Vrije Universiteit Amsterdam

Bachelor Thesis

Sunfish: Enabling Predictive Analytics for Datacenters Through Digital Twinning

---

Author: Mateusz Kwiatkowski (2805533)

1st supervisor:	Prof. Dr. Ir. Alexandru Iosup
daily supervisor:	drs. Dante Niewenhuis
2nd reader:	drs. Matthijs Jansen

A thesis submitted in fulfillment of the requirements
for the VU Bachelor of Science degree in Computer Science

July 9, 2026

Contents

Chapter 1

Introduction

Presently, computer and network systems play a crucial part in the digital industry. The transport, education and government sectors largely depend on digital services, which are hosted in datacenters [1]. To address the recent rise in demand for computation, due to the advancements in Artificial Intelligence, managers expand datacenters with new components and more heterogeneous architectures (*e.g.,* GPUs, NPUs) [2]. However, in return datacenter complexity increases significantly. To make better operational decisions despite the massive scale, promising technologies arise such as Datacenter Digital Twins [3].

Datacenters house large volume of computers for processing and storage of data from various organizations and fields of activity. Over 3 million jobs in the Netherlands directly depend on cloud services, which are hosted in datacenters. Since many public services continue to move online (*e.g.,* online administration and taxation, education), the fraction of Dutch professionals who depend on the cloud for work will exceed 35% by 2025 [1].

In the modern Artificial Intelligence (AI) economy datacenters need diverse and scalable server architectures, because inference-based workloads require more heterogeneous server components (GPUs, TPUs, NPUs *etc.*) to perform well. Nowadays, datacenter operators try to meet customer expectations by adding more specialized hardware [2], at a cost of increased system complexity. In return, operating a modern datacenter warehouse with thousands of diversified servers presents a difficult challenge that requires fast and well-informed decisions from on-site engineers.

The computational requirements of AI are expected to increase in the future [3]. Datacenter complexity will continue to grow, and it will become more difficult to manage. Future servers will include even more specialized hardware, which, while improving datacenter performance, will exhibit behaviour that is harder to predict. Already the rapid expansion of datacenters has increased the presence of service failures across all cloud services [4]. Preventing failure-caused outages in advance could help datacenter operators reduce operational costs, as over 20% of all reported outages amount to more than 1 million US$ [5].

In short, the high computational demand of AI and the end of Dennard's scaling have resulted in the rise of larger and more heterogeneous datacenter architectures [2]. Both

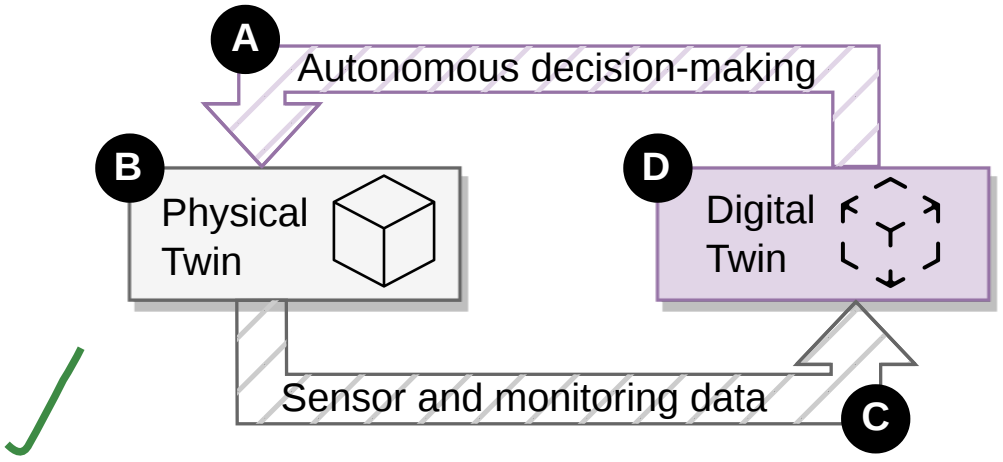

Figure 1.1: Elements of the digital twin ecosystem [6] include: the insights and decisions coming from the digital twin (Ⓐ), the physical infrastructure (Ⓑ), the data coming from the physical twin telemetry (Ⓒ), and the digital counterpart to the physical twin (Ⓓ).

events create a need for more careful datacenter management to tackle the unprecedented complexity and ensure availability of all cloud services. To address this new problem a concept of a datacenter Digital Twin (DT) was proposed [3].

1.1 Context

A DT is a virtual model of an intended or actual real-world system that serves as its counterpart for purposes such as simulation, integration, testing, monitoring and maintenance. The digital twin replicates the physical system to predict failures, prescribe real-time actions for mitigating unexpected events, observing and evaluating the behaviour of the system [7].

Most modern DT usages are related to prognostics and system health management [8]. For example, in aerospace engineering, the DT analyzes operational data (*e.g.,* temperature, vibration) to predict when a airplane component is likely to fail. The DT can reliably manage the health of the physical entity by detecting fatigue cracks on aircraft wings or damage to the wind turbine blades [9]. This allows maintenance to be scheduled proactively, reducing unplanned downtime and preventing catastrophic failures. Forecasting future maintenance and managing the physical health of an object or facility are the prime purpose of many DTs used in practice [10].

The first mention of a DT dates back to 2003, when Dr. Michael Grieves of Dassault Systémes introduced the 3 core components of a DT: the virtual entity, physical entity and the two-way connection (see Figure 1.1). Due to insufficient technological foundations, little work is available on DTs between 2003 and 2018, and it is only with the rapid growth of cloud computing, Internet-of-Things (IoT) and Big Data analytics that DTs have re-emerged. Today, research is focused on bridging the gap between the long-established

foundations of DTs and new, novel applications in academia and industry, such as the Datacenter Digital Twin (DCDT) [8, 3].

A DCDT mirrors the structure, context and behaviour of a datacenter [3]. The foundation to any digital twin is good monitoring and sensing capabilities in the physical entity. Datacenters, meet this requirement easily because they already connect hundreds of monitoring sensors. With hundreds of gigabytes of useful information coming from distributed IoT sensors inside the warehouse, we can gain insight into failure patterns, energy usage, heat dissipation *etc.* What remains challenging is to connect the physical and virtual spaces with a bi-directional connection to use the monitoring insights and data analysis results for autonomous decision-making. Crucial to DCDT operation are predictive capabilities and the continuous interaction with the real-world datacenter.

There already exist DCDT deployments. For example, ExaDigiT [11] is a framework for digital twin development of supercomputers. It has been demonstrated at the Frontier supercomputer and it facilitates virtual prototyping and system optimization.

Nonetheless, existing DCDT's are still very limited in their capabilities as the definition and scope of a DCDT concept is shallow and unclear. After all, only recently did the hardware capabilities needed to continuously simulate a datacenter become available [8]. Many DCDT frameworks still lack critical data analysis components, fault detection mechanisms, profiling techniques *etc.* [12], rendering them unusable in large-scale systems. Such limitations gravely reduce the applicability of DCDT's in real world scenarios [1]. DCDT's are urgently needed, because datacenters exhibit hundreds unexpected events every day,such as *e.g.,* service failures or hardware faults. Downtime, which is the result of failures, disturbs the users and produces unfulfilled Service Level Agreements (SLA) [4]. However, predicting datacenter behaviour quickly and reliably is a non-trivial problem that remains insufficiently unaddressed in the existing DCDT architectures [12, 3] and deployments [11].

1.2 Problem statement

We envision DCDT's as systems indispensable in future datacenters, actively interacting with the real-world facility, lowering operational costs and predicting hardware failure and software faults. In this work, we address the lack of a unified DCDT system model and the absence of predictive capabilities in existing DCDT system designs. We argue that the current state-of-the-art DCDT's lack sufficient predictive capabilities that are essential to real-time facility management of a modern datacenter. A DT without predictive capabilities cannot maintain the health of the datacenter effectively. We posit that including holistic predictive analysis in DCDT design can aid in efficient datacenter management and prevent missing SLA's. We propose that digital twinning can be enhanced by integrating predictive analytics through Operational Data Analysis (ODA).

1.3 Research Questions

Main Research Question: How to enable predictive analytics in datacenters through digital twinning?

We divide the problem of designing a predictive DCDT into three research questions:

RQ_1 *How to assess the current state-of-the-art of digital twinning for datacenters?*
There is currently a lack of a unified system model of what constitutes a DCDT, and the differences between existing DCDT deployments. It is necessary that we establish a common model of a DCDT in the research community. We must develop a holistic DCDT model that factors in the necessary components of a DT. This is very challenging, because the DCDT system model must address many kinds of operational and technical requirements, compatible with the existing background on DTs.

RQ_2 *How to design a DCDT system model using discrete-event simulation and predictive data analysis?*
Existing DCDT frameworks lack the necessary predictive capabilities to prevent unplanned behaviour in datacenters. In this work, we aim to explore the design space of a predictive DCDT and the different design trade-offs. Through discrete-event simulation, we aim provide the foundation for the system model to interact with a physical datacenter. This is a very challenging task, because there are many functional and non-functional requirements of a DCDT that need careful consideration. The architecture must comply with the generic DT model and address the non-trivial challenges in operating a modern datacenter.

RQ_3 *How to evaluate and validate a DCDT model in relation to system requirements*?
To understand the operation of the proposed system and whether it meets its design goals we need to measure it's performance. This is a challenging and non-trivial task that requires a careful design of a set of experiments that realistically show datacenter digital twin workings.

1.4 Research Methodology

To answer RQ_1 we conduct a literature review as proposed by *Kitchenham et al.* [13] along with the guidance of the supervisor. Firstly, we determine the right review method. Secondly, we identify the various works related to DCDT's using various search strings (*e.g.,* "Datacenter Digital Twinning", "ICT Virtual Twin"). To search for the results we use the digital libraries of Google Scholar, DBLP, ACM Digital Library, IEEExplore, Springer *etc.* Thirdly, we select work relevant to our research and organize the details of each article. A potential outcome of this could be a system model for DCDT's. We envision the literature review can supply us with potential use-cases for the predictive DCDT. Based

on the found use-cases, we formulate the functional and non-functional requirements for the predictive DCDT reference architecture.

To answer RQ_2 we closely follow the *AtLarge Design Process* [14] under the guidance of the supervisor, and propose a simulation-based DCDT system that meets the requirements listed as a part of RQ_1. Firstly, following the literature review, we list the functional and non-functional requirements of a predictive DCDT. We specify the pragmatic and innovative design possibilities to include in the reference architecture. The designed system builds upon the OpenDC platform for datacenter simulation [15], extending it with predictive analysis capabilities. Lastly, we ensure that the design is scientific and testable and can be evaluated with comprehensive experiments.

To answer RQ_3 we implement a prototype of the designed reference architecture. We design comprehensive experiments that evaluate and validate the prototype based on the reference architecture. We first gather a set of questions worth asking about the performance and impact of the predictive DCDT and then set out to answer them with the prototype. We define the correct experiment setup(s) and perform the experiments on a specified hardware, considering different usage scenarios.

1.5 Thesis Contributions

1. **Conceptual**:

   C_1 We conduct a comprehensive literature review and detailed analysis of existing works on digital twinning in the scientific research community. We collect and organize the DCDT's characteristics and based on our findings we propose a unified system model of the design space.

   C_2 We propose the design of *Sunfish* (SF), a discrete-event DCDT for reliable and timely failure prediction in datacenters. SF includes a set of novel system components which leverage ODA and discrete-event simulation.

   C_3 We evaluate SF using a novel experimentation technique and datacenter workload traces from the industry. We design a method to evaluate DCDTs without expensive and costly real-world experimentation. We conduct a set of experiments and analyse the results.

2. **Technical:**

   C_1 We prototype SF following the established DT design principles using discrete-event simulation and ODA. We include the code as an Open Science artifact and ensure the prototype remains accessible to the broader scientific community including exhaustive project documentation.

   C_2 We provide the experiment setup, validation and evaluation of SF for predicting datacenter failures in real-time as an Open Science artifact.

1.6 Plagiarism Declaration

I hereby declare that this thesis is my own independent work and writing. The thesis does not contain any material copied from other sources (person, Internet, or AI), and has not been submitted for assessment elsewhere. I acknowledge that the usage of material from other works or paraphrase of such material without proper citations or credit will be treated as plagiarism. I declare that this thesis is free from AI generated content and has been written without the help of any AI tools. In order to adhere to the strictest restrictions on AI-usage in higher education, this work follows the Berkley School of Law Artificial Intelligence Policy, as stated in https://www.law.berkeley.edu/wp-content/uploads/2026/05/AI-Final-Policy-26.pdf.

1.7 Reference Fraud

I hereby declare that all the references in this thesis refer to genuine scientific work published in peer-reviewed journals or other sources of reliable and safe online information (*e.g.,* Wikipedia articles) and have been used in accordance to the article authors' wishes. Additionally, under the guidance of the supervisor this work adheres to the strictest rules for referencing and to prove the originality of all references, each BibTeX citation contains a `note` field with the following comment: *This BibTeX citation comes from:* followed by the URL leading directly to the citation source. In case of citations not formatted in BibTeX, the same format follows but with adequate reference-style name (*e.g.,* APA, Chicago, MLA).

1.8 Societal Impact

Any program that is difficult to understand and reason about is sure to accumulate technical debt. However, sometimes large-scale systems can be complex and hard to comprehend inherently. In such scenario, software that can aid system management is necessary. Computer Systems, more complex now than ever before, must remain accessible to be beneficial to the digital society. This work addresses the four grand societal challenges related to this goal: (1) manageability (2) responsibility (3) sustainability (4) usability [1]. SF addresses (1) directly by making large-scale datacenter management easier. We address (2) by ensuring our work adheres to the Findability, Accessibility, Interoperability and Reusability (FAIR) principles of Open Science. Moreover, in this thesis we try to make DCDT systems more understandable to the broader scientific community by providing a unified system model. Additionally, we contribute to responsible software design by adhering to best software engineering practices in the design of the prototype. (3) is addressed indirectly, as the consequences of the insights provided by a holistic, ODA powered DCDT can help datacenter managers make decisions that are more sustainable in the future. We contribute to (4) by helping predict unexpected failures and lowering operational costs, ensuring datacenters can continue to be usable in the future. We believe this work has a strong societal

impact due to addressing the four grand societal challenges described by Iosup *et al.* and we hope through this work we can advance the scientific research community towards a more sustainable future.

1.9 Open Science

Abiding the FAIR data principles, the entire source code of the prototype and related work has been made available at the `https://git.denounce.ai/opendc.git` repository. The reuse and reproduction of experiments is explained in a detailed guide at the root of the repository, along with the necessary dependencies and experimental setup.

1.10 Thesis Structure

The remainder of the thesis is structured as depicted in Figure 1.2. In Chapter 2, we describe the relevant background information. In Chapter 3, we present the design of DCDT. In Chapter 5 we evaluate a prototype of the system and validate it against the set of functional and non-functional requirements. In Chapter 6 we conclude the thesis with a summary of contributions and potential future work.

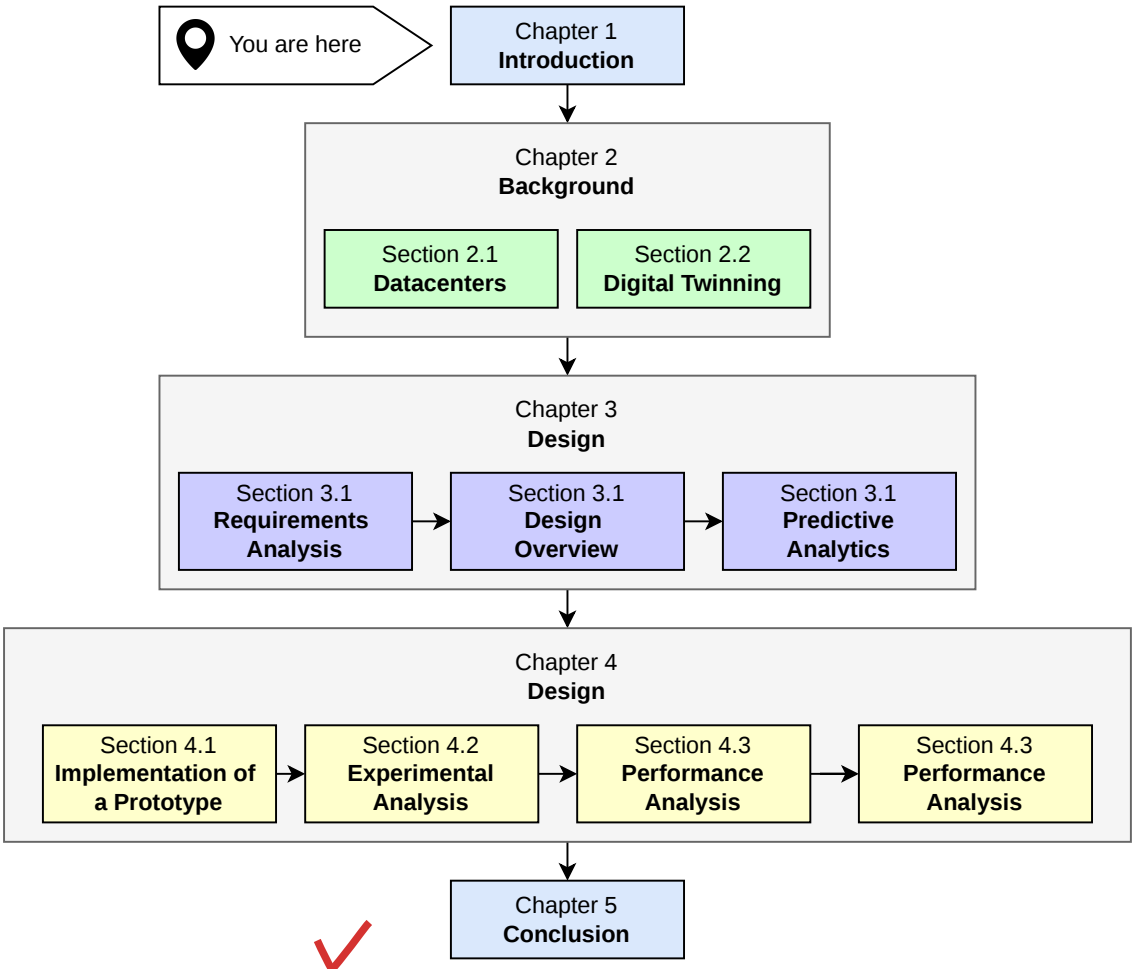

Figure 1.2: Structure of this thesis, with suggested reading flows.

Chapter 2

Background

The contribution in this chapter is three-fold:

- C_1 We provide a brief overview on datacenter simulation (Section 2.1.1), compute failures (Section 2.1.2), and digital twinning (Section 2.2.1).
- C_2 We survey the state-of-the-art concerning datacenter digital twinning (Section 2.2.2).
- C_3 We construct a system model for existing datacenter digital twins (Section 2.3)

2.1 Datacenters

In this section we provide a short background on datacenter simulation and hardware failures. We find it useful to provide a brief introduction to both topics so as to ensure reader's fullest understanding of subsequent chapters. Since datacenters are important building blocks of the digital society, reliable warehouse management is a key priority for datacenter operators. Incorrect management decisions can lead to missed SLAs [1] and even large financial penalties [5]. However, efficient and timely management is a difficult challenge, because datacenters are extremely complex facilities. To help datacenter operators, the scientific community proposes to simulate datacenters to make more informed decisions.

2.1.1 Datacenter Simulation

Simulation empowers better design, testing and management of datacenters [15]. A well-designed datacenter simulator can estimate a months-long workload in a few minutes or hours. To simulate is to "imitate of real-world process or system over time, enabling the study of, and experimentation with the internal interactions of complex systems" [28] In this project we only consider *discrete-event simulation*.

Project	Environment	Stakeholders	Highlighted Features	GUI
CloudSim [16]	Cloud, Fog [17], Edge	Research	VC*, N, S, E, WF† [18], FD†, EXP†, CM, PI†	✓† [19]
SimGrid [20]	Grid, P2P, Cloud [21]	Research, Edu [22]	VC*† [23], N*, S, E*, WF* [24]	✓† [24]
DGSim [25]	Grid	Research	WF, F, EXP	✗
GroudSim [26]	Grid, Cloud	Research	WF, CM, F	✗
iCanCloud [27]	Cloud	Research	VC, N*, S, CM	✓*
OpenDC [15]	Cloud	Research, Edu.	VC*, N, S, E*, CM, FS*, ML, WF, F*, PI, EXP*	✓*

Table 2.1: Comparison of selected datacenter simulators. **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 interface; **Tools:** EXP = Experiment automation; **Support:** ✓ = Yes, ✗ = No; † = extension, not integrated; ⋆ = advanced, carefully calibrated feature. Author: Mastenbroek *et al.* [15]

Discrete-event simulation represents system operations as a sequence of events over time, with an assumption that no changes occur between the events. Due to the scale and complexity of datacenters, most simulators use discrete-event simulation [15].

Alternatives to simulation include real-world experimentation and mathematical analysis. However, experimentation *in situ* is unsustainable, expensive and difficult to reproduce and mathematical analysis is not scalable to modern datacenters [15]. Therefore, in this project we only consider simulation as a foundation for the DCDT. There exist many datacenter simulation tools, for example DGSim [25], CloudSim [16], SimGrid [20], iCanCloud [27], GroudSim [26] and OpenDC [15]. See Table 2.1 for a comparison of selected datacenter simulators, combined by Mastenbroek *et al.* [15]. In order to narrow the scope of the project, we only consider OpenDC as a simulator for the digital twin design. We decided to use OpenDC, because we find it important for a simulator to model hardware failures well. *Failure models* are a carefully calibrated, advanced feature of OpenDC. Further details about OpenDC can be referred to in the linked literature [15].

2.1.2 Compute Failures

A failure is defined as "an event that makes a system fail to operate according to its specifications" [29]. We distinguish 2 failure types: 1. software failures 2. hardware failures. For example, a hypervisor crash a software failure. Each Virtual Machine (VM) within the crashed hypervisor is killed as a result. An example of a hardware failure is a host crash,

where a single server stops working (*e.g.,* as a result of a disk fault, or faulty power supply cable). Hardware and software failures in datacenters result in service downtime, missed SLA and user inconvenience [4, 29]. OpenDC uses the notion of a *failure model* to simulate failures, alongside *failure traces*. A failure model consists of two statistical distributions: (1) to model service unavailability (2) to model service availability. A failure trace is defined by an interval, duration, and intensity of several failures, which are later looped throughout the simulated workload (source `opendc.org`). In summary OpenDC enables experimentation with failures that enables insights that are not provided by other state-of-the-art software. However, the fidelity of failure modeling inside a datacenter simulation is still insufficient to predict in failures in real-time, as they happen in a physical datacenter. Since a datacenter simulator is quite different from a digital twin, we cannot use the same computation methods from simulation to predict real-time failures. Digital twinning is an improvement upon pure simulation.

2.2 Digital Twinning

In this section we explore how the datacenter management can be improved using a novel modelling technique, digital twinning. We present the generic, field-agnostic DT definition and investigate how *datacenter* digital twinning applies the definition in practice.

2.2.1 What is Digital Twinning?

"A *digital twin* is a set of virtual information constructs that mimics the structure, context and behaviour of a natural, engineered or social system, is dynamically updated with data from its physical twin, has predictive capability, and informs decisions that realize value" [30]. A crucial characteristic that differentiates digital twinning from simulation and statistical modelling is the *digital thread*: a bi-directional channel that enables continuous interaction between the virtual and physical entities. The longer the DT is working, the more accurate its predictions, because a holistic twin aggregates historical patterns together with up-to-date monitoring data. A generic DT architecture is depicted in Figure 2.1 Section 1 from Tao *et al.* [8].

Digital twinning has only recently become feasible because of the developments in High Performance Computing (HPC). Between 2003 and 2011 the compute needed to run a DT was simply not present. As such, while the concept existed, the hardware did not catch up yet. However, in the last decade, multicore computing paradigms and the advent of GPU computing has finally enabled computation needed to run digital twins. As a result, digital twins have become more relevant today than 10 years ago [8].

A crucial part any of any DT is *predictive modelling*, which drives actionable insights [30] (see Figure 2.2). Predictive modelling uses statistics to predict outcomes. When deployed commercially, for example in datacenters, predictive modelling is often referred to as predictive analytics [31]. Almost any statistical model can be used for prediction purposes, but nowadays predictive analysis is synonymous with machine learning. A primary exam-

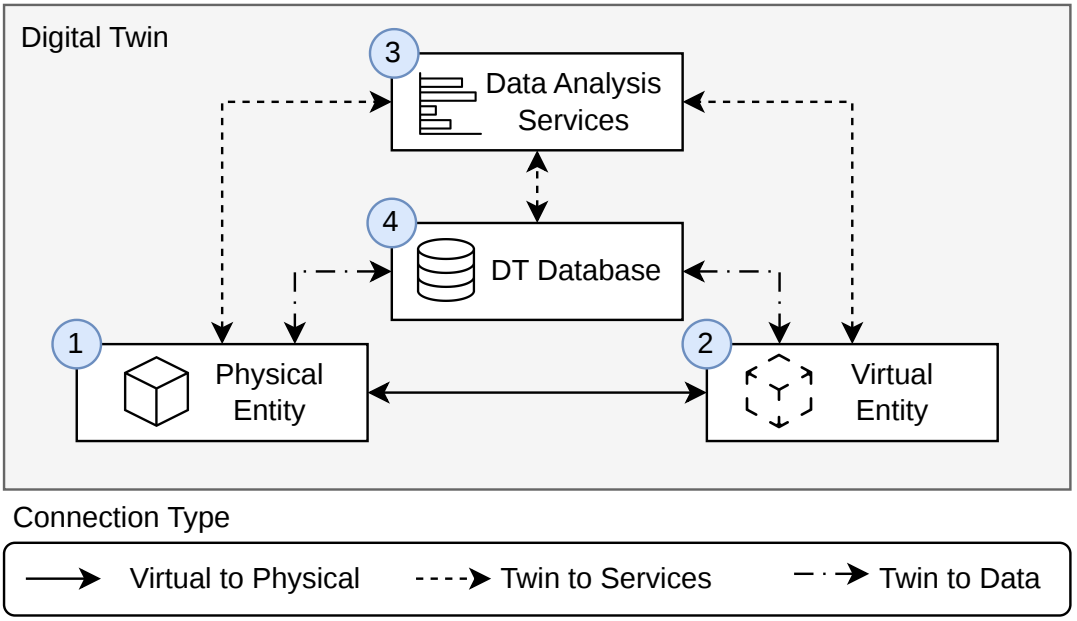

Figure 2.1: A basic framework for the DT. Four core elements of a DT are defined: The physical entity (❶) and the simulated virtual twin (❷). A service for out-of-band data analytics (❸) and a persistent storage of historical data (❹) are crucial to the DT because they are necessary to gain meaningful monitoring insights. Adapted from Tao *et al.* [8].

ple of popular analysis type is linear regression. However, any modelling technique, *e.g.*, *discrete-event simulation* can be used to make the predictions.

Predictive analysis belongs to a larger domain of ODA. ODA is the "use of operational data instrumentation, analysis, integration, and archiving, towards effective design, commissioning, and optimization of datacenter operations" [32].

Operational analytics are present at all layers of a datacenter. A reference architecture, proposed by *Suman et al.* describes the kind of operational analysis performed at each distributed system layer. For example, at the resource manager layer, ODA should enable workload modeling capabilities, which use predictive analysis to determine submitted user job properties [12]. There exist several ODA frameworks, for example OMNI [32], Wintermute DCDB [33], ExaMon [34], AutoDiagn [35] and ODAbler [12].

A major limitation of predictive analytics is that history cannot always predict the future. Using historical data to predict outcomes works only under the assumption that there are certain long lasting patterns in the system. Additionally, no matter how extensive is the training data, there is always the possibility of new variables that have not been considered or even defined, yet are critical to the outcome of the prediction [31].

2.2.2 Digital Twins for Datacenters

In this section, we survey the work related to datacenter digital twinning. We summarize our results in Table 2.2 to compare and contrast the features of existing datacenter digital

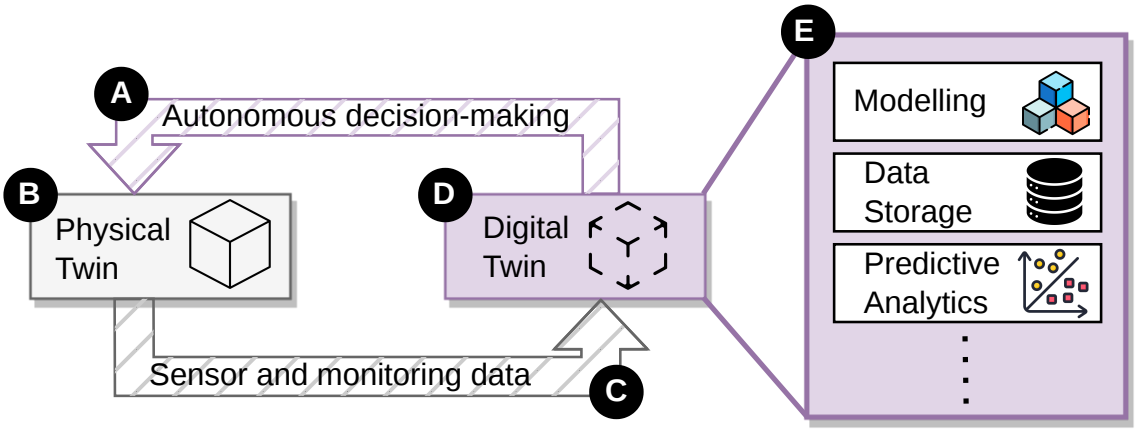

Figure 2.2: Datacenter Digital Twin diagram. There are 5 core elements to any Digital Twin: Ⓐ The Digital → Physical Twin link, Ⓑ the Physical Twin (*e.g.,* the datacenter), Ⓒ the Physical → Digital Twin link, Ⓓ the Digital Twin, Ⓔ the features necessary to any Digital Twin.

twins. We select only the digital twins that adhere closest to the National Academy of Sciences, Engineering, and Medicine (NASEM) definition [30].

ExaDigiT [11] is an open-source framework for developing digital twins of supercomputers. It consists of 3 modules: (1) resource allocator and power simulator (2) thermal cooling model (3) augmented reality 3D model of the supercomputer. ExaDigiT has been used at the Frontier supercomputer at the Oak Ridge National Laboratory in the USA, successfully predicting potential energy losses at the supercomputer. Brewer *et al.* include alongside the framework architecture an open-source artifact and a set of extensive verification and validation experiments. The authors differentiate between different digital twins within ExaDigiT, such as (1) descriptive twin (2) informative twin (3) predictive twin (4) comprehensive twin (5) autonomous twin that together form the system. The *predictive twin* leverages data driven operational analytics to create Machine Learning (ML) models. Authors argue that alongside simulation, ML models should also have a significant role for modeling system workloads in *e.g.,* application fingerprinting. Within the *autonomous twin* the authors use Reinforcement Learning (RL) to train agents that can be used to make control decisions in order to optimize different processes. In order to model the cooling system the authors use the Modelica software, and to predict energy power draw they coded a Python script. The authors provide a intuitive way to interact with the system using a visual dashboard, and an advanced augmented reality model. The authors posit that the best way to address the 3V's of data (velocity, volume and variety) is to use augmented reality coupled with dashboards.

SmartDC [36] is a digital twin solution for optimization of power consumption in datacenters. Specifically, Zhang *et al.* propose that using AI enhanced modeling paired with digital twinning can help make dynamic adjustments to the datacenter cooling subsystem. SmartDC has been proven to ensure efficient energy-saving rate of a China Telecom dat-

Project	Simulation Technique	Focus	Stakeholders	Modelling Capability
ExaDigiT [11]	CFD/HT, AI/ML	Energy Loss Prediction, Heat Modelling	HPC Engineers and Operators	3D*, CH‡, VP*, PE‡, RA, SE†
SmartDC [36]	CFD/HT, BIM, AI/ML	Heat Modelling, PUE optimization	Cloud Datacenter Engineers	CH†, PE, 3D*
DyTwin [37]	Gaussian Process Regression, AI/ML	Anomaly Detection	Cloud Datacenter Operators	A*, FD, VP*, SE†
ChatTwin [38]	?	Digital Twin Definition Language	Cloud Datacenter Engineers	3D*
Reducio [39]	POD, Gaussian Process Modelling (ML)	Heat Modelling	Edge and Hyper-scale Datacenter Operators	CH‡, 3D*, SE
NetGraph [40]	Graphs	Network Management	Cloud Datacenter Operators	VP*, RA*, N*, SE†
Kalibre [41]	CFD/HT, ML	Heat Modelling	Cloud Datacenter Engineers	CH‡, 3D*

Table 2.2: Comparison of selected datacenter digital twins. **Modelling capability:** 3D = Visualizations; CH = Cooling/Heat, PE = Power/Energy Consumption, A = Anomaly Detection, N = Network Modelling, SE = Scenario Exploration, VP = Virtual Prototyping, FD = Federation, RA = Resource Allocation; **Data Analytics**: ‡ = Predictive Analysis; $\star$ = Descriptive Analysis, † = Prescriptive Analysis.

acenter at 41%. However, the main purpose of SmartDC is not to continuously interact with the facility, but to provide additional training data for a more accurate, ML solution. The digital twin is designed to provide extra datasets for training AI models.

DyTwin [37] is an adaptive digital twin with visualization and anomaly detection features. The system, developed at Hewlett Packard (HP) is a precursor to the vision on datacenter digital twinning published by Athavale *et al.* [3]. DyTwin is the only system capable of failure detection in datacenters. Moreover, it is the only system to incorporate the idea of federation into the concept of digital twinning. DyTwin is designed to interact not only with the physical facility, but also other federated digital twins. Taheri *et al.* show that DyTwin can effectively detect 100% of CPU usage anomalies (*i.e.,* irregularities that affect CPU utilization, ranging from 5% to 60%).

ChatTwin [38] is an AI and Large Language Models (LLMS) powered system that enables easy deployment and configuration of digital twins for datacenters. It is a *text-to-3D* approach to building digital twins of datacenters. ChatTwin is the only work in the field that does not share the simulation technique used to construct the digital twin based on ChatTwin's configuration. Li *et al.* provide a thorough set of experiments to show ChatTwin generates the JavaScript Object Notation (JSON) DCDT configuration efficiently, but do not share the final 3D visualization results.

Reducio [39] is a system designed to further optimize the Computational Fluid Dynamics (CFD) approach to datacenter modeling. Instead of using plain CFD the authors focus on Proper Orthogonal Decomposition (POD) approaches to approximate the heat transfer.

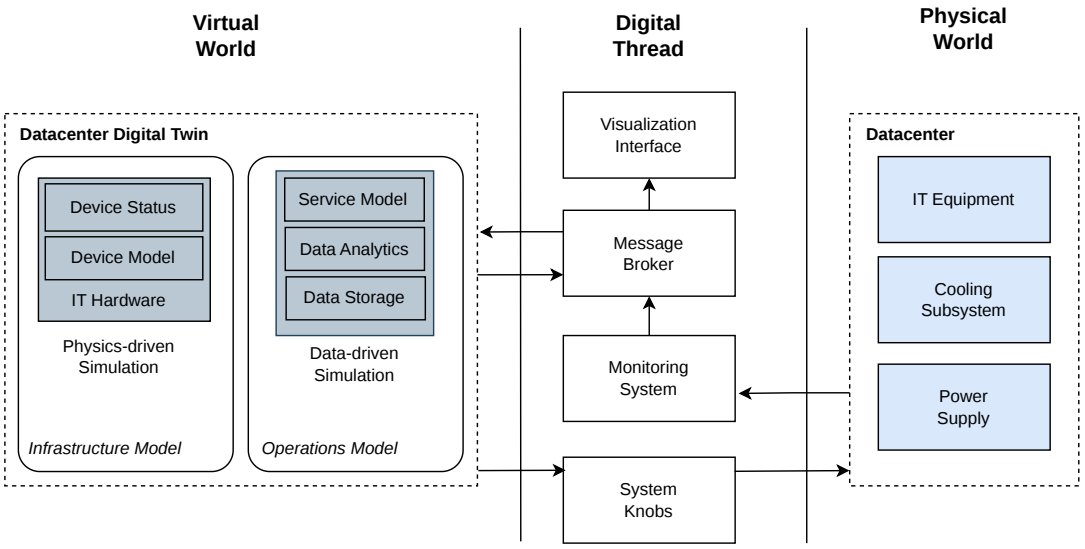

Figure 2.3: A generic system model for datacenter digital twin deployments. The design of DyTwin [37] indirectly incorporates in its architecture a "virtual-to-virtual" digital thread between different digital twins. Zhao *et al.* likewise present key elements to the digital thread in their architecture [42]. We add the *Digital Thread* to our model explicitly.

Using the POD technique, the authors are able to model the datacenter more efficiently, achieving sub 1 degree Celsius Mean Absolute Error (MAE) in temperature prediction. Moreover, their model outperforms the CFD approaches. Cao *et al.* evaluate their system on an edge datacenter with 70 CPU-only server racks (see Figure 3 in [39]), and on a hyper-scale datacenter with thousands of servers (see Figure 4 in [39]). Their results show promising gains in physics-based datacenter modelling over the conventional approaches.

NetGraph [40], designed by Huawei Technologies and China Mobile. NetGraph is the only system in our literature review that focuses on network management (see Table 2.2). Moreover, NetGraph employs a unique modelling technique, combining device, network and service models using graph theory. The authors evaluate their system in a Huawei datacenter with over 50000 server racks. With over 20 million connections in the network graphs, the system is a prime example of datacenter digital twin potential.

Kalibre [41] is a system designed by Wang *et al.* in order to overcome the cons of CFD. To lessen the computational overhead, Wang *et al.* propose to use a knowledge-based neural surrogate to calibrate the different CFD models. Kalibre takes the best of both ML and CFD approaches and achieves sub 1 degree Celsius MAE, similarly to Reducio [39].

2.3 System Model for Datacenter Digital Twinning

Chapter 3

Design

Our contribution in this chapter is three-fold:

- C_1 We analyze the requirements for *Sunfish* (Section 3.1).
- C_2 We propose a conceptual design for *Sunfish*'s architecture (Section 3.2)
- C_3 We describe how *Sunfish* enables predictive analytics through digital twinning **(??)**

3.1 Requirements Analysis

In this section we determine the requirements that should be fulfilled by *Sunfish*. We present here the stakeholders identified by our literature survey (see Section 2.2.2) and the relevant use-cases. Afterwards, we list the functional and non-functional requirements for *Sunfish*.

3.1.1 Stakeholders

We identify four main stakeholders of a predictive datacenter digital twin:

S1 – Datacenter Managers
Responsible for maintenance and operation of the warehouse, operators manage the datacenter daily. They interact with the servers, bring downed hosts up and ensure customers' services run smoothly at all times. Datacenter operators need to ensure different SLAs are met, energy costs are balanced and carbon emission quota is maintained.

S2 – Datacenter Technicians
The term datacenter engineers encompasses datacenter architects and technicians alike. From the moment the datacenter layout is determined, to the physical process of booting the server racks for the first time, datacenter engineers help build and

maintain the datacenter. They must continuously adapt to changing requirements and ensure everything goes smoothly [3].

S3 – Scientists and Academia
Digital twinning generates unprecedented amount of data. To bring valuable insights, the ingested metrics must be effectively analyzed. Scientists can draw conclusions from the monitoring data and in some deployments already benefit from the voluminous output of DCDTs [36].

S4 – Customers and Users
Digital twins are already used to visualize both facilities and human beings alike for the benefit of users and customers. Cloud and HPC users are not directly interacting with the system, but pay for services hosted in the datacenter and are one of the primary stakeholders of digital twinning. Not only through 3D datacenter visualizations, but also from the continuously generated metrics can users and customers benefit from DCDTs.

3.1.2 Use-cases

Based on the identified stakeholders we list 6 potential use-cases for a predictive datacenter digital twin:

UC1 – Energy Optimization
Predicting and modeling energy optimization is curcial for DCDTs. It is the main use-case of some existing systems [11]. Effective energy optimization, including the adjustments to the consumed energy type is a crucial use-case of DCDTs.

UC2 – Failure Management
Predictive maintenance of both hardware and software failures alike, by simulating the possible failure distribution of a running workload, can lower downtime, terminated tasks and ensure SLAs are not missed [3].

UC3 – Heat Modelling
Heat modeling is the primary use case of existing DCDTs (see Table 2.2). Correct thermal management of the warehouse can optimize cooling strategies, leading to lower bills and maintenance costs.

UC4 – Network Traffic Modelling
Congestion management and traffic routing can effectively benefit from digital twinning. Detecting bottlenecks, adjusting datacenter protocols and calibrating switches and interconnects is already a important use-case for DCDT [40].

UC5 – Virtual Prototyping
Digital twinning can be used to provide insight into the system before changes are made. Using a DCDT, the operators can change, model and shape the datacenter to

estimate their effect. Virtual prototyping encompasses interacting with an existing datacenter model, or with a proof-of-concept, not yet constructed warehouse.

UC6 – Monitoring and Visualization
3D visualizations and dashboards are of the utmost importance to all stakeholders, due to the insights they provide. With real-time data ingestion and the two-way feedback loop, DCDTs can empower descriptive analytics. This use-case already shapes many existing systems (see Table 2.2).

3.1.3 Functional Requirements

Based on a subset of the above use-cases, we formulate the functional and non-functional requirements for *Sunfish*:

FR1 – The system should be able to handle workloads of arbitrary size.
Existing systems range from Cloud through the Edge to HPC digital twins. Therefore, *Sunfish* must support workloads similar in length and type to the commercial setting. Without **FR1**, *Sunfish* will be incomplete, and like the majority of the Table 2.2 systems, its use-case will be niche. **FR1** is necessary to avoid overly-specializing the DCDT.

FR2 – The system should support failure detection.
Failures are of crucial concern to datacenter operators. The system detect and report failures to datacenter operators. Without **FR2**, the system will not be able to help datacenter operators meet SLAs. Including **FR2** is necessary to ensure that failures, which which can cause financial penalties, are detected in a timely manner so as to meet the different SLAs.

FR3 – The system should be capable of long-term and short-term data storage.
FR3 is necessary for **FR4** and **FR5**. Without **FR3**, the system cannot support ODA techniques. A system cannot be considered a DT without insights stemming from accurate data analytics [30]. NASEM digital twin definition requires both real-time insights, and guidance based on historical-patterns. Therefore, it is imperative to ensure **FR3**.

FR4 – The system should support descriptive data analytics.
There are many types of data analytics present is existing deployments [12]. In order to scope down the project, we only select several use-cases for SF. Out of the 4 prime analytics type, we find in the literature survey the predictive analytics to be the most urgently needed, and descriptive analytics to be the most prevailing type of data processing. Therefore, in order to achieve performance on par with previous systems, and to ensure our system meets the official NASEM definition, **FR4** is needed.

FR5 – The system should enable predictive data analytics.
The system should help future researchers incorporate predictive data analytics engines, regardless of the statistical modelling technique. This is our novel contribution to the scientific field of DCDTs. Without **FR5**, we miss the aim of our entire project.

FR6 – The system should be able to process arbitrary amounts of telemetry.
The size of the AI economy is expected to grow [3]. As a result, both current and future datacenters will generate huge amount of data. Our system must be capable of ingesting and processing different metrics regardless of the volume of incoming messages. Without **FR6**, we hinder the adoption of *Sunfish* in modern and future datacenters.

FR7 – The model should be capable of clear data visualization. The system must have a user-friendly, visual interface for data analytics, and support them in real-time. Without **FR7**, we exclude important stakeholders such as customers and users from the potential benefactors of our system.

3.1.4 Non-functional Requirements

In addition to the functional requirements, we also present non-functional requirements for *Sunfish*:

NFR1 – The system should enable real-time insights and visualizations.
The system must work in real-time, without significant delay. The system must support datacenter operators with insights at fine-grained granularity, so that insights derived from data analysis remain accurate upon reception by datacenter operators. Without **NFR1**, *Sunfish*'s insights will not be timely, and will be useless to datacenter operators.

NFR2 – The system should log the ingestion and processing of metrics.
The system should provide a log of the current network traffic to and from the digital twin.

NFR3 – The system should adhere to modern software development standards.
The system should follow modern coding principles and guidelines to ensure reproducibility and usefulness for future work. Without **NFR3**, it will be difficult for current and future digital twin developers to include predictive analytics using *Sunfish* in their deployments.

NFR4 – The system should provide insights at varying levels of confidence.
With the huge amount of telemetry data incoming from the physical twin, the digital counterpart must be able to filter and pre-process the telemetry. Without **NFR4**, the system will present overwhelming amount of information to its users, rendering it unusable.

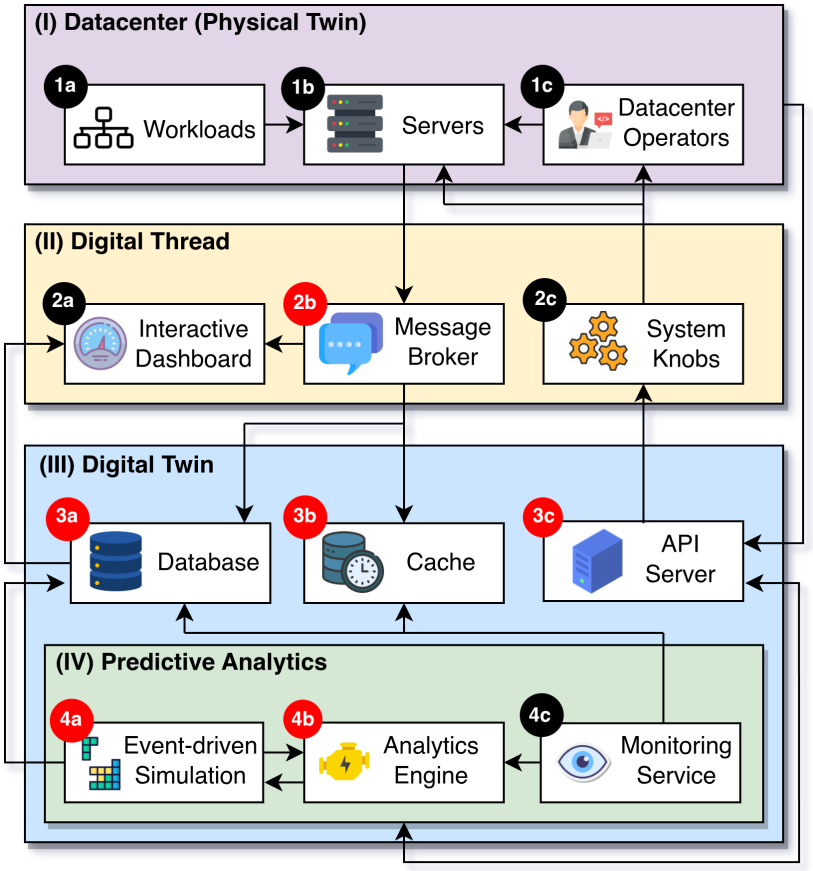

Figure 3.1: The predictive datacenter digital twin reference architecture. We call the system *Sunfish*. The architecture was designed with the *AtLarge Design Process* [14] over several iterations in the past months.

3.2 Design of *Sunfish*

As a result of the *AtLarge Design Process* [14] designed a reference architecture for a predictive datacenter digital twin. Figure 3.1 encompasses 4 main elements: (I) physical datacenter (II) digital thread (III) digital twin (IV) predictive analytics.

The physical datacenter (I) encompasses 3 core elements important to digital twinning. Workloads (1a) include the hardware requirements of each datacenter job and the submission time. They are executed on the datacenter compute (1b), which is controlled partly by the Datacenter Operators (1c). Component (1c), while seemingly unimportant, is crucial to the digital twin design. We envision DCDTs as systems that contain a human-in-the-loop, which can control and overwrite the system's autonomous decisions. Datacenter Operators (1c) interact with both the Servers (1b), and have the ability to overwrite the potential autonomous digital twin decisions, stemming from component (2c), the System Knobs.

The Digital Thread (II) is a novel contribution from Chapter 2. It separates the physical

world from the virtual world, and contains components that do not belong to either twin, or belong to both twins. It contains the Interactive Dashboard (2a), the Message Broker (2b), and System Knobs (2c).

To fulfill the functional requirements of our system, we incorporate element (2a), the Interactive Dashboard to our system. This dashboard ingests data coming directly from the Message Broker (2b) and from the long-term storage (3a), the Database. The Interactive Dashboard (2a) allows datacenter operators to see the telemetry data arrive to the digital twin in real time.

The Message Broker (2b) is a crucial component to the reference architecture, because it facilities the physical twin → virtual twin connection. A low-latency, high-throughput message broker partly meets our functional requirements to enable arbitrary amounts of telemetry data transfer. We elaborate on the specific components that make up the message broker (2b) in Section 3.2.1.

The System Knobs (2c) represent the different cogs within the datacenter software and hardware (1b) that can be adjusted during runtime (*e.g.,* to optimize Power Usage Effectiveness (PUE), change cooling strategy, allocate compute resources). For example, System Knobs (2c) within the datacenter scheduler can be tuned to schedule jobs on Servers (1b) that are least likely to experience future downtime. The autonomous actions of the digital twin (the tuning of the System Knobs (2c)) can be further adjusted by Datacenter Operators (1c).

In our design, we explicitly differentiate between the physical and virtual space by including the Digital Twin (III) in a separate box. The Digital Twin (III) constitutes of the long-term storage (3a), short-term storage (3b), the API Server (3c) and the Predictive Analytics (IV) module.

Long-term storage, namely a Database (3a) is crucial to enable real-time visualizations (2a), and to model long-term system behaviour. This fulfills the functional requirements for our system, as we set out to differentiate between in-band data analytics, and out-of-band data analysis. The data is consumed by the Database (3a) directly from the Message Broker (2b). In-between, simple data pre-processing can take place.

Short-term storage, in the form of a Cache (3b) is essential for in-band data analytics, "on-the-go". The Cache (3b) retains the data consumed from the Message Broker (2b), and for a short period of time keeps a copy. The data analytics performed on the Database (3a) dataset and the Cache (3b) adhere to the use-cases for predictive (long-term) and descriptive (long-term, short-term) data analytics.

The Message Broker (2b) enables one-way connection (physical twin → digital twin). The API Server (3c) enables the digital twin → link. The API Server (3c) communicates directly with the System Knobs (2c) in order to take meaningful action, based on the (predictive) insights generated by the Digital Twin (III). Additionally, the Physical Twin (III) can query the API Server (3c) for one-shot requests (*e.g.,* to create a new datacenter prototype configuration, to request special data analysis). Moreover, the Datacenter Operators (1c) can query the API Server (3c) for extra insights, when necessary.

The Predictive Analytics (IV) module is an extensible part of the reference architecture,

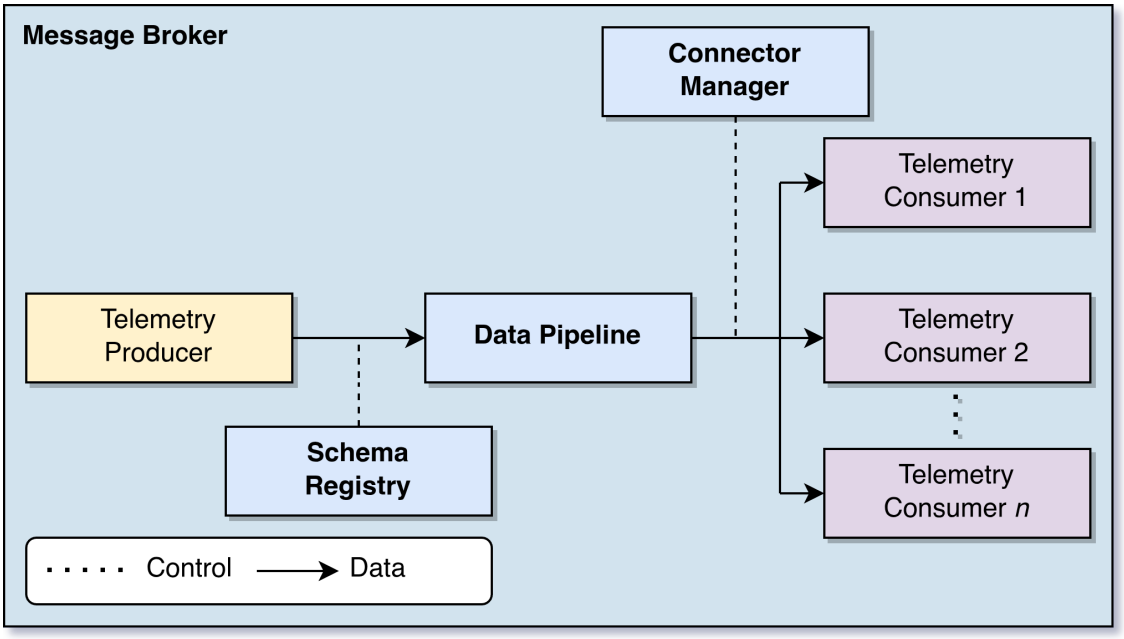

Figure 3.2: The detailed view of the Message Broker (2b) from Figure 3.1.

enabling different kinds of predictive analysis. In our design, to facilitate meaningful predictions we incorporate an Event-driven Simulator (4a), Analytics Engine (4b), and a Monitoring Service (4c).

We chose an Event-driven Simulator (4a) as a core element of predictive analytics. For the rationale behind this decision see Section 2.1. The predictions from the Event-driven Simulator (4a) communicate directly with the Database (3a) to store the simulation results, and with the Analytics Engine (4b).

The Analytics Engine (4b) performs descriptive and predictive data analytics on the results from the Event-driven Simulator (4a) and the telemetry data collected from the physical datacenter (stored in (3a, 3b)). This component is highly extensible to accommodate the different types of analytics (*e.g.,* ML, AI, statistical methods). The results of the data analysis are propagated to the API Server (3c) to be queued for retrieval by the physical twin.

The Monitoring Service (4c) serves as a separate thread to watch over the collected telemetry in real-time. It facilities detection of irregularities in collected data, which adheres to the functional requirements set out for the system. Any discrepancies are communicated to the Analytics Engine (4b) for further analysis, and potential insights.

3.2.1 Message Broker

The Message Broker (2b) is a component is slightly more complex, and necessities a separate diagram. In Figure 3.2 we present the composite elements that make up the Message Broker. In particular, the *schema registry* and the *connector manager* play a crucial part

in fulfilling the functional requirements of our system. The *schema registry* allows the telemetry producer to submit *any* data format for sending (and storing) the telemetry data.The registry is responsible for detecting what kind of format is the data sent in, and automatically adjusting the schema within the *data pipeline*. The connector manager is responsible for joining multiple distinct services to a single data pipeline. In the reference architecture, the consumers would constitute the Database (3a), the Cache (3b), and the Interactive Dashboard (2a). What is remarkable about the connector manager is the ability to swiftly connect more consumers to the system. This way, the predictive digital twin can facilitate multiple different types of analytics engines or techniques. Additionally, the setup is Figure 3.2 is currently standard industry practice for large software deployments.

Chapter 4

Implementation

In this chapter we describe the implementation of SF. The main contribution of this chapter towards answering *RQ3* is the prototype of SF. After reading one should understand the technical decisions, choice of tools and modifications to existing software necessary for evaluation of SF in Chapter 5.

Any complex system is more than the sum of its parts [43]. To understand why SF it is crucial to provide a holistic view on the prototype. Therefore, the rest of the chapter is structured in a top-down approach: Section 4.1 presents the rationale for using the specific software packages, Section 4.2 shows the flow of data within the system, and Section 4.3 details the different modifications and new software extensions to `OpenDC`. Lastly, Section 4.4 carefully explains the design decisions behind the major Python modules.

4.1 Overview

At the onset of the project, we decided SF will use only state-of-the-art software, deployed in the industry or evaluated in peer-reviewed scientific publications. The mapping of software packages used onto the reference architecture can be seen in Figure 4.1. In order to facilitate visualizations and interactive dashboards, we decided to use `Grafana` (2a) [44]. To enable the flow of data into the DT, we use `Kafka` (2b) [45]. To store the in-band data we use a `Redis` (3b) [46] cache, and for out-of-band data we use a `PostgreSQL`(3a) [47]. To enable predictive analytics, we chose a discrete-event simulator, `OpenDC`(4a) [48]. The `Analytics Engine` (4b), `Monitoring Service` (4c), and `HTTP Server` (3c) are described in detail in Section 4.4.

`Grafana` (2a)is a state-of-the-art industry tool to visualize dashboards. We posit it is crucial to include a user-friendly User Interface (UI). A number of previous publications on DTs find dashboards important [37, 12, 49]. We chose `Grafana` (2a) instead of other software packages because of its seamless integration with `PostgreSQL` (3a). `Grafana` (2a) provides good separation of concerns and compartmentalization as it does not store the displayed metrics itself. Instead, it queries the `PostgreSQL` (3a) database in real-time [44], unlike *e.g.,* `InfluxDB`. Good alternatives to `Grafana` (2a) are `Kibana` [50], `Prometheus` [51],

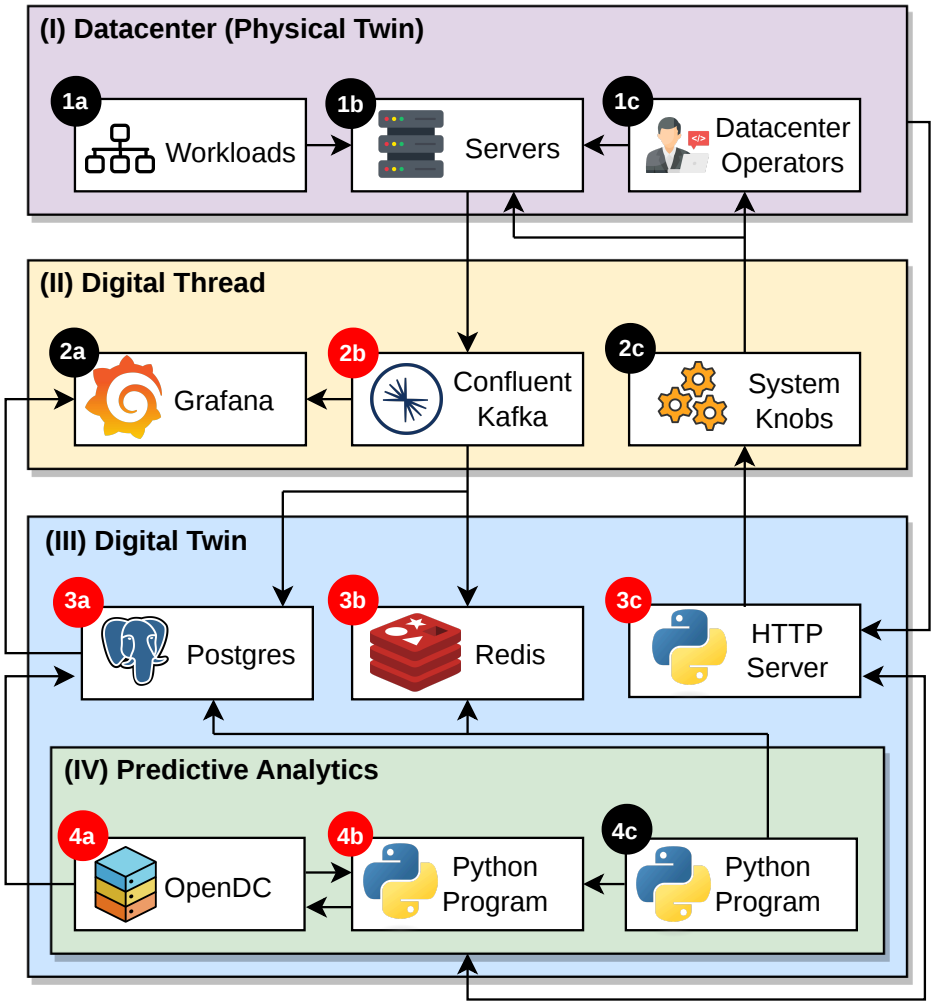

Figure 4.1: The prototype and its components based on the architecture. The time-series data flows first to the `Grafana` (2a) dashboard, `PostgreSQL` (3a) database and `Redis` (3b) cache as advised in [37].

and `Graphite` [52].

`Kafka` (2b), particularly `Kafka` developed by Confluent [45] is a battle-tested message broker that provides versatile capabilities to transfer huge volumes of data with little latency, in real-time. We decided to use `Confluent Kafka` instead of `Kafka` developed by the Apache Foundation, because of it's masterful connector system allowing to easily add sources and sinks (*e.g.,* `PostgreSQL` (3a) sink, `Redis` (3b) sink, `OpenDC` source (4a)). Additionally, as opposed to `Apache Kafka`, `Confluent Kafka` comes equipped with a `Schema Registry`. The `Schema Registry` is a important component that allows the storage of database and cache schemas for easy retrieval. With `Schema Registry`, we ensure that the data stored in `PostgreSQL` (3a) tables and in `Redis` (3b) streams contains the exact same schema. Moreover, `Schema Registry` is compatible with versatile data interchange

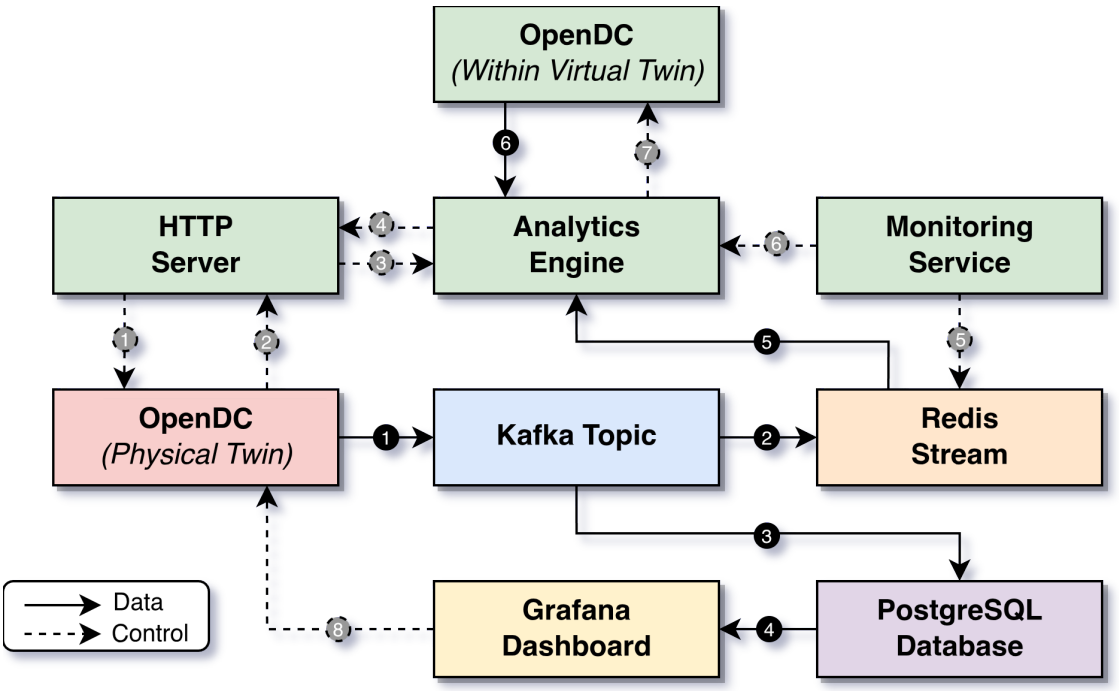

Figure 4.2: The data flow within SF.

formats, such as `ProtoBuf` [53] (see Listing 4.1).

`Redis` (3b), is a key value store that provides efficient store and retrieval operations [46]. In particular, `Redis` (3b) is capable of storing *streams* – append only logs which allow for fast and quick query of large volumes of data. `Redis` (3b) is the industry leader in key value caching. The only alternative to `Redis` (3b) is `memcached` [54], which does not provide the capability to integrate with `Kafka` (2b).

`PostgreSQL` (3a) is a database management system, necessary to store large volumes of out-of-band data coming from the physical datacenter. The `PostgreSQL` (3a) server provides a simple and straightforward interface to query the data via `psql`. Importantly, to adhere to the single responsibility principle, `PostgreSQL` (3a) does not provide any UI. Additionally, there exist many integrations between `PostgreSQL` (3a) and other software, including `Kafka` (2b). The many alternatives to `PostgreSQL` (3a) are listed in [47]. An alternative used in previous work is `InfluxDB` [12].

Lastly, to enable predictive analytics we use a state-of-the-art discrete-event simulator, `OpenDC`(4a) [48]. `OpenDC` (4a) is a leading software package capable of modeling complex datacenter phenomena and workloads (*e.g.,* failures, workflows, machine learning). For a specific overview of advantages of `OpenDC` (4a) and a thorough comparison with other alternatives, see Table 2.1.

4.2 Data Flow

In this section we describe the data flow within Figure 4.1 using a separate diagram. Efficient data flow is of utmost importance to DTs. In Figure 4.2 we present the moving of data within SF. In the diagram whenever we refer to *control*, we mean small, one-in-a-while data packets that contain either instructions, insights or small amount of data. Whenever we refer to *data*, we mean the hundreds of thousands of metrics exported with minimal latency from the operating datacenter (`OpenDC`). We describe the flow of data through a timeline.

First, the datacenter informs the DT of an upcoming workload (②). This packet contains the datacenter topology and the upcoming workload tasks (workload trace, collected from *e.g.,* BitBrains). The DT stores this data locally, and passes it forward to the `Analytics Engine` (③). The analytics engine queries the `OpenDC` simulator to run a simulation of what might happen in the datacenter under such workload (⑦). `OpenDC` returns the potential results to the `Analytics Engine` directly (❻). This data for the purposes of the prototype is stored in the `.parquet` files. In real-world scenario, this data flow (❻) would be connected to a separate `Kafka` topic.

In the meantime, the datacenter executes the workload. The datacenter continuously sends the metrics into the `Kafka` topic (❶) (*i.e.,* the datacenter is the *producer*). As this happens in real-time, `Redis` ingests the data from the `Kafka` topic (❷) alongside `PostgreSQL` (❸) (*i.e.,* `Redis` and `PostgreSQL` are the *consumers*). At the same time, `Grafana` polls the `PostgreSQL` database for incoming metrics (❹). Each time `PostgreSQL` ingests the data from the `Kafka` topic (❸), at the same time `Grafana` updates its real-time dashboard (⑧). This provides real-time feedback to datacenter operators (⑧).

Simultaneously, the `Monitoring Service` checks the in-band data within `Redis`, in real-time (⑤). Should anything unusual occur, the `Monitoring Service` notifies the `Analytics Engine` to perform necessary analysis (❻). Then, the `Analytics Engine`, ingests the data from the `Redis` stream (❺) and analyzes it for further insights. All insights generated in this way, are sent to the `HTTP Server` (❹) to communicate to the system knobs within the datacenter and to the datacenter operators (①).

Of significant importance are data flows (❷) and (❸). Due to the massive volume of data incoming from the physical datacenter, the `Redis` sink (❷) has been configured to filter out relevant packets. `Kafka` comes with excellent capability to efficiently compare data packets against a condition and filter our packets that are of no use to the `Analytics Engine` (see Listing 4.3). On the contrary, the `PostgreSQL` sink (❸) contains all metrics collected by the datacenter sensors (see Listing 4.2). This setup achieves excellent abstraction level, because only the most important metrics are forwarded to the `Analytics Engine`, with the majority of packets being filtered out.

```
syntax = "proto3";

package proto;

option java_package = "org.opendc.common";
option java_outer_classname = "ProtobufMetrics";

message ProtoExport {
  string timestamp = 1;
  string host_id = 2;
  int32 tasksactive = 3;
  double cpuutilization = 4;
  double energyusage = 5;
  double uptime = 6;
  double downtime = 7;
}
```

Listing 4.1: The Redis and PostgreSQL schema (`schema.proto` file). All the large volume data packets that travel within the system follow this format (*i.e.,* ❶ through ❻). The communication of small datapackets takes place using the Hyper-text Transfer Protocol (HTTP) protocol.

4.3 Extensions to OpenDC

`OpenDC` is a state-of-the-art datacenter simulator. In order to turn it into a DT, we have made several design decisions and extensions.

1. **`SmartScheduler`**
   The new `SmartScheduler` is a scheduling mechanism capable of incorporating the insights from the DT into its scheduling decisions. It relies on the functionality of the `HTTPClient` to poll the DT at each scheduling step for potential insights. For example, if DT sends to the datacenter a list of hosts likely to fail in the future, the `SmartScheduler` acts as *system knobs* to enforce the DT insights (*i.e.,* it can be mapped to ❷c from Figure 4.1).

2. **`KafkaMonitor`**
   The datacenter acts as the *producer* of metrics, ingested by the `Kafka` topic (see Figure 4.2). We equip `OpenDC` with a new `ComputeMonitor` capable of exporting data directly into a `Kafka` topic. The `KafkaMonitor` class implements the generic `ComputeMonitor` interface, therefore the new extension follows the other metric exporting options in `OpenDC` (*i.e.,* follows the implementation for exporting metrics into `.parquet files`).

3. **`HTTPClient`**
   The `HTTPClient` offers the necessary functionality to communicate between the DT and the datacenter. We decided to use the HTTP protocol for short, one-off communications between the DT and the datacenter, as is common industry practice.

```
name=postgresql-sink
connector.class=io.confluent.connect.
    jdbc.JdbcSinkConnector
tasks.max=1
# Crucial for Schema Registry
key.converter=org.apache.kafka.
    connect.storage.StringConverter
value.converter=io.confluent.connect.
    protobuf.ProtobufConverter
value.converter.schema.registry.url=
    http://localhost:8081
topics=postgres_topic
connection.url=jdbc:postgresql
    ://127.0.0.1:5432/opendc
connection.user=matt
connection.password=admin
auto.create=true
auto.evolve=true
insert.mode=insert
table.name.format=postgres_topic
```

Listing 4.2: Kafka PostgreSQL sink.

```
name=kafka-connect-redis
topics=postgres_topic
tasks.max=1
connector.class=com.redis.kafka.
    connect.RedisSinkConnector
key.converter=org.apache.kafka.
    connect.storage.StringConverter
value.converter=io.confluent.connect.
    protobuf.ProtobufConverter
value.converter.schema.registry.url=
    http://localhost:8081
transforms=downtime
transforms.downtime.type=io.confluent
    .connect.transforms.Filter$Value
transforms.downtime.filter.condition=
    $[?(@.downtime == '0.0')]
transforms.downtime.filter.type=
    exclude
transforms.downtime.missing.or.null.
    behavior=fail
```

Listing 4.3: Kafka Redis sink.

4.4 Python Modules

`Analytics Engine`, `HTTPServer`, `MonitoringService` are Python modules we prototyped for the reference architecture evaluation. It is important to note that we do not evaluate the performance of SF due to the substantial overheads of the Python interpreter. `Kafka`, `Redis` and `PostgreSQL` enable milisecond-latency and huge throughput within the system, however what stops us from a performance evaluation is the high cost of Python interpretation. For future work, we envision a system that implements the reference architecture in a compiled language, *e.g.,* `C` or `C++`.

1. **`AnalyticsEngine`**
   The `AnalyticsEngine` module is necessary in order to encapsulate the logic of data preprocessing and analysis from monitoring. This component can contain capabilities for different statistical metrics, subject to DTs focus. In SF `AnalyticsEngine` continuously checks whether the incoming datacenter sensor readings exceed different thresholds. For example, the `AnalyticsEngine` is capable of calculating a similarity score S between potential failure distributions and the true failure distribution.

2. **`HTTPServer`**
   The `HTTPServer` is crucial for interrupting the operation of the datacenter in order to adjust its operation or offer insights. It maintains a python `Queue` structure. The `Queue` *producer* is the `AnalyticsEngine` (④). The *consumer* is the `HTTPClient` within `OpenDC` (*i.e.,* the real datacenter, (❷), (❶)). Both the consumer and producer interact with the `Queue` via the network and different HTTP endpoints of `HTTPServer`.

3. **`MonitoringService`**

   The `MonitoringService` is responsible for interacting with the `Redis` key value store. It contains a `while True` Python loop which contains the function call to fetch the latest changes to the `Redis` stream. Upon update, the `MonitoringService` informs the `AnalyticsEngine` that new data is awaiting `AnalyticsEngine` (⑥).

Chapter 5

Evaluation

The contribution of this chapter is two-fold:

C1 We provide a novel method for evaluating datacenter digital twins in Section 5.1.

C2 We provide a set of exhaustive experiments to evaluate *Sunfish* in Sections 5.2 and 5.3.

Our findings indicate:

F1 Digital twinning can be used for failure detection to the benefit of datacenter operators. *Sunfish* is able to effectively differentiate between large failures and insignificant downtime.

F2 *Sunfish* is capable of dynamic adjustments to the scheduling policy of the datacenter, during workload runtime.

F3 If supplied with a state-of-the-art predictive analytics engine, *Sunfish* is capable of lowering the number of terminated tasks.

5.1 Experimental Setup

In this section we describe the technical setup used to evaluate *Sunfish*. However, Figure 3.1 assumes the system designer is capable of connecting the digital twin directly to the datacenter. This raises a problem, we cannot just go and test digital twins on large systems, because we do not have large systems at hand. Moreover, real-world experimentation is costly and unsustainable in the long run [15]. To overcome this problem, we present a novel datacenter digital twin method capable of evaluating a DCDT without the physical datacenter. Figure 5.1 details our approach.

In this approach, we replace the real-world datacenter with *another* instance of the event-driven simulator from Figure 3.1. In our implementation, this is a second `OpenDC` process (see Figure 4.1). The "physical twin" simulator is capable of fully replacing the real-world

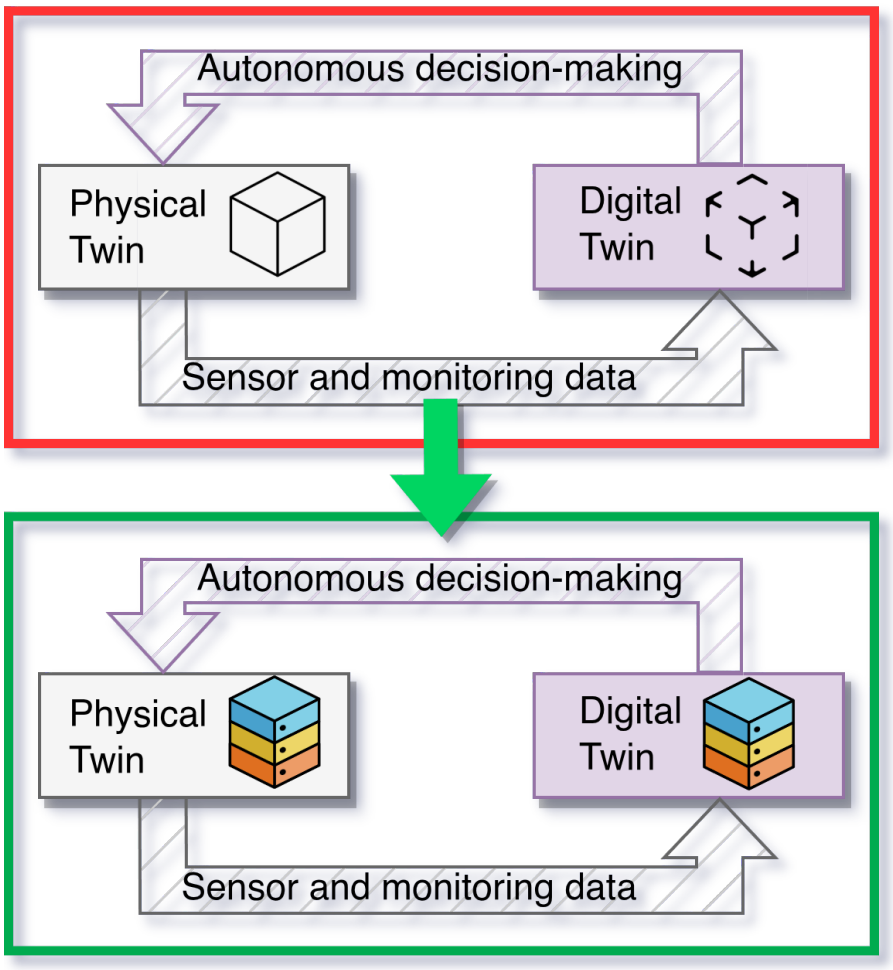

Figure 5.1: A novel evaluation method which solves the issue of real-world experimentation, which is unsustainable and costly [15].

facility, and allows for reproducible experimentation. For a detailed overview of the data flow within Figure 5.1, see Figure 4.2.

The technical setup used for all experiment adheres to the `OpenDC` documentation (see Mastenbroek *et al.* [15]). The workload trace used for all experiments comes from BitBrains [55]. In the experiments we model a Dutch SURF datacenter for scientific computing. The cluster, SURF-SARA, contains 277 hosts, each with 128GB of RAM and 16 processing cores running at maximum 2.1GHz [49]. The scheduling policy for all experiments is the `FilterScheduler` which considers the RAM and CPU capacity for choosing hosts to run tasks on. This scheduling policy is also used by `SmartScheduler`, as outlined in Section 4.4, albeit with modifications to enable the system knobs to take autonomous action.

In all experiments we use either *failure traces* or *failure models*. For a brief explanation on the differences between the two, consult Section 2.1.2. In the experiments we use traces

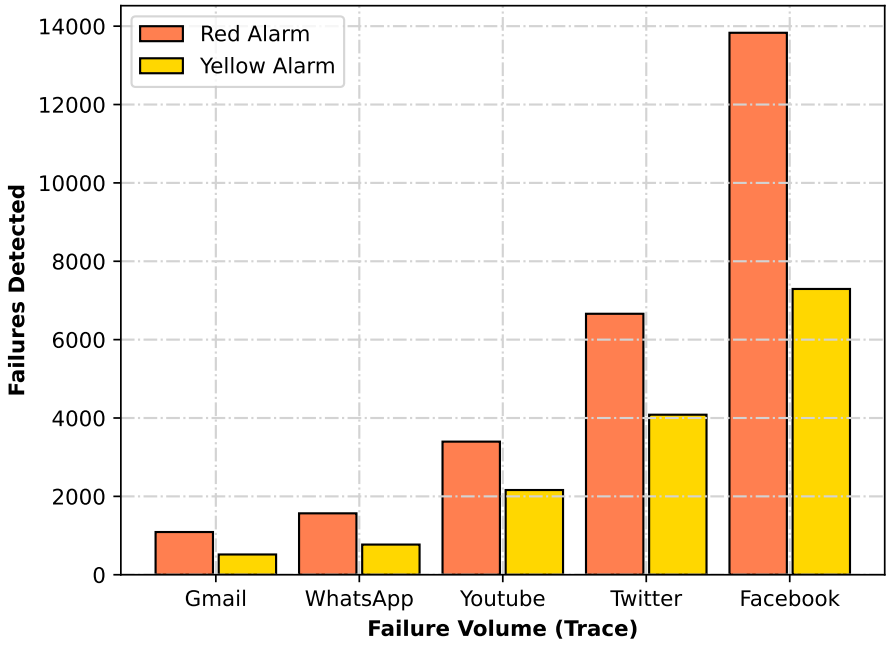

Figure 5.2: The results of Experiment 1. ■ Red Alarms signify 90% of acceptable failure threshold was reached. ■ Yellow Alarms signify 80% of the threshold was reached.

from the archive developed by Talluri *et al.* [56]. We chose a diverse range of failure models, based on the mean failure intensity in each trace (indicated in parentheses). As a result, we chose the traces from: (I) Gmail (53.26%), (II) WhatsApp (57.97%), (III) YouTube (62.1%), (IV) Twitter (65%), (V) Facebook (64%). In Section 5.3 we used a failure trace from Skype. This is the only trace that can be paired with a corresponding failure model (Figure 5.6). Additionally, in Section 5.2 we find a need to define a threshold based on a statistical distribution of failures. For this purpose, we use a normal distribution with mean 1.5 and standard deviation 1.5. Importantly, in our figures we do not report the standard deviation of our experiments. This is due to the fact that `OpenDC` is a fully deterministic simulator, and on each simulation run, given the same random seed will produce exactly the same results. We believe the deviation in the results of the experiments stemming only from the random number generator is not meaningful, therefore none of the figures contain the standard deviation bars.

5.2 Experiment 1: Failure Detection

The purpose of this experiment is two fold: 1. to show our system works correctly 2. to show our system fulfills the functional and non-functional requirements. To this end, we replicate an experiment from Taheri *et al.* [37]. Inspired by the idea of red and yellow alarms, based on the different confidence threshold, we adapt their experiment to our system. The experimental setup is as defined in Section 5.1. The experiment can be

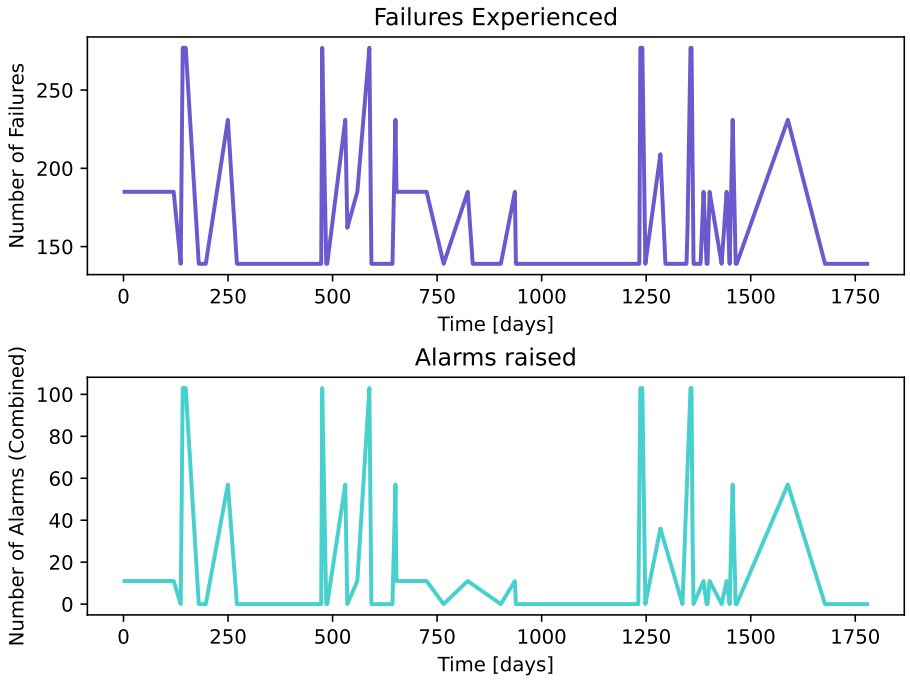

Figure 5.3: Comparison between the total number of raised alarms and the ground truth failure distribution during a BitBrains workload in the SURF-SARA cluster. The failure traced used in this experiment models Gmail outage reports [56].

described as follows: (1) firstly, we use `OpenDC` and a failure model with the normal distribution $\mathcal{N}(\mu = 1.5, \sigma = 1.5)$ to model the failures we might expect from a given workload. (2) then, using the predictions, we establish a threshold acceptable to datacenter operators (*i.e.,* how many failures can we tolerate before we raise any alarm) (3) the red alarm is raised when 90% of the threshold is reached, and the yellow alarm is raised when 80% of the threshold is reached. (4) lastly, the `OpenDC` acting as the real datacenter runs the workload, and *Sunfish* closely monitors the datacenter to see if the number of failures exceeds the accepted threshold. The results are in Figure 5.2. Figure 5.2 indicates *Sunfish* is capable of accurately detecting failures in datacenters. What is more, using the different threshold values, *Sunfish* can differentiate between serious failures and insignificant, single host problems. Importantly, the more failure-intense the trace, the more alarms are raised on behalf of the digital twin.

Additionally Figure 5.3 backs our claims, and verifies the results obtained in Figure 5.2. In the figure we can see a clear correlation between the total number of alarms raised, and the actual number of failures have occurred at each time during the workload. For this visualization, we combined both the red and yellow alarms into a single metric.

However, Taheri *et al.* present their results differently, using the *anomaly detection rate* instead. The rate is simply calculated as the anomalies detected correctly over the true amount of anomalies [37]. Therefore, in Figure 5.4 we also plot the failure detection rate. What is surprising is Taheri *et al.* report almost negligible false positive rate and of

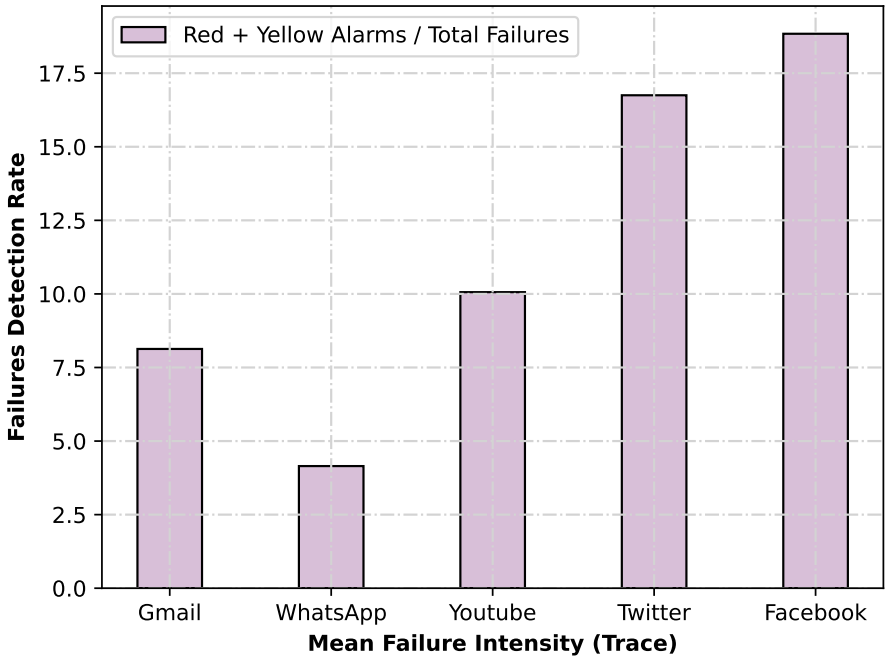

Figure 5.4: In this figure we show the total failure detection rate (■ Red + Yellow Alarms / Total Failures). Our results are much different from DyTwin's performance [37]. We believe this is due to the irreconcilable differences between our experimental setups.

their system. Moreover, they conclude through DyTwin's experimental setup, Taheri *et al.* achieve 100% anomaly detection rate. In our experiment, the numbers differ significantly. Figure 5.4 shows the mean failure detection rate to be around 12%. Compared to the DyTwin deployment, the difference is staggering. However, the discrepancy stems from the fact in our setup we differentiate between different types of failures. This capability is not present in the DyTwin digital twin [37]. As a result, we interpret Figure 5.4 as showing on average, 12% of failures in the workload are severe. Unusually, the WhatsApp failure detection rate is the lowest, contrary to the mean failure intensity, which places WhatsApp trace as the 2nd least failure-intensive trace.

5.3 Experiment 2: Failure Prediction

In Section 5.2 we show *Sunfish* is capable of incorporating descriptive analytics. Through experiment-based evaluation, we concluded *Sunfish* can detect and differentiate between severe and one-off host failures. In this section we try to show *Sunfish* can additionally work well together with a predictive analytics engine, enabling actionable insights into the future behaviour of the datacenter.

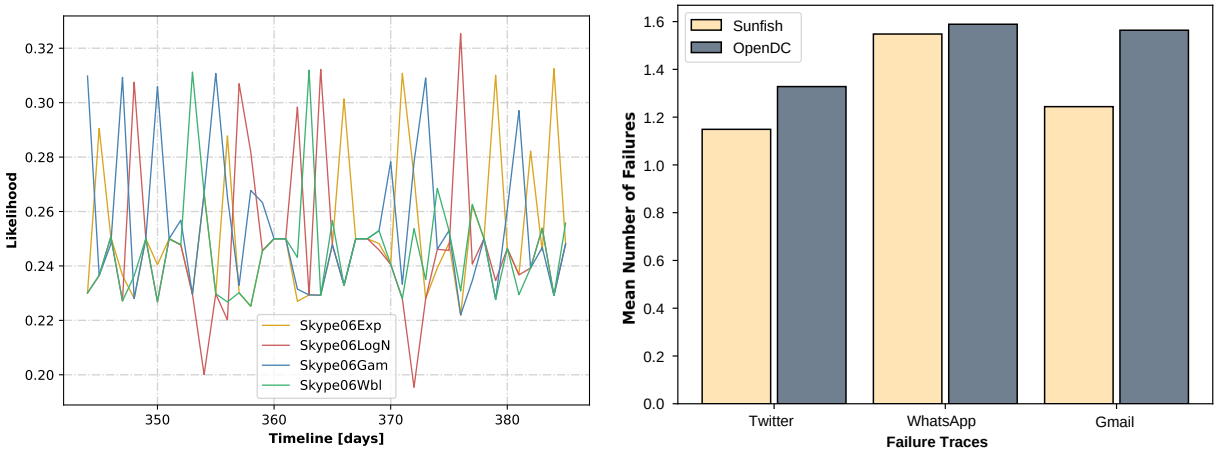

Figure 5.5: Left figure shows the potential failure distribution likelihood to approximate the true failure distribution. Right figure shows the results of the conceptual experiment to show the *potential* gains of employing a good predictive analytics engine with *Sunfish*.

	Availability intervals				Unavailability intervals			
Trace	**Exp**(μ)	**Wbl**(k, λ)	**LogN**(μ, σ)	**Gam**(k, λ)	**Exp**(μ)	**Wbl**(k, λ)	**LogN**(μ, σ)	**Gam**(k, λ)
lanl05	1779.99	0.48 816.60	5.56 2.39	0.35 5102.71	5.92	0.58 2.18	0.05 1.42	0.38 15.44
g5k06	32.41	0.48 14.37	1.51 2.42	0.34 94.35	7.41	0.35 0.47	-2.00 2.20	0.19 39.92
microsoft99	67.01	0.55 35.30	2.62 1.84	0.41 162.19	16.49	0.60 9.34	1.42 1.54	0.46 35.52
websites02	11.85	0.46 3.68	0.23 2.02	0.31 38.67	1.18	0.65 0.61	-1.12 1.13	0.50 2.37
pl05	159.49	0.33 19.35	1.44 2.86	0.20 788.03	49.61	0.36 5.59	0.40 2.45	0.21 237.65
ldns04	141.06	0.51 79.30	3.25 2.33	0.39 362.43	8.61	0.63 5.62	0.91 1.64	0.51 16.87
overnet03	2.29	0.85 2.04	0.19 0.98	0.91 2.53	12.00	0.44 2.98	0.08 1.80	0.29 41.64
nd07cpu	13.73	0.45 4.16	0.30 2.20	0.30 46.16	4.25	0.51 0.74	-1.02 1.27	0.28 15.07
skype06	16.27	0.64 10.86	1.60 1.57	0.53 30.79	14.31	0.63 9.48	1.40 1.73	0.50 28.53

Figure 5.6: The failure models table, by Javadi *et al.* [29].

5.3.1 Context

In order to predict when a host failure might occur, the most straightforward approach is to use long-established statistical methods. Our goal was to approximate the real failure distribution of a workload, using past data, and relevant statistical distributions. For the task at hand, we chose the Skype trace, because it is supported by 4 different failure models, based on past Skype workload data. These 4 statistical distribution, published in a peer-reviewed journal are in Figure 5.6 [29]. The Skype trace model was taken from the Cloud Uptime Archive [56]. The goal was to use the failure distribution to predict when a host will fail, and then in advance re-schedule all the tasks from the hosts onto different machines before it crashes.

Initial experiment results were unpromising. Using the insights from the failure models we were not able to do better than the baseline (switching hosts on and off randomly). To investigate why this might be the case, we run an experiment to identify which failure distribution at any given moment is most likely to resemble the actual, ground truth failure

distribution. Using a similarity score $\mathcal{S}$, which is a weighted average of the exported metrics, we tried to determine the most similar distribution at any given time. The results are in Figure 5.5.

In Figure 5.5 we can notice an almost random fluctuation of the similarity score $\mathcal{S}$. Any given failure model, at any time interval is almost as likely to model the actual failures as the other models. Moreover, the similarity score $\mathcal{S}$ of each failure model is never higher than 32%. This shows, the difficulty of good predictive analytics, and the correct design of a predictive analytics engine, which is not within the scope of this thesis.

Undeterred, we set out for a different solution to show *Sunfish* is capable of incorporating a predictive analytics engine. Instead, we designed a *conceptual experiment*. In this setup, we *assume* the predictive analytics engine is capable of fully predicting when each failure is going to happen with 100% accuracy. Equipped with this assumption, which only serves to show *Sunfish* meets the functional and non-functional requirements, we conducted the second experiment. The results are in Figure 5.5 on the right side. Figure 5.5 shows that using a perfect predictive analytics engine, *Sunfish* is capable of lowering the total number of failures significantly.

Chapter 6

Conclusion

Datacenter manageability is a top-priority for the digital society. Over 3 million jobs in the Netherlands directly depend on cloud services, which are hosted in datacenters [1]. Datacenter digital twinning, a promising management technique can offer unique insight into complex facility behaviour [3]. In this thesis we paved the way to more advanced DCDTs. We contribute to the scientific community a set of findings that we hope will prove helpful in enabling predictive analytics in both existing DCDTs and future projects. Starting from a thorough investigation into the new, emerging field of datacenter digital twinning, we designed a system capable of incorporating sophisticated data analysis techniques. We ended our project with a novel evaluation method used in a set of exhaustive experiments. We answer the main research question by addressing each sub-research question.

6.1 Answers to Research Questions

RQ_1 *How to asses the current state-of-the-art of digital twinning for datacenters?*
In order to answer this research question, we conducted a semi-structured literature review. Our findings indicate that the field of datacenter digital twinning is still under development, and there exist few DCDT deployments. The current efforts in modelling datacenters focus on very specialized parts of datacenter management, *i.e.,* cooling and heat modelling, network mapping. Many crucial features, inherent to the DT definition are still missing from current DCDTs. Present, standalone DCDT systems fail to offer the holistic capabilities envisioned by the inventors of DTs. The results of the literature survey are in Table 2.2, which contains systems which we found through a semi-structured literature review process. We first used structured queries, followed by a mix of snowballing and manual search. As a result, the second contribution to answering research question 2 is a holistic system model that encompasses the features of all the systems from Table 2.2 (see Figure 2.3).

RQ_2 *How to design a reference architecture for a predictive datacenter digital twin using discrete-event simulation?*
To answer this research question, we first brainstormed the potential use-cases for a

predictive DCDT. The use-cases are based on the findings of our literature survey. We list the use-cases we found in Chapter 3. Based on a set of use-cases we created a set of functional and non-functional requirements to guide our system design. Then, using the *AtLarge Design Process* we created the reference architecture that enables predictive analysis for datacenter operators through digital twinning.

RQ_3 *How to validate and evaluate a datacenter digital twin architecture in relation to system requirements?*
To answer the last research question we crated a prototype to evaluate our system. Lacking the physical datacenter to experiment with, we came up with a novel digital twin evaluation method, that uses discrete-event simulation to model the physical datacenter. Our main findings indicate that SF can reliably differentiate between large host failures and insignificant downtime using predictions based on the results from `OpenDC`, a state of the art datacenter modelling software. Moreover, we show that SF can be used to incorporate predictive analytics systems and significantly lower the total number of task failures during a workload.

6.2 Future Work

We envision DCDTs as systems that encompass features necessary to model the entire datacenter. It came to our attention that with the explosive growth of AI and the diversification of datacenters under way, DTs will be indispensable in datacenter management. To power the predictions we envision an ML-based inference engine as a necessary component of digital twinning. The need for ML arises naturally in scenarios where large volumes of data, requiring little to no preprocessing meet the demand for estimating future facility behaviour. For future work in failure prediction, we envision an Approximate Bayesian Computation (ABC) approach to estimate the real failure distribution within the datacenter. Additionally, power usage optimization is a critical concern in datacenter management. We hope future attempts to enhance datacenter digital twinning can enable datacenter operators with actionable insights towards lowering the power consumption.

Acronyms

ABC Approximate Bayesian Computation. 40

AI Artificial Intelligence. 3, 8, 16, 21, 24, 40

CFD Computational Fluid Dynamics. 16, 17

DCDT Datacenter Digital Twin. 5–9, 12, 16, 19, 20, 22, 32, 39, 40

DT Digital Twin. 4–7, 13, 14, 20, 26, 29–31, 39, 40

FAIR Findability, Accessibility, Interoperability and Reusability. 8

HP Hewlett Packard. 16

HPC High Performance Computing. 14

HTTP Hyper-text Transfer Protocol. 30

IoT Internet-of-Things. 4, 5

JSON JavaScript Object Notation. 16

LLMS Large Language Models. 16

MAE Mean Absolute Error. 17

ML Machine Learning. 16, 17, 24, 40

NASEM National Academy of Sciences, Engineering, and Medicine. 15, 20

ODA Operational Data Analysis. 5, 7, 8, 14, 15, 20

POD Proper Orthogonal Decomposition. 16, 17

PUE Power Usage Effectiveness. 23

RL Reinforcement Learning. 16

SF *Sunfish*. 7, 8, 20, 26, 28–31, 40

SLA Service Level Agreements. 5, 11, 13, 18–20

UI User Interface. 26, 28

VM Virtual Machine. 12

Bibliography

[1] Alexandru Iosup, Fernando Kuipers, Ana Lucia Varbanescu, Paola Grosso, Animesh Trivedi, Jan S. Rellermeyer, Lin Wang, Alexandru Uta, and Francesco Regazzoni. Future computer systems and networking research in the netherlands: A manifesto. *CoRR*, abs/2206.03259, 2022. URL https://doi.org/10.48550/arXiv.2206.03259.

[2] Dejan S. Milojicic, Paolo Faraboschi, Nicolas Dubé, and Duncan Roweth. Future of HPC: diversifying heterogeneity. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*, pages 276–281. IEEE, 2021. URL https://doi.org/10.23919/DATE51398.2021.9474063.

[3] Jyotika Athavale, Cullen E. Bash, Wesley Brewer, Matthias Maiterth, Dejan S. Milojicic, Harry Petty, and Soumyendu Sarkar. Digital twins for data centers. *Computer*, 57(10):151–158, 2024. URL https://doi.org/10.1109/MC.2024.3436945.

[4] Sacheendra Talluri, Leon Overweel, Laurens Versluis, Animesh Trivedi, and Alexandru Iosup. Empirical characterization of user reports about cloud failures. In Esam El-Araby, Vana Kalogeraki, Danilo Pianini, Fr édéric Lassabe, Barry Porter, Sona Ghahremani, Ingrid Nunes, Mohamed Bakhouya, and Sven Tomforde, editors, *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2021, Washington, DC, USA, September 27 - Oct. 1, 2021*, pages 158–163. IEEE, 2021. URL https://doi.org/10.1109/ACSOS52086.2021.00039.

[5] Douglas Donnellan, Andy Lawrence, and Rose Weinshenk. Executive summary: Annual outage analysis 2025, May 2025. URL https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2025.

[6] Alexandru Iosup. A vu on digital twins to improve the performance and technological sustainability of datacenters in the continuum. MODSIM World, Seattle, US, 2024. URL https://atlarge-research.com/talks/2023-modsim-alex.html.

[7] Wikipedia contributors. Digital twin — Wikipedia, the free encyclopedia, 2026. URL https://en.wikipedia.org/w/index.php?title=Digital_twin&oldid=1351132391. [Online; accessed 17-May-2026].

[8] Fei Tao, Meng Zhang, Yushan Liu, and A.Y.C. Nee. Digital twin driven prognostics and health management for complex equipment. *CIRP Annals*, 67(1):169–172,

2018. ISSN 0007-8506. URL https://www.sciencedirect.com/science/article/pii/S0007850618300799.

[9] Eric J. Tuegel, Anthony R. Ingraffea, Thomas G. Eason, and S. Michael Spottswood. Reengineering aircraft structural life prediction using a digital twin. *International Journal of Aerospace Engineering*, 2011(1):154798, 2011. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2011/154798.

[10] Eric Tuegel. The airframe digital twin: Some challenges to realization. 04 2012.

[11] Wesley Brewer, Matthias Maiterth, Vineet Kumar, Rafal P. Wojda, Sedrick Bouknight, Jesse Hines, Woong Shin, Scott Greenwood, David Grant, Wesley Williams, and Feiyi Wang. A digital twin framework for liquid-cooled supercomputers as demonstrated at exascale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2024, Atlanta, GA, USA, November 17-22, 2024*, page 23. IEEE, 2024. URL https://dl.acm.org/doi/10.1109/SC41406.2024.00029.

[12] Shekhar Suman, Xiaoyu Chu, Dante Niewenhuis, Sacheendra Talluri, Tiziano De Matteis, and Alexandru Iosup. Enabling operational data analytics for datacenters through ontologies, monitoring, and simulation-based prediction. In Simonetta Balsamo, William J. Knottenbelt, Cristina L. Abad, and Weiyi Shang, editors, *Companion of the 15th ACM/SPEC International Conference on Performance Engineering, ICPE 2024, London, United Kingdom, May 7-11, 2024*, pages 120–126. ACM, 2024. URL https://doi.org/10.1145/3629527.3652897.

[13] Barbara A. Kitchenham, Rialette Pretorius, David Budgen, Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen G. Linkman. Systematic literature reviews in software engineering - A tertiary study. *Inf. Softw. Technol.*, 52(8):792–805, 2010. URL https://doi.org/10.1016/j.infsof.2010.03.006.

[14] 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. URL https://doi.org/10.1109/ICDCS.2019.00175.

[15] 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 Laurent Lefèvre, Stacy Patterson, Young Choon Lee, Haiying Shen, Shashikant Ilager, Mohammad Goudarzi, Adel Nadjaran Toosi, and Rajkumar Buyya, editors, *21st IEEE/ACM International Symposium on Cluster, Cloud and*

Internet Computing, CCGrid 2021, Melbourne, Australia, May 10-13, 2021, pages 455–464. IEEE, 2021. URL https://doi.org/10.1109/CCGrid51090.2021.00055.

[16] 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. *Softw. Pract. Exp.*, 41(1):23–50, 2011. URL https://doi.org/10.1002/spe.995.

[17] 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. *Softw. Pract. Exp.*, 47 (9):1275–1296, 2017. URL https://doi.org/10.1002/spe.2509.

[18] 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. URL https://doi.org/10.1109/eScience.2012.6404430.

[19] 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. URL https://doi.org/10.1109/AINA.2010.32.

[20] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Simgrid: a sustained effort for the versatile simulation of large scale distributed systems. *CoRR*, abs/1309.1630, 2013. URL http://arxiv.org/abs/1309.1630.

[21] Etienne Michon, Julien Gossa, Stéphane Genaud, Léo Unbekandt, and Vincent Kherbache. Schlouder: A broker for iaas clouds. *Future Gener. Comput. Syst.*, 69: 11–23, 2017. URL https://doi.org/10.1016/j.future.2016.09.010.

[22] Henri Casanova, Ryan Tanaka, William Koch, and Rafael Ferreira da Silva. Teaching parallel and distributed computing concepts in simulation with WRENCH. *J. Parallel Distributed Comput.*, 156:53–63, 2021. URL https://doi.org/10.1016/j.jpdc.2021.05.009.

[23] Takahiro Hirofuchi, Adrien Lebre, and Laurent Pouilloux. Simgrid VM: virtual machine support for a simulation framework of distributed systems. *IEEE Trans. Cloud Comput.*, 6(1):221–234, 2018. URL https://doi.org/10.1109/TCC.2015.2481422.

[24] 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 (version 2.0). https://doi.org/10.5281/zenodo.7158509, November 2020. URL https://doi.org/10.5281/zenodo.7158509. Accessed on YYYY-MM-DD.

[25] 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*, Lecture Notes in Computer Science, pages 13–25. Springer, 2008. URL https://doi.org/10.1007/978-3-540-85451-7_3.

[26] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. Groudsim: An event-based simulation framework for computational grids and clouds. In Mario R. Guarracino, Frédéric Vivien, Jesper Larsson Träff, Mario Cannataro, Marco Danelutto, Anders Hast, Francesca Perla, Andreas Knüpfer, Beniamino Di Martino, and Michael Alexander, editors, *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*, Lecture Notes in Computer Science, pages 305–313. Springer, 2010. URL https://doi.org/10.1007/978-3-642-21878-1_38.

[27] 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. *J. Grid Comput.*, 10(1):185–209, 2012. URL https://doi.org/10.1007/s10723-012-9208-5.

[28] Jerry Banks, John S. Carson II, Barry L. Nelson, and David M. Nicol. *Discrete-Event System Simulation, 5th New Internatinal Edition*. Pearson Education, 2010. ISBN 978-1-292-02437-0. This BibTeX citation comes from https://dblp.org/rec/books/daglib/0034857.html?view=bibtex.

[29] Bahman Javadi, Derrick Kondo, Alexandru Iosup, and Dick H. J. Epema. The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems. *J. Parallel Distributed Comput.*, 73(8):1208–1223, 2013. URL https://doi.org/10.1016/j.jpdc.2013.04.002. This BibTeX citation comes from https://dblp.org/rec/journals/jpdc/JavadiKIE13.html?view=bibtex.

[30] National Academy of Engineering, National Academies of Sciences Engineering, and Medicine. Foundational research gaps and future directions for digital twins. The National Academies Press, Washington, DC, 2024. ISBN 978-0-309-70042-9. URL https://nap.nationalacademies.org/catalog/26894/foundational-research-gaps-and-future-directions-for-digital-twins.

[31] Wikipedia contributors. Predictive modelling — Wikipedia, the free encyclopedia, 2026. URL https://en.wikipedia.org/w/index.php?title=Predictive_modelling\

&oldid=1334478640. This BibTeX citations comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Predictive_modelling&id=1362368215&wpFormIdentifier=titleform#BibTeX_entry.

[32] Norman Bourassa, Walker Johnson, Jeff Broughton, Deirdre McShane Carter, Sadie Joy, Raphael Vitti, and Peter Seto. Operational data analytics: Optimizing the national energy research scientific computing center cooling systems. In *48th International Conference on Parallel Processing, ICPP 2019 Workshop Proceedings, Kyoto, Japan, August 05-08, 2019*, pages 5:1–5:7. ACM, 2019. URL https://doi.org/10.1145/3339186.3339210.

[33] Alessio Netti, Micha Müller, Carla Guillén, Michael Ott, Daniele Tafani, Gence Ozer, and Martin Schulz. DCDB wintermute: Enabling online and holistic operational data analytics on HPC systems. In Manish Parashar, Vladimir Vlassov, David E. Irwin, and Kathryn M. Mohror, editors, *HPDC '20: The 29th International Symposium on High-Performance Parallel and Distributed Computing, Stockholm, Sweden, June 23-26, 2020*, pages 101–112. ACM, 2020. URL https://doi.org/10.1145/3369583.3392674.

[34] Andrea Borghesi, Martin Molan, Michela Milano, and Andrea Bartolini. Anomaly detection and anticipation in high performance computing systems. *IEEE Trans. Parallel Distributed Syst.*, 33(4):739–750, 2022. URL https://doi.org/10.1109/TPDS.2021.3082802.

[35] Umit Demirbaga, Zhenyu Wen, Ayman Noor, Karan Mitra, Khaled Alwasel, Saurabh Garg, Albert Y. Zomaya, and Rajiv Ranjan. Autodiagn: An automated real-time diagnosis framework for big data systems. *IEEE Trans. Computers*, 71(5):1035–1048, 2022. URL https://doi.org/10.1109/TC.2021.3070639.

[36] Ziting Zhang, Yu Zeng, Haoran Liu, Chaoyue Zhao, Feng Wang, and Yunqing Chen. Smart DC: an AI and digital twin-based energy-saving solution for data centers. In *2022 IEEE/IFIP Network Operations and Management Symposium, NOMS 2022, Budapest, Hungary, April 25-29, 2022*, pages 1–6. IEEE, 2022. URL https://doi.org/10.1109/NOMS54207.2022.9789853.

[37] Ebad Taheri, Pedro Bruel, Pavana Prakash, Gourav Rattihalli, Ninad Hogade, Aditya Dhakal, Rolando P. Hong Enriquez, Torsten Wilde, Leo Popokh, Dejan S. Milojicic, and Cullen E. Bash. Dytwin: Federated adaptive digital twins for data centers - visualization and anomaly detection. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis , Atlanta, GA, USA, November 17-22, 2024*, pages 847–852. IEEE, 2024. URL https://doi.org/10.1109/SCW63240.2024.00120.

[38] Minghao Li, Ruihang Wang, Xin Zhou, Zhaomeng Zhu, Yonggang Wen, and Rui Tan. Chattwin: Toward automated digital twin generation for data center via large language models. In *Proceedings of the 10th ACM International Conference on Systems*

for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2023, Istanbul, Turkey, November 15-16, 2023, pages 208–211. ACM, 2023. URL https://doi.org/10.1145/3600100.3623719.

[39] Zhiwei Cao, Ruihang Wang, Xin Zhou, and Yonggang Wen. Reducio: model reduction for data center predictive digital twins via physics-guided machine learning. In Jorge Ortiz, editor, *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2022, Boston, Massachusetts, November 9-10, 2022*, pages 1–10. ACM, 2022. URL https://doi.org/10.1145/3563357.3564050.

[40] Hanshu Hong, Qin Wu, Feng Dong, Wei Song, Ronghua Sun, Tao Han, Cheng Zhou, and Hongwei Yang. Netgraph: An intelligent operated digital twin platform for data center networks. In *NAI'21: Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration, Virtual Event, USA, August 27, 2021*, pages 26–32. ACM, 2021. URL https://doi.org/10.1145/3472727.3472802.

[41] Ruihang Wang, Xin Zhou, Linsen Dong, Yonggang Wen, Rui Tan, Li Chen, Guan Wang, and Feng Zeng. Kalibre: Knowledge-based neural surrogate model calibration for data center digital twins. In *BuildSys '20: The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, Virtual Event, Japan, November 18-20, 2020*, pages 200–209. ACM, 2020. URL https://doi.org/10.1145/3408308.3427982.

[42] Haitao Zhu and Botao Lin. Digital twin-driven energy consumption management of integrated heat pipe cooling system for a data center. volume 373, page 123840, 2024. ISSN 0306-2619. URL https://www.sciencedirect.com/science/article/pii/S0306261924012236.

[43] Wikipedia contributors. Systems thinking — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Systems_thinking&oldid=1360310082, 2026. This BibTeX citation comes from: https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Systems_thinking&id=1360310082&wpFormIdentifier=titleform#BibTeX_entry.

[44] Wikipedia contributors. Grafana — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Grafana&oldid=1362050182, 2026. This BibTeX citation comes from: https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Grafana&id=1362050182&wpFormIdentifier=titleform#BibTeX_entry.

[45] Wikipedia contributors. Confluent — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Confluent&oldid=1361669280, 2026. This BibTeX citation comes from: https://en.wikipedia.org/w/index.php?

title=Special:CiteThisPage&page=Confluent&id=1361669280&wpFormIdentifier=titleform#BibTeX_entry.

[46] Wikipedia contributors. Redis — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Redis&oldid=1351092330, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Redis&id=1351092330&wpFormIdentifier=titleform#BibTeX_entry.

[47] Wikipedia contributors. Postgresql — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=PostgreSQL&oldid=1357817665, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=PostgreSQL&id=1357817665&wpFormIdentifier=titleform#BibTeX_entry.

[48] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. Opendc. This BibTeX citation comes from https://github.com/atlarge-research/opendc. It has been converted from a `CITATION.cff` file using `cffconvert` (see https://citation-file-format.github.io/).

[49] Radu Nicolae, Jules van der Toorn, Stavriana Kraniti, Houcen Liu, and Alexandru Iosup. Opendt: Exploring datacenter performance and sustainability with a self-calibrating digital twin. In Roberto Verdecchia, Catia Trubiani, Radu Calinescu, and Ana Lucia Verbanescu, editors, *Companion of the 17th ACM/SPEC International Conference on Performance Engineering, ICPE Companion 2026, Florence, Italy, May 4-8, 2026*, pages 142–147. ACM, 2026. URL `https://doi.org/10.1145/3777911.3800634`. This BibTeX citation comes from https://dblp.org/rec/conf/wosp/NicolaeTKLI26.html?view=bibtex.

[50] Wikipedia contributors. Kibana — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kibana&oldid=1342816695, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Kibana&id=1342816695&wpFormIdentifier=titleform.

[51] Wikipedia contributors. Prometheus (software) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Prometheus_(software)&oldid=1349456026, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Prometheus_%28software%29&id=1349456026&wpFormIdentifier=titleform.

[52] Wikipedia contributors. Graphite (software) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Graphite_(software)&oldid=1357227590, 2026. This BibTeX citation comes from

https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Graphite_%28software%29&id=1357227590&wpFormIdentifier=titleform.

[53] Wikipedia contributors. Protocol buffers — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Protocol_Buffers&oldid=1357670495, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Protocol_Buffers&id=1357670495&wpFormIdentifier=titleform#BibTeX_entry.

[54] Wikipedia contributors. Memcached — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Memcached&oldid=1361835153, 2026. This BibTeX citation comes from https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Memcached&id=1361835153&wpFormIdentifier=titleform#BibTeX_entry.

[55] 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. URL https://doi.org/10.1109/CCGrid.2015.60. This BibTeX citation comes from https://dblp.org/rec/conf/ccgrid/ShenBI15.html?view=bibtex.

[56] Sacheendra Talluri, Dante Niewenhuis, Xiaoyu Chu, Jakob Kyselica, Mehmet Çetin, Alexander Balgavy, and Alexandru Iosup. Cloud uptime archive: Open-access availability data of web, cloud, and gaming services. *IEEE Trans. Parallel Distributed Syst.*, 37(5):1136–1152, 2026. URL https://doi.org/10.1109/TPDS.2026.3673519. This BibTeX citation comes from https://dblp.org/rec/journals/tpds/TalluriNCKCBI26.html?view=bibtex.

Appendix