*, 7(11):2114–2124, 2014.

[7] Paul Albuquerque, Bastien Chopard, Christian Mazza, and Marco Tomassini. On the impact of the representation on fitness landscapes. In *Genetic Programming, European Conference, Edinburgh, Scotland, UK, April 15-16, 2000, Proceedings*, volume 1802 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[8] Omid Alipourfard, Jiaqi Gao, Jérémie Koenig, Chris Harshaw, Amin Vahdat, and Minlan Yu. Risk based planning of network changes in evolving data centers. In Tim Brecht and Carey Williamson, editors, *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pages 414–429. ACM, 2019.

[9] Abdulrahman Almutairi, Muhammad Ihsanulhaq Sarfraz, and Arif Ghafoor. Risk-aware management of virtual resources in access controlled service-oriented cloud datacenters. *IEEE Transactions Cloud Computing*, 6(1):168–181, 2018.

[10] Anders SG Andrae and Tomas Edler. On global electricity usage of communication technology: trends to 2030. *Challenges*, 6(1):117–157, 2015.

[11] Georgios Andreadis, Laurens Versluis, Fabian Mastenbroek, and Alexandru Iosup. A reference architecture for datacenter scheduling: design, validation, and experiments. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, USA, November 11-16, 2018*, pages 37:1–37:15. IEEE / ACM, 2018.

[12] Georgios Andreadis, Fabian Mastenbroek, Vincent Van Beek, and Alexandru Iosup. Capelin: Data-driven compute capacity procurement for cloud datacenters using portfolios of scenarios. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–1, 2021.

[13] Rhonda Ascierto. Too big to fail? facebook's global outage. *Uptime Institute Journal*, Oct 2021. URL `https://journal.uptimeinstitute.com/too-big-to-fail-facebooks-global-outage`. [Online; accessed 9-Dec-2021].

[14] Henri E. Bal, Dick H. J. Epema, Cees de Laat, Rob van Nieuwpoort, John W. Romein, Frank J. Seinstra, Cees Snoek, and Harry A. G. Wijshoff. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer*, 49(5):54–63, 2016.

[15] Ilyas Bambrik. A survey on cloud computing simulation and modeling. *SN Computer Science*, 1(5):249, 2020.

[16] Ajay Kumar Bangla, Alireza Ghaffarkhah, Ben Preskill, Bikash Koley, Christoph Albrecht, Emilie Danna, Joe Jiang, and Xiaoxue Zhao. Capacity planning for the google backbone network. In *ISMP 2015 (International Symposium on Mathematical Programming)*, 2015.

[17] Jerry Banks, John S. Carson II, Barry L. Nelson, and David M. Nicol. *Discrete-Event System Simulation*. Pearson, 2014.

[18] Michel Barbeau and Evangelos Kranakis. *Principles of ad hoc networking*. Wiley, 2007.

[19] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2018.

[20] Bruno Basalisco, Sixten Rygner Holm, Christoffer Haag Theilgaard, Cecilia Gustafsson, Niels Christian Fredslund, Martin Bo Westh Hansen, Eva Rytter Sunesen, and Martin Hvidt Thelle. European data centres: How Google's digital infrastructure investment is supporting sustainable growth in Europe. Technical report, Copenhagen Economics, Feb 2018.

[21] Salman Abdul Baset, Long Wang, and Chunqiang Tang. Towards an understanding of oversubscription in cloud. In *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE'12, San Jose, CA, USA, April 24, 2012*. USENIX Association, 2012.

[22] Rabih Bashroush. A comprehensive reasoning framework for hardware refresh in data centers. *IEEE Transactions on Sustainable Computing*, 3(4):209–220, 2018.

[23] Rabih Bashroush and Andy Lawrence. Beyond PUE: Tackling IT's wasted terawatts. Technical report, Uptime Institute, Jan 2020.

[24] Rabih Bashroush and Eoin Woods. Architectural principles for energy-aware internet-scale applications. *IEEE Software*, 34(3):14–17, 2017.

[25] Dominic Battré, Frances M. T. Brazier, Kassidy P. Clark, Michel A. Oey, Alexander Papaspyrou, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. A proposal for ws-agreement negotiation. In *Proceedings of the 2010 11th IEEE/ACM International Conference on Grid Computing, Brussels, Belgium, October 25-29, 2010*, pages 233–241. IEEE Computer Society, 2010.

[26] Emery D. Berger, Stephen M. Blackburn, Matthias Hauswirth, and Michael W. Hicks. A checklist manifesto for empirical evaluation: A preemptive strike against a replication crisis in computer science. `https://blog.sigplan.org/2019/08/28/a-checklist-manifesto-for-empirical-evaluation-a-preemptive-strike-against-a-replication-crisis-in-computer-science`, 2019. [Online; accessed 9-Dec-2021].

[27] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.

[28] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[29] Daniel Bizo, Rhonda Ascierto, Andy Lawrence, and Jacqueline Davis. 2021 Data Center Industry Survey Results. Technical report, Uptime Institute, Sep 2021.

[30] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

[31] David Breitgand, Zvi Dubitzky, Amir Epstein, Alex Glikson, and Inbar Shapira. Sla-aware resource overcommit in an iaas cloud. In *8th International Conference on Network and Service Management, CNSM 2012, Las Vegas, NV, USA, October 22-26, 2012*, pages 73–81. IEEE, 2012.

[32] Dominik Brodowski. Cpu frequency and voltage scaling code in the linux(tm) kernel. `https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt`, 2020. [Online; accessed 9-Dec-2021].

[33] James Byrne, Sergej Svorobej, Konstantinos M. Giannoutakis, Dimitrios Tzovaras, Peter J. Byrne, Per-Olov Östberg, Anna Gourinovitch, and Theo Lynn. A review of cloud computing simulation platforms and related environments. In Donald Ferguson, Víctor Méndez Muñoz, Jorge S. Cardoso, Markus Helfert, and Claus Pahl, editors, *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017*, pages 651–663. SciTePress, 2017.

[34] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[35] Marcus Carvalho, Daniel A. Menascé, and Francisco Vilar Brasileiro. Capacity planning for iaas cloud providers offering multiple service classes. *Future Generation Computer Systems*, 77:97–111, 2017.

[36] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, 2014.

[37] Henri Casanova, Rafael Ferreira da Silva, Ryan Tanaka, Suraj Pandey, Gautam Jethwani, William Koch, Spencer Albrecht, James Oeth, and Frédéric Suter. Developing accurate and scalable simulators of production workflow management systems with WRENCH. *Future Generation Computer Systems*, 112: 162–175, 2020.

[38] Henri Casanova, Ryan Tanaka, William Koch, and Rafael Ferreira da Silva. Teaching parallel and distributed computing concepts in simulation with WRENCH. *Journal of Parallel and Distributed Computing*, 156:53–63, 2021.

[39] Gabriel G. Castañé, Alberto Nuñez, Pablo Llopis, and Jesús Carretero. E-mc^2: A formal framework for energy modelling in cloud computing. *Simulation Modelling Practice and Theory*, 39:56–75, 2013.

[40] Patralekha Chatterjee. Indian air pollution: loaded dice. *The Lancet Planetary Health*, 3(12):e500–e501, 2019.

[41] Pravir K. Chawdhry, Rajkumar Roy, and Raj K. Pant. *Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag, 1997.

[42] Weiwei Chen and Ewa Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *8th IEEE International Conference on E-Science, e-Science 2012, Chicago, IL, USA, October 8-12, 2012*, pages 1–8. IEEE Computer Society, 2012.

[43] Philip Chow and Matthew Walker. Risk mitigation strategies for emergency power upgrades in critical facilities. In *2019 IEEE/IAS 55th Industrial and Commercial Power Systems Technical Conference (I CPS)*, pages 1–6, 2019.

[44] Peter Christoffersen. *Elements of financial risk management*. Academic Press, 2011.

[45] Andy Cockburn, Pierre Dragicevic, Lonni Besançon, and Carl Gutwin. Threats of a replication crisis in empirical computer science. *Communications of ACM*, 63(8):70–79, 2020.

[46] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 153–167. ACM, 2017.

[47] Rafael Ferreira da Silva, Loïc Pottier, Tainã Coleman, Ewa Deelman, and Henri Casanova. Workflowhub: Community framework for enabling scientific workflow research and development. In *IEEE/ACM Workflows in Support of Large-Scale Science, WORKS@SC 2020, Atlanta, GA, USA, November 12, 2020*, pages 49–56. IEEE, 2020.

[48] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. RAPL: memory power estimation and capping. In *Proceedings of the 2010 International Symposium on Low Power Electronics and Design, 2010, Austin, Texas, USA, August 18-20, 2010*, pages 189–194. ACM, 2010.

[49] Homayoon Dezfuli, Allan Benjamin, Christopher Everett, Gaspare Maggio, Michael Stamatelatos, Robert Youngblood, Sergio Guarro, Peter Rutledge, James Sherrard, Curtis Smith, and Rodney Williams. *NASA Risk Management Handbook*. National Aeronautics and Space Administration, 2011.

[50] Karim Djemame, Iain Gourlay, James Padgett, Georg Birkenheuer, Matthias Hovestadt, Odej Kao, and Kerstin Voß. Introducing risk management into the grid. In *Second International Conference on e-Science and Grid Technologies (e-Science 2006), 4-6 December 2006, Amsterdam, The Netherlands*, page 28. IEEE Computer Society, 2006.

[51] Karim Djemame, James Padgett, Iain Gourlay, and Django Armstrong. Brokering of risk-aware service level agreements in grids. *Concurrency and Computation: Practice and Experience*, 23(13):1558–1582, 2011.

[52] Karim Djemame, Django Armstrong, Jordi Guitart, and Mario Macías. A risk assessment framework for cloud computing. *IEEE Transactions on Cloud Computing*, 4(3):265–278, 2016.

[53] Paul M Duvall, Steve Matyas, and Andrew Glover. *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[54] Erwin Van Eyk, Alexandru Iosup, Johannes Grohmann, Simon Eismann, André Bauer, Laurens Versluis, Lucian Toader, Norbert Schmitt, Nikolas Herbst, and Cristina L. Abad. The SPEC-RG reference architecture for faas: From microservices and containers to serverless platforms. *IEEE Internet Computing*, 23(6):7–18, 2019.

[55] Dror G. Feitelson, Dan Tsafrir, and David Krakov. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing*, 74(10):2967–2982, 2014.

[56] Neil M Ferguson, Derek AT Cummings, Simon Cauchemez, Christophe Fraser, Steven Riley, Aronrag Meeyai, Sopon Iamsirithaworn, and Donald S Burke. Strategies for containing an emerging influenza pandemic in southeast asia. *Nature*, 437(7056):209–214, 2005.

[57] Ana Juan Ferrer, Francisco Hernández-Rodriguez, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, Raül Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora A. Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, Nikolaus Forgó, Tabassum Sharif, and Craig Sheridan. OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1):66–77, 2012.

[58] Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Pedro R. M. Inácio, and Mário M. Freire. Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, May 8-12, 2017*, pages 400–406. IEEE, 2017.

[59] Josep Oriol Fitó and Jordi Guitart. Business-driven management of infrastructure-level risks in cloud providers. *Future Generation Computer Systems*, 32:41–53, 2014.

[60] Flexera. State of the Cloud Report. Technical report, Flexera, Sep 2021. [Online; accessed 9-Dec-2021].

[61] Martin Fowler. Continuous integration. `https://martinfowler.com/articles/continuousIntegration.html`, May 2006. [Online; accessed 9-Dec-2021].

[62] Matthieu Gallet, Nezih Yigitbasi, Bahman Javadi, Derrick Kondo, Alexandru Iosup, and Dick H. J. Epema. A model for space-correlated failures in large-scale distributed systems. In *Euro-Par 2010 - Parallel Processing, 16th International Euro-Par Conference, Ischia, Italy, August 31 - September 3, 2010, Proceedings, Part I*, volume 6271 of *Lecture Notes in Computer Science*, pages 88–100. Springer, 2010.

[63] Gartner Inc. Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences. Press Release, Nov 2021. [Online; accessed 9-Dec-2021].

[64] Frank Gens. Worldwide and Regional Public IT Cloud Services 2019–2023 Forecast. Tech. Rep. by IDC, Doc. #US44202119, Aug 2019. [Online; accessed 9-Dec-2021].

[65] Glanz. Data Centers Waste Vast Amounts of Energy, Belying Industry Image. *N.Y. Times*, 2012.

[66] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In Marinho P. Barcellos, Jon Crowcroft, Amin Vahdat, and Sachin Katti, editors, *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*, pages 58–72. ACM, 2016.

[67] Albert G. Greenberg, James R. Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *Comput. Commun. Rev.*, 39(1):68–73, 2009.

[68] Stijn Grove, Judith de Lange, Eline Stuivenwold, Erik Barentsen, Zoë Derksen, Peter Vermeulen, and Penny Madsen. State of the Dutch Centers 2021. Technical report, Dutch Data Center Association, Jun 2020.

[69] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 1332–1340. IEEE, 2011.

[70] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[71] Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[72] Dirk Harryvan, Marco Verzijl, and Max Amzarakov. Analysis LEAP Track 1 'Powermanagement'. Technical report, Netherlands Enterprise Agency, Jan 2021.

[73] Md Sabbir Hasan, Yousri Kouki, Thomas Ledoux, and Jean-Louis Pazat. Exploiting renewable sources: When green SLA becomes a possible reality in cloud computing. *IEEE Transactions on Cloud Computing*, 5(2):249–262, 2017.

[74] Nikolas Herbst, André Bauer, Samuel Kounev, Giorgos Oikonomou, Erwin Van Eyk, George Kousiouris, Athanasia Evangelinou, Rouven Krebs, Tim Brecht, Cristina L. Abad, and Alexandru Iosup. Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(4):19:1–19:36, 2018.

[75] Ralph Hintemann. Data centers 2018. Efficiency gains are not enough: Data center energy consumption continues to rise significantly - Cloud computing boosts growth. Technical report, Borderstep Institute for Innovation and Sustainability, May 2020.

[76] Takahiro Hirofuchi, Adrien Lebre, and Laurent Pouilloux. Simgrid VM: virtual machine support for a simulation framework of distributed systems. *IEEE Transactions on Cloud Computing*, 6(1):221–234, 2018.

[77] Matthew Hutson. Artificial intelligence faces reproducibility crisis. *Science*, 359(6377):725–726, 2018.

[78] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H. J. Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.

[79] Alexandru Iosup, Omer Ozan Sonmez, and Dick H. J. Epema. Dgsim: Comparing grid resource management architectures through trace-based simulation. In Emilio Luque, Tomàs Margalef, and Domingo Benitez, editors, *Euro-Par 2008 - Parallel Processing, 14th International Euro-Par Conference, Las Palmas de Gran Canaria, Spain, August 26-29, 2008, Proceedings*, volume 5168 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2008.

[80] Alexandru Iosup, Alexandru Uta, Laurens Versluis, Georgios Andreadis, Erwin Van Eyk, Tim Hegeman, Sacheendra Talluri, Vincent van Beek, and Lucian Toader. Massivizing computer systems: A vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, pages 1224–1237. IEEE Computer Society, 2018.

[81] Alexandru Iosup, Laurens Versluis, Animesh Trivedi, Erwin Van Eyk, Lucian Toader, Vincent van Beek, Giulia Frascaria, Ahmed Musaafir, and Sacheendra Talluri. The atlarge vision on the design of distributed systems and ecosystems. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*, pages 1765–1776. IEEE, 2019.

[82] Raj Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[83] Bahman Javadi, Derrick Kondo, Alexandru Iosup, and Dick Epema. The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems. *JPDC*, 73(8), 2013.

[84] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. Analysis of large-scale multi-tenant GPU clusters for DNN training workloads. In *ATC*, 2019.

[85] JetBrains s.r.o. The Kotlin programming language. Press Kit., 2021. [Online; accessed 1-December-2021].

[86] Nicola Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561:163–166, 09 2018.

[87] Foued Jrad, Jie Tao, and Achim Streit. SLA based service brokering in intercloud environments. In *CLOSER 2012 - Proceedings of the 2nd International Conference on Cloud Computing and Services Science, Porto, Portugal, 18 - 21 April, 2012*, pages 76–81. SciTePress, 2012.

[88] George Kamiya and Oskar Kvarnström. Data centres and energy–from global headlines to local headaches. *International Energy Agency*, 2019. URL `https://www.iea.org/commentaries/data-centres-and-energy-from-global-headlines-to-local-headaches`. [Online; accessed 9-Dec-2021].

[89] James M Kaplan, William Forrest, and Noah Kindler. Revolutionizing data center energy efficiency. In *Uptime Institute Symposium*, 2008.

[90] Naim Kheir. *Systems Modeling and Computer Simulation*. Marcel Dekker, Inc, 2018.

[91] Dalibor Klusácek and Simon Tóth. On interactions among scheduling policies: Finding efficient queue setup using high-resolution simulations. In *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, volume 8632 of *Lecture Notes in Computer Science*, pages 138–149. Springer, 2014.

[92] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1–19. IEEE, 2019.

[93] Younggyun Koh, Rob C. Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, and Calton Pu. An analysis of performance interference effects in virtual environments. In *2007 IEEE International Symposium on Performance Analysis of Systems and Software, April 25-27, 2007, San Jose, California, USA, Proceedings*, pages 200–209. IEEE Computer Society, 2007.

[94] Andreas Kohne, Marc Spohr, Lars Nagel, and Olaf Spinczyk. Federatedcloudsim: a sla-aware federated cloud simulation framework. In Yehia Elkhatib and Stefan Walraven, editors, *Proceedings of the 2nd International Workshop on CrossCloud Systems, CCB@Middleware 2014, Bordeaux, France, December 8, 2014*, pages 3:1–3:5. ACM, 2014.

[95] Andreas Kohne, Damian Pasternak, Lars Nagel, and Olaf Spinczyk. Evaluation of sla-based decision strategies for VM scheduling in cloud data centers. In Yehia Elkhatib and Mohamed Faten Zhani, editors, *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms, CrossCloud@EuroSys 2016, London, United Kingdom, April 18-21, 2016*, pages 6:1–6:5. ACM, 2016.

[96] Greg Kroah-Hartman. The linux kernel driver interface. `https://www.kernel.org/doc/html/latest/process/stable-api-nonsense.html`, 2016. [Online; accessed 9-Dec-2021].

[97] Julian Kudritzki and Matt Stansberry. *Risk management for IT infrastructure*. Uptime Institute, 2016.

[98] Andy Lawrence. The gathering storm: Climate change and data center resiliency. Technical report, Uptime Institute, Nov 2020.

[99] Andy Lawrence. Extreme weather affects nearly half of data centers. *Uptime Institute Journal*, Mar 2021. URL `https://journal.uptimeinstitute.com/extreme-weather-affects-nearly-half-of-data-centers`. [Online; accessed 9-Dec-2021].

[100] Andy Lawrence. Annual outage analysis 2021. Technical report, Uptime Institute, Mar 2021.

[101] Hui Li. Realistic workload modeling and its performance impacts in large-scale escience grids. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):480–493, 2010.

[102] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 973–990. USENIX Association, 2019.

[103] Eric Loken and Andrew Gelman. Measurement error and the replication crisis. *Science*, 355(6325): 584–585, 2017.

[104] Kuan Lu, Thomas Röblitz, Ramin Yahyapour, Edwin Yaqub, and Constantinos Kotsokalis. Qos-aware sla-based advanced reservation of infrastructure as a service. In Costas Lambrinoudakis, Panagiotis Rizomiliotis, and Tomasz Wiktor Wlodarczyk, editors, *IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 288–295. IEEE Computer Society, 2011.

[105] Kuan Lu, Ramin Yahyapour, Philipp Wieder, Edwin Yaqub, Monir Abdullah, Bernd Schloer, and Constantinos Kotsokalis. Fault-tolerant service level agreement lifecycle management in clouds using actor system. *Future Generation Computer Systems*, 54:247–259, 2016.

[106] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[107] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin Van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. Opendc 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. In *21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2021, Melbourne, Australia, May 10-13, 2021*, pages 455–464. IEEE, 2021.

[108] Etienne Michon, Julien Gossa, Stéphane Genaud, Léo Unbekandt, and Vincent Kherbache. Schlouder: A broker for iaas clouds. *Future Generation Computer Systems*, 69:11–23, 2017.

[109] Brad L. Miller and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, 9(3), 1995.

[110] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, 1998. ISBN 978-0-262-63185-3.

[111] Jeffrey C. Mogul and John Wilkes. Nines are not enough: Meaningful metrics for clouds. In *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019, Bertinoro, Italy, May 13-15, 2019*, pages 136–141. ACM, 2019.

[112] Jeffrey C. Mogul, Rebecca Isaacs, and Brent Welch. Thinking about availability in large service infrastructures. In Alexandra Fedorova, Andrew Warfield, Ivan Beschastnikh, and Rachit Agarwal, editors, *Proceedings of the 16th Workshop on Hot Topics in Operating Systems, HotOS 2017, Whistler, BC, Canada, May 8-10, 2017*, pages 12–17. ACM, 2017.

[113] David Mytton. Hiding greenhouse gas emissions in the cloud. *Nature Climate Change*, 10(8):701–701, Aug 2020. ISSN 1758-6798.

[114] David Mytton. Data centre water consumption. *npj Clean Water*, 4(1):11, Feb 2021. ISSN 2059-7037. doi: 10.1038/s41545-021-00101-w.

[115] David Mytton. Renewable energy for data centers. Technical report, Uptime Institute, Feb 2021.

[116] Falak Nawaz, Omar Hussain, Farookh Khadeer Hussain, Naeem Khalid Janjua, Morteza Saberi, and Elizabeth Chang. Proactive management of SLA violations by capturing relevant external events in a cloud of things environment. *Future Generation Computer Systems*, 95:26–44, 2019.

[117] Alberto Nuñez, José Luis Vázquez-Poletti, Agustín C. Caminero, Gabriel G. Castañé, Jesús Carretero, and Ignacio Martín Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

[118] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. Groudsim: An event-based simulation framework for computational grids and clouds. In *Euro-Par 2010 Parallel Processing Workshops - HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31-September 3, 2010, Revised Selected Papers*, volume 6586 of *Lecture Notes in Computer Science*, pages 305–313. Springer, 2010.

[119] Anamika Pandey, Michael Brauer, Maureen L Cropper, Kalpana Balakrishnan, Prashant Mathur, Sagnik Dey, Burak Turkgulu, G Anil Kumar, Mukesh Khare, Gufran Beig, et al. Health and economic impact of air pollution in the states of india: the global burden of disease study 2019. *The Lancet Planetary Health*, 5(1):e25–e38, 2021.

[120] Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim von Kistowski, Ahmed Ali-Eldin, Cristina L. Abad, José Nelson Amaral, Petr Tuma, and Alexandru Iosup. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Transactions on Software Engineering*, 47(8):1528–1543, 2021.

[121] Harold Pashler and Eric-Jan Wagenmakers. Editors' introduction to the special section on replicability in psychological science. *Perspectives on Psychological Science*, 7:528 – 530, 2012.

[122] Pb7 Research and The METISfiles. De toekomst van de digitale economie. Technical report, Dutch Data Center Association, Jun 2020.

[123] J Sebastian Perera. International space station risk management. In *Risk Management Colloquium III, Palo Alto, California, September*, pages 17–19, 2002.

[124] D.V. Pym. Risk management. *PM Network*, 1(3):33–36, Aug 1987.

[125] V. Dinesh Reddy, Brian Setz, G. Subrahmanya V. R. K. Rao, G. R. Gangadharan, and Marco Aiello. Metrics for sustainable data centers. *IEEE Transactions on Sustainable Computing*, 2(3):290–303, 2017.

[126] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *Don't Focus on Big Data*, 2, 2017.

[127] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In Michael J. Carey and Steven Hand, editors, *ACM Symposium on Cloud Computing, SOCC '12, San Jose, CA, USA, October 14-17, 2012*, page 7. ACM, 2012.

[128] Merijn Rengers and Carola Houtekamer. Facebook vs. zeewolde: hoe lokale politici moeten beslissen over een landelijke kwestie. *NRC Handelsblad*, Nov 2021. URL `https://www.nrc.nl/nieuws/2021/11/26/facebook-vs-zeewolde-hoe-een-handvol-lokale-politici-plots-staat-voor-een-besluit-met-landelijke-implicaties`. [Online; accessed 9-Dec-2021].

[129] Katharine Ricke, Laurent Drouet, Ken Caldeira, and Massimo Tavoni. Country-level social cost of carbon. *Nature Climate Change*, 8(10):895–900, Oct 2018.

[130] Mark Seymour and Sherman Ikemoto. Design and management of data center effectiveness, risks and costs. In *2012 28th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, pages 64–68, 2012.

[131] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training NLP models: A concise overview. *Computing Research Repository*, abs/2004.08900, 2020. URL https://arxiv.org/abs/2004.08900.

[132] Prateek Sharma, Stephen Lee, Tian Guo, David E. Irwin, and Prashant J. Shenoy. Managing risk in a derivative iaas cloud. *IEEE Transactions on Parallel and Distributed Systems*, 29(8):1750–1765, 2018.

[133] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. United States Data Center Energy Usage Report. Technical report, Lawrence Berkeley National Laboratory, Jun 2016.

[134] Arman Shehabi, Sarah J Smith, Eric Masanet, and Jonathan Koomey. Data center growth in the united states: decoupling the demand for services from electricity use. *Environmental Research Letters*, 13 (12), Dec 2018.

[135] Siqi Shen, Vincent van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2015, Shenzhen, China, May 4-7, 2015*, pages 465–474. IEEE Computer Society, 2015.

[136] Sukhpal Singh, Inderveer Chana, and Rajkumar Buyya. STAR: sla-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*, 8(4):1040–1053, 2020.

[137] Statistics Netherlands (CBS). Electricity supplied to data centres, 2017-2019, Apr 2021. URL https://www.cbs.nl/en-gb/custom/2020/51/electricity-supplied-to-data-centres-2017-2019. [Online; accessed 9-Dec-2021].

[138] Stimular and Milieu Centraal. Notitie CO2-emissiefactoren stroom. Technical report, Milieu Centraal, Feb 2020.

[139] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/p19-1355.

[140] David Talby and Dror G. Feitelson. Improving and stabilizing parallel computer performance using adaptive backfilling. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA*. IEEE Computer Society, 2005.

[141] Ryan Tanaka, Rafael Ferreira da Silva, and Henri Casanova. Teaching parallel and distributed computing concepts in simulation with WRENCH. In *2019 IEEE/ACM Workshop on Education for High-Performance Computing, EduHPC@SC 2019, Denver, CO, USA, November 17, 2019*, pages 1–9. IEEE, 2019.

[142] Luis Tomás and Johan Tordsson. An autonomic approach to risk-aware data center overbooking. *IEEE Transactions on Cloud Computing*, 2(3):292–305, 2014.

[143] Rubén Trapero, Jolanda Modic, Miha Stopar, Ahmed Taha, and Neeraj Suri. A novel approach to manage cloud security SLA incidents. *Future Generation Computer Systems*, 72:193–205, 2017.

[144] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice Hall, 2016.

[145] Dylan Tweney. Amazon website goes down for 40 minutes, costing the company $5 million. *VentureBeat*, Oct 2013. URL https://venturebeat.com/2013/08/19/amazon-website-down. [Online; accessed 9-Dec-2021].

[146] United States Environmental Protection Agency. Green power partnership national top 100, Oct 2021. URL https://www.epa.gov/greenpower/green-power-partnership-national-top-100. [Online; accessed 9-Dec-2021].

[147] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan S. Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. Is big data performance reproducible in modern cloud networks? In *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 513–527. USENIX Association, 2020.

[148] Alexandru Uta, Kristian Laursen, Alexandru Iosup, Paul Melis, Damian Podareanu, and Valeriu Codreanu. Beneath the surface: An mri-like view into the life of a 21st-century datacenter. *login Usenix Mag.*, 45(3), 2020.

[149] Vincent van Beek, Jesse Donkervliet, Tim Hegeman, Stefan Hugtenburg, and Alexandru Iosup. Self-expressive management of business-critical workloads in virtualized datacenters. *Computer*, 48(7): 46–54, 2015.

[150] Vincent van Beek, Giorgos Oikonomou, and Alexandru Iosup. A CPU contention predictor for business-critical workloads in cloud datacenters. In *IEEE 4th International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2019, Umea, Sweden, June 16-20, 2019*, pages 56–61. IEEE, 2019.

[151] A. Vasan, A. Sivasubramaniam, V. Shimpi, T. Sivabalan, and R. Subbiah. Worth their watts? - an empirical study of datacenter servers. In *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–10, January 2010.

[152] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, Bordeaux, France, April 21-24, 2015*, pages 18:1–18:17. ACM, 2015.

[153] Akshat Verma and Gargi Dasgupta. Server workload analysis for power minimization using consolidation. In Geoffrey M. Voelker and Alec Wolman, editors, *2009 USENIX Annual Technical Conference, San Diego, CA, USA, June 14-19, 2009*. USENIX Association, 2009.

[154] Laurens Versluis, Roland Mathá, Sacheendra Talluri, Tim Hegeman, Radu Prodan, Ewa Deelman, and Alexandru Iosup. The workflow trace archive: Open-access data from public and private computing infrastructures. *IEEE Transactions on Parallel and Distributed Systems*, 31(9):2170–2184, 2020.

[155] David Vose. *Risk analysis: a quantitative guide*. John Wiley & Sons, 2008.

[156] Franc Weerwind and Stijn Steenbakkers. Ruimtelijke Strategie Datacenters: Routekaart 2030 voor de groei van datacenters in Nederland. Technical report, Ministry of the Interior and Kingdom Relations, Mar 2019.

[157] Montri Wiboonrat. Risk anatomy of data center power distribution systems. In *2008 IEEE International Conference on Sustainable Energy Technologies*, pages 674–679, 2008.

[158] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April 2010*, pages 446–452. IEEE Computer Society, 2010.

[159] Lonneke Wielders and Sanne Nusselder. Emissiekentallen elektriciteit. Technical report, CE Delft, Jan 2020.

[160] Franz Wilhelmstötter. *Jenetics Library User's Manual*, 2021. [Online; accessed 1-December-2021].

[161] Caesar Wu and Rajkumar Buyya. *Cloud Data Centers and Cost Modeling*. Morgan Kaufmann, 2015. ISBN 978-0-12-801413-4.

[162] Yiting Xia, Ying Zhang, Zhizhen Zhong, Guanqing Yan, Chiunlin Lim, Satyajeet Singh Ahuja, Soshant Bali, Alexander Nikolaidis, Kimia Ghobadi, and Manya Ghobadi. A social network under social distancing: Risk-driven backbone management during COVID-19 and beyond. In James Mickens and Renata Teixeira, editors, *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021*, pages 217–231. USENIX Association, 2021.

[163] Chee Shin Yeo and Rajkumar Buyya. Integrated risk analysis for a commercial computing service in utility computing. *Journal of Grid Computing*, 7(1):1–24, 2009.

[164] Liang Yu, Tao Jiang, Yang Cao, Shiyong Yang, and Zhiqiang Wang. Risk management in internet data center operations under smart grid environment. In *IEEE Third International Conference on Smart Grid Communications, SmartGridComm 2012, Tainan, Taiwan, November 5-8, 2012*, pages 384–388. IEEE, 2012.

[165] Liang Yu, Tao Jiang, Yang Cao, and Qian Zhang. Risk-constrained operation for internet data centers in deregulated electricity markets. *IEEE Transactions on Parallel and Distributed Systems*, 25(5):1306–1316, 2014.

[166] Qi Zhang, Ludmila Cherkasova, Guy Mathews, Wayne Greene, and Evgenia Smirni. R-capriccio: A capacity planning and anomaly detection tool for enterprise services with live workloads. In *Middleware 2007, ACM/IFIP/USENIX 8th International Middleware Conference, Newport Beach, CA, USA, November 26-30, 2007, Proceedings*, volume 4834, pages 244–265, 2007.

[167] Zhou Zhou, Jemal H. Abawajy, Morshed U. Chowdhury, Zhigang Hu, Keqin Li, Hongbing Cheng, Abdulhameed A. Al Elaiwi, and Fangmin Li. Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms. *Future Generation Computer Systems*, 86:836–850, 2018.