Vrije Universiteit Amsterdam

Honours Programme, Research Thesis

M3SA: Exploring the Performance and Climate Impact of Datacenters by Multi-Model Simulation and Analysis

---

Author: Radu Nicolae (2760443)
r.nicolae@vu.nl

1st supervisor:	Prof. Dr. Ir. Cav. Alexandru Iosup	(VU Amsterdam)
2nd supervisor:	Dante Niewenhuis, M.Sc.	(VU Amsterdam)

A report submitted in fulfillment of the requirements for the Honours Programme, which is an excellence annotation to the VU Bachelor of Science degree in Computer Science/Artificial Intelligence/Information Sciences

August 24, 2024

Abstract

Datacenters are vital for the digital society and represent a considerable fraction of global energy consumption, estimated between 1% and 2%, and are expected to rise to 8% by 2030, further reducing already over-exploited resources. In the operations of every datacenter, simulators play a crucial role in predicting the capabilities of real or virtual infrastructure, under various workloads. In the race to decrease the significant energy consumption and improve the efficiency of existing datacenters, multiple simulation instruments and tools have been developed, such as OpenDC, DCSim, and CloudSim. Although many of them proved to be useful in the process of lowering energy consumption, the current state-of-art is based on singular models embedded in either simulation or analytical frameworks, that offer good predictive capabilities only for the limited context in which they were developed. In the Computer Science field, it has never been built any tool that provides simulations based on multiple idiosyncratic models.

We propose M3SA, a tool for ICT simulation analysis, using multi-model techniques. M3SA provides solutions to improve the performance of existing simulations of the Information and Communication Technology infrastructure. We argue that, albeit still valuable, the development of individual simulation models is insufficient to make accurate predictions. We propose to create simulation tools that can leverage multiple models and combine their results. We filter out extremes and compute the most likely curve or range of values over time. By employing multi-model processes, we can develop more efficient simulators, which would aid in developing, configuring, and operating efficient, energy-conserving, and cost-friendly datacenters.

Keywords

ICT, datacenters, multi-model simulation, meta-simulation, energy utilization, sustainability

Contents

1
Introduction

The datacenter infrastructure is critical for the digital society. To keep up with the fast-paced rate of daily data traffic, currently estimated at 274 exabytes per day [64], existing datacenters are being increased in size, and new datacenters are being built. This is directly related to the environmental issue of overexploiting our limited resources and deepening the concerning problem of climate change. Datacenters already consume between 1% and 2% of global electricity [37], and this percentage is expected to rise to 8% by 2030 [57]. Some of the Information and Communication Technology (ICT) infrastructures with the largest energy consumption are hyperscale datacenters, with a consumption over 200 TWH/year, worldwide, and cloud datacenters, with a total yearly consumption estimated at 140 TWh/year [54]. For comparison, in 2018, only hyperscale datacenters consumed over 200 TWh, which is more energy than the Netherlands (117 TWh) and Romania (55 TWh) combined, according to the International Energy Agency [3]. The massive energy consumption of the datacenters directly contributes to the global carbon footprint, accounting for 1% of global greenhouse gas emissions (GHG) in 2021 [59, 1, 36].

Datacenters have a significant impact on the worldwide economy. The digital economy represents 5% of the international Growth Domestic Product (GDP) and 3% of the international employment, leading to a total of €4.81 trillion in 2021 [82, 12]. In 2024, the Netherlands hosts 307 datacenters, ranking third in the European Union [68]. This makes the Netherlands a key region on the ICT map. In 2019, digital services hosted in datacenters represented 33% of the Dutch economy, generating €242 billion in GDP, and 2.1 million jobs [57]. Despite the high economic importance of this large-scale, critical infrastructure, insignificant improvements have been made in terms of efficiency during the past decade [72]. The aim is to improve the efficiency of the infrastructure; a commonly used datacenter efficiency metric is the Power Usage Effectiveness (PUE), which we aim to converge towards the optimum of 1.0. The Climate Neutral Data Centre Pact mandates that, by 2025, datacenters will achieve an annual PUE of 1.3 and 1.4 for the datacenters running at full capacity in warm climates [19]. Although the average PUE significantly reduced from 2.6, in 2007, to 1.6, in 2015, the decline has stagnated over the past years [57, 69]. Furthermore, the sharp increase in energy prices in 2022 had a significant financial impact on datacenters with a bad (high) PUE [57].

Predicting the capacity of datacentres is a critical yet non-trivial simulation problem that could lead to significant service improvements, cost savings, and environmental sustainability worldwide. Simulation enables large-scale and fine-grained exploration analysis and comparison of datacenter technologies [56]. The constant accelerating increasing rate and demand for computing power have led to a substantial expansion of the datacenter infrastructure, both in terms of scale and complexity, in the need to serve stakeholders in industry, society, government, and academia [57]. Datacenter simulation is highly important economically and environmentally. The climate impact between simulating a datacenter configuration and performance and actual building, configuring, and running that experiment is significantly lower for the simulation-based approach, assuming the simulation is accurate and correct; for example, a recent analysis estimates a ratio of 1:116,000,000,000 in energy consumed to conduct simulations over the equivalent real-world experiments [38]. The ratio could be increasingly worse as the scale of the infrastructure increases as scalable simulation techniques are used. Inaccurate simulations can lead to infrastructure failing to achieve the desired benchmarks or result in a less efficient setup than if no simulation had been conducted [17, 50, 55, 75, 22, 76].

We argue that, albeit still valuable, the current simulation state-of-the-art is limited, imprecise, and inaccurate, as the existing simulators are predicting based on a singular model, prone to errors and mishandling edge

cases. In general, we observe that all the existing simulation instruments, able to predict datacenters metrics under different configurations (topologies) and various workloads to execute, rely exclusively on predictions of idiosyncratic models.

In this work, we address the precision and accuracy gap between the current simulation, based on individual simulation models, and a novel proposed ICT simulation technique, based on multiple simulation models. We propose, design, and implement a novel simulation system that leverages multiple individual simulation models into a unified tool. We explore the methodologies used by researchers across various scientific fields to create simulators that incorporate multiple models for enhanced prediction and accuracy. We also explore the specific calibration and optimization techniques implemented by these researchers to enhance the accuracy of their predictions [21, 63, 33, 58, 7]. We investigate the integration of multiple models under a unified analytical tool and understand how simulation based on multiple models can be integrated into ICT. We then envision the concept of simulation based on multiple models, analyze design choices, and materialize the concept into a tool. Furthermore, we propose a meta-simulation concept, a novel model that predicts using the predictions of multiple singular models. We materialize this concept into the simulation tool and deliver a unitary, holistic system compatible with any ICT simulator. Lastly, we integrate our system into a peer-reviewed, state-of-the-art datacenter simulator and evaluate the system's capabilities against real-world data.

1.1 Problem Statement

Over the past decades, the number of datacenters has grown exponentially as a response to the high demand for computing [54, 18]. Proven to be critical instruments in datacenter designing, scaling, maintaining, and building, many datacenter simulators have been developed and used worldwide [7, 56, 34]. These instruments are therefore required to provide highly accurate simulation data, leading to critical consequences when they fail to [75, 22, 76].

We argue that, albeit still valuable, the development and usage of simulators using individual simulation models is insufficient to make accurate predictions. A singular model only offers accurate predictions for the limited context it has been developed for, often failing to handle real-life edge case scenarios [21, 58, 56]. Moreover, these models are usually calibrated to ideal scenarios and lack adaptability to unforeseen circumstances, such as sudden spikes in demand, equipment failures, power outages, or environmental impact [57]. Such scenarios can lead to significant discrepancies between predicted and actual performance, potentially resulting in financially costly downtime [75], health-threatening [22], and overall systems shutdowns [76].

Different individual models are precise in different individual cases. Similarly, individual models pose errors in different circumstances, especially when encountering edge cases.

Many individual prediction models have been proposed and implemented into datacenter simulators for different scenarios, topologies, and workloads [34]. However, no such datacenter simulator aggregates into a unified tool multiple simulation models. We posit that an advantage of combining multiple models is making it easier to identify the biases of each individual model by simply contrasting the results obtained across multiple models for the same simulated scenario, thus alleviating idiosyncratic errors of individual models and strengthening the credibility of simulation results. There currently exists no theoretical model that can solve this issue, which means that an empirical approach could at least provide initial insights into the possible advantages and drawbacks of the multi-model approach.

Up to the moment of publication, no tool has been built in the ICT field that leverages multiple simulation models and presents individual model predictions against other models. Also, no tool has been built in ICT that uses other models' predictions, thus filtering out idiosyncratic errors, granular imprecision, and inaccuracies that singular models may pose when edge cases are encountered.

1.2 Research Questions

To address the aforementioned challenges, we raise the main research question **(MRQ)**, from which we refine a sequence of three research questions **(RQ)**.

MRQ How to explore, implement, and test a concept and a metric to express the efficiency of strategies for integrating multiple simulation models into a unified tool?

Research Question 1

Large-scale and small-scale sciences already use simulation instruments which rely on multiple models, calibrated and adjusted for the needs of the scientific field [65, 33, 58]. However, Computer Systems, as a medium-scale scientific field, is fundamentally different than other-scale sciences [4] regarding simulation granularity, scale, and analyzed phenomena. We identify the main challenge of adapting the simulation concepts employed by other sciences, and designing a tool applied to Computer Systems, able to simulate and provide comprehensive insights, synthesized from the predictions of multiple models.

Toward answering RQ1, we identify the primary requirement to synthesize existing models based on well-defined criteria. These models would serve as components within a simulation framework using multiple models. Once selecting a good set of models, we run multiple individual models and accumulate their results into a unified simulation tool, which we label the *Multi-Model*. The concept of simulation based on multiple models represents a novelty in the realms of ICT and raises the research question:

RQ1 How to design a Multi-Model simulator that leverages the results of singular models?

Research Question 2

It has never been developed in the medium-scale scientific field of Computer Systems, a tool that is able to predict using existing predictions. We identify the main challenge of designing a performant and universal tool for ICT simulators. We also identify the challenge of defining accuracy and then evaluating the validity of the novel tool.

Toward answering RQ2, we identify the main requirement of envisioning and implementing a meta-simulation concept, in which a new simulation model is created using the predictions of multiple individual simulation models. This novel model enhances decision-making by providing distilled simulations using various meta-simulation techniques. This raises the research question:

RQ2 How to design a Meta-Model simulator that combines the outputs of the Multi-Model, for a more accurate prediction?

Research Question 3

Proposing a (successful) novel simulation concept is a rarity in our field and represents a potential massive-scale contribution if widely adopted. The main challenge is demonstrating the adaptability and evaluating the validity of the proposed system. This raises three sub-challenges. Firstly, we identify the (sub-)challenge of materializing the concept proposed in RQ1 and RQ2 into an engineered tool with minimal redundancy, maximized performance, abstraction, and universality, following industry-standard, state-of-the-art techniques. Secondly, we identify the consequent (sub-)challenge of integrating the engineered tool with an top-tier, peer-reviewed, state-of-the-art, simulator, while keeping the overall system (simulator and tool) performant. Thirdly, we identify the (sub-)challenge of evaluating the overall system against well-defined criteria.

After answering RQ1 and RQ2, we obtain M3SA - a concept and tool able to integrate within any datacenter simulator. M3SA proposes and delivers the Multi-Model and the Meta-Model as a unified system. To answer RQ3, we identify the main requirement of integrating M3SA within a large-scale, peer-reviewed, top-tier datacenter simulator and evaluate the system's functionality. We identify the subsequent requirement of evaluating the performance, prediction validity, usability, and universality of M3SA. This raises the research question:

RQ3 How to integrate and evaluate M3SA?

1.3 Approach

Throughout the research and engineering process, we approach the problem statement and the subsequent research questions with a distributed systems approach, "a combination of conceptual, technical, and experimental work" [55], guided by the state-of-the-art AtLarge Design Process [40].

To answer RQ1, we conduct a literature survey on simulators using multiple models in other sciences. Further, we conduct a literature survey to identify, synthesize, and select existing models based on good criteria focused on performance and diversity. After we fulfill this component, we have an extensive background on the existing simulation-related work in ICT, datacenter simulation instruments, and simulation concepts based on multiple models in other sciences. We present these background elements in Section 2. Upon assembling extensive, comprehensive background knowledge, we build the Multi-Model, a unified tool that can leverage the simulation outputs of individual models', visually represent their simulations, and make statistics. We define functional and non-functional requirements which guide the research and engineering process. We analyze design choices and the overall simulation concept employing multiple models, analyze data aggregation (windowing), and engineer toward universality, scalability, and performance.

To answer RQ2, we expand upon the conceptualized and materialized Multi-Model simulation tool from RQ1. We envision the concept of meta-simulation and set functional and non-functional requirements, which guide our work throughout the research and engineering process. We propose an overall system architecture able to make such predictions, analyze design choices, propose meta-simulation techniques, and engineer a system. We term this tool *Meta-Model*. We provide the tools from RQ1 (Multi-Model) and RQ2 (Meta-Model) as a holistic embedded system. We term this system M3SA. We design M3SA as performant and scalable, able to integrate within any datacenter simulator.

To answer RQ3, we integrate M3SA within a peer-reviewed, large-scale, top-tier datacenter simulator. We present the structure of the datacenter simulator, provide comprehensive integration documentation, and present, in practice, the integration of our system within the chosen datacenter simulator. Then, we evaluate the overall system performance and accuracy through reproducible experiments, and present real-life possible use cases of M3SA as applied to a simulator.

1.4 Contributions

This paper will impact the scientific community by merging multiple models, comparing their results, and presenting an optimal, precise, and accurate simulation tool based on the output results from the subsequent models. This concept has been implemented in other scientific fields and yielded impactful results, yet marks a pioneer in the realm of Computer Science. This can pioneer a new stage in the sub-field of cloud-infrastructure simulations, a significant simulation advancement, and hence a considerable improvement in datacenter infrastructure development, scaling, and *massivizing*.

With this research, our key contributions are:

C1 We conduct a literature survey to identify, synthesize, and select relevant models based on performance reported by authors and diversity identified by our survey process. We review existent energy and emission models integrated in peer-reviewed datacenter simulators.

C2 We synthesize a simulation system using multiple models for extensively handling real-life scenarios. We investigate the integration of multiple models into a unified analytical tool and develop such a tool, termed *Multi-Model*. We design the Multi-Model as open to the public and easily adaptable to any datacenter simulator as a top layer applied to the existing codebase.

C3 We investigate alternatives to synthesize a composite model capable of selecting the most accurate prediction at fine time granularity. We term this simulation tool *Meta-Model*. We design the Multi-Model (**C2**) and the Meta-Model as a unified embedded system - M3SA. We further investigate, synthesize, and run testing extensive analysis of the accuracy, precision, and performance of the *Meta-Model*.

C4 We integrate and evaluate M3SA using a peer-reviewed, top-tier datacenter simulator. We present the overview of M3SA, in Figure 1. We use OpenDC, an open-source platform for cloud datacenter simulation, built through 7+ years of development and operation [55, 56, 57].

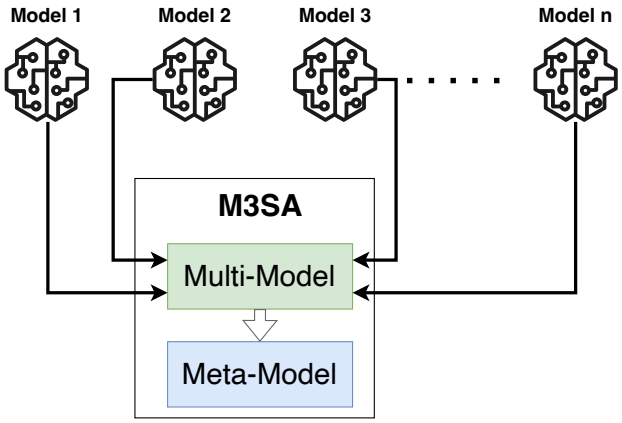


Figure 1: M3SA overview.

This paper also has a significant impact on my personal development as an independent researcher and represents my first steps in the computer systems research field, my first contributions to the community, and my first scientific work towards *Massivizing Computer Systems*.

1.5 Plagiarism Declaration

I confirm that this thesis is my own work, is not copied from any other source (person, Internet, or machine), and has not been submitted elsewhere for assessment. The work, findings, and formulations that do not represent my contribution, are given explicit recognition via citations.

1.6 Thesis Structure

The remainder of this paper is structured as represented in Figure 2. In Section 2, we describe relevant background information. In Section 3, we present the Multi-Model concept and tool. In Section 4, we present the Meta-Model concept and tool. In Section 5, we integrate and evaluate a prototype of the system. In Section 6.

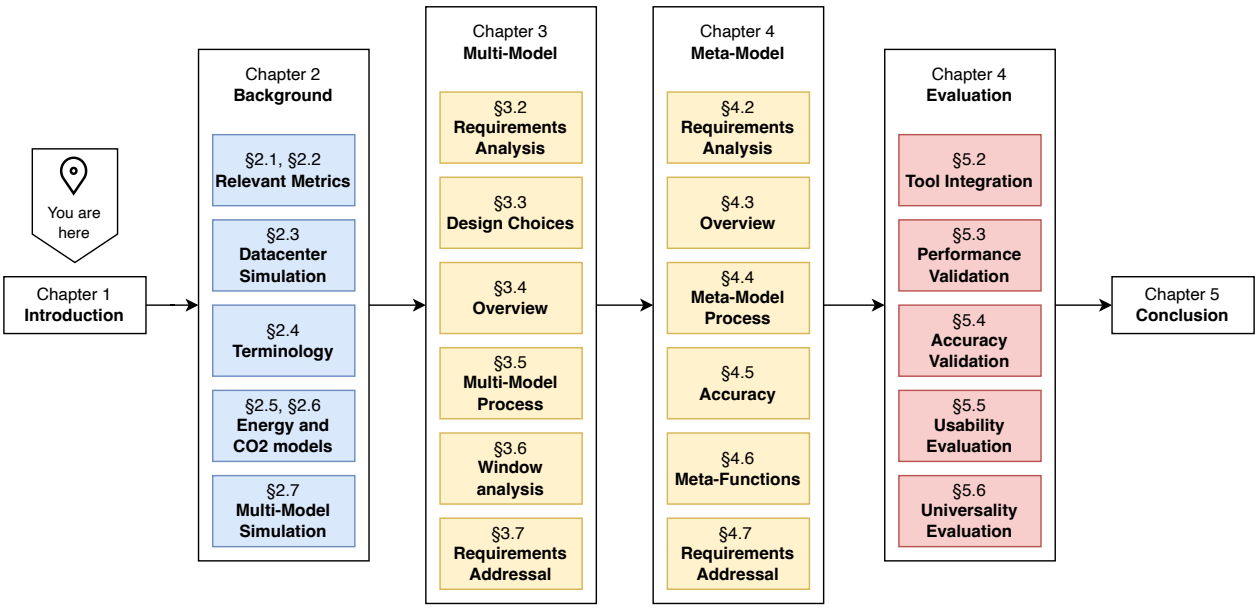


Figure 2: The structure of this paper.

2
Background

In this chapter, we present a comprehensive overview of the subjects related to ICT simulation, which serves as a foundation for the rest of the paper.

Metrics are crucial decision-making tools that help establish a common ground. When researching ICT infrastructure, it is essential to comprehend, analyze, and utilize the metrics commonly used in the community. This work identifies two main categories of metrics that are highly relevant for ICT simulation: energy metrics (Section 2.1) and CO2 emission metrics (Section 2.2).

In Section 2.3, we present related research on datacenter simulator tools and introduce the concept of simulation models. We further divide the simulation models into two categories: energy usage models (Section 2.5) and CO2 emissions models (Section 2.6).

After establishing a comprehensive high-level background in datacenter simulation models, we present the current state-of-the-art employed by the simulators, which simulate using singular models (Section 2.7). Then, we introduce the concept of simulating using multiple models. We provide insights from other sciences that have successfully designed, implemented, and tested this concept (Section 2.8).

2.1 Energy Metrics

2.1.1 Power Usage Effectiveness (PUE)

Introduced in 2006 by Malone et al. [52, 34], Power Usage Effectiveness (PUE) is an end-user tool consisting of a metric *"for understanding how well a datacenter is delivered energy to its information technology equipment"* [9]. PUE is the ratio of the total energy and the energy that is used for the actual computation.

$$PUE = \frac{E_T}{E_{IT}} \tag{1}$$

where E_T denotes the total energy used by the datacenter and E_{IT} denotes the energy used by the IT components of the datacenter.

Equation (1) [9] provides a high-level mathematical equation to compute the Power Usage Effectiveness factor of an ICT infrastructure. PUE can take values between a minimum of 1.0 and an infinite maximum. Lower values of PUE are better, and the aim is to get as close to 1.0 as possible. A PUE of 1.0 means that the IT equipment uses all energy received by the datacenter, yet impossible to achieve due to the laws of physics.

The Climate Neutral Data Centre Pact [19] mandates that, by 2025, new datacenters in cool climates will meet an annual PUE target of 1.3 and 1.4 for new datacenters in warm climates. By 2030, all datacenters must meet the PUE target of 1.3 in cool climates and 1.4 in warm climates. [19]. Despite significant improvements in PUE, from an average of 2.6 in 2007 to 1.6 in 2015, the decline has stagnated in recent years (Figure 3), while the overall energy usage is alarmingly increasing [59]. Despite Google achieving an average annual PUE of 1.1 in 2023 [28], and BTDC (Sweden) setting a PUE record of 1.014 in 2021 [48, 73], the global average PUE remains worryingly high, at 1.58 in 2023 [69]. Besides environmental concerns, the sharp increase in

energy prices in 2022 has a significant economical impact on administrators of datacenters with a bad (high) PUE factor [57].

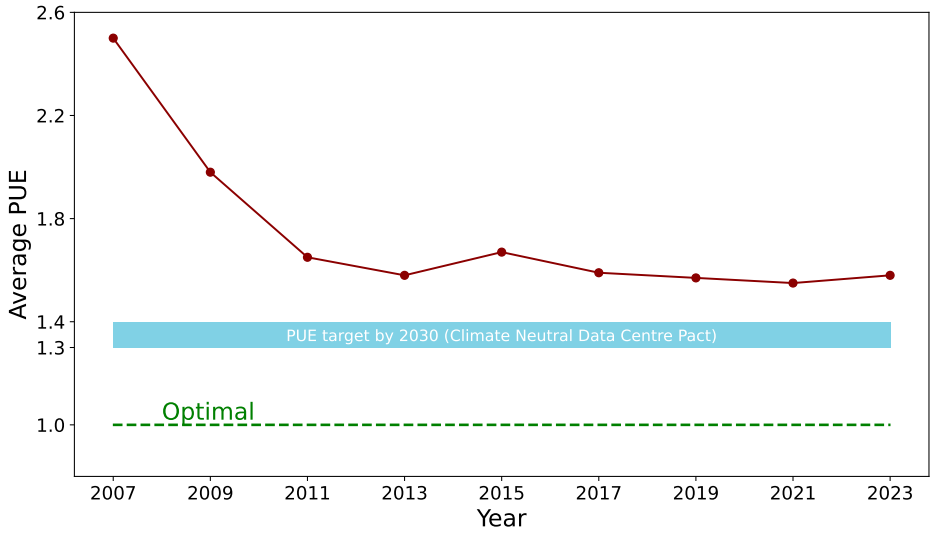


Figure 3: PUE Evolution Between 2007 and 2023 [69].

To transpose the numbers above to real examples, we will consider a hyperscale datacenter, which aims to improve the average annual PUE. In our hypothesis, we consider the annual PUE of the datacenter equal to 1.58, denoted as x_1. This value represents the average PUE of datacenters worldwide in 2023 [69]. To meet and improve beyond the Pact-mandated metrics, the administrators want to improve and achieve an average annual PUE of 1.25 (i.e., the target PUE), denoted as x_2. We assume that the datacenter consumes 100 GWh per year (i.e., total facility energy) and is denoted as y. We denote the energy used for computation (i.e., IT Equipment Energy) as z_1, for the current PUE, and as z_2, for the target PUE. We assume the average price per GWh is approximately €350,000 (i.e., the average price per GWh in 2024, in the Netherlands [71]), and denote as p.

$$p \approx 350,000\,EUR \quad \text{(approx. price per GWh, Netherlands, 2024 [71])} \tag{2}$$
$$x_1 = 1.58 \quad \text{(current PUE of the datacenter)} \tag{3}$$
$$x_2 = 1.25 \quad \text{(target PUE of the datacenter)} \tag{4}$$
$$y = 100\,GWh \quad \text{(total yearly consumption)} \tag{5}$$
$$z_1 = \frac{y}{x_1} \approx 63.29\,GWh \quad \text{(IT components yearly consumption with the } x_1\text{)} \tag{6}$$
$$z_2 = \frac{y}{x_2} = 80.00\,GWh \quad \text{(IT components yearly consumption with the } x_2\text{)} \tag{7}$$

$$i = \frac{|z_1 - z_2|}{z_2} \approx \frac{|x_1 - x_2|}{x_2} \approx 20.89\% \quad \text{(energy saved)} \tag{8}$$
$$\Delta z \approx z_2 - z_1 \approx 16.71\,GWh \quad \text{(energy saved yearly)} \tag{9}$$
$$\Delta p \approx \Delta z \cdot p \approx 5,848,500\,EUR \quad \text{(money saved yearly)} \tag{10}$$

Under the aforementioned hypothesis, we identify a 20.89% improvement in energy consumption, resulting in approximately 16.71 GWh saved per year, equivalent to savings of approximately 5,848,500 EUR.

2.1.2 Datacenter Performance Efficiency (DCPE)

Derived from PUE, Datacenter Performance Efficiency (DCPE), also referred to as Compute Power Efficiency (CPE), is a metric used to measure the computational efficiency of datacenters. DCPE was introduced by *Malone et al.* and used to capture the fraction of energy used for computation.

$$DCPE = \frac{U_{IT}}{P} = \frac{U_{IT} \cdot E_{IT}}{E_T} \tag{11}$$

U_{IT} is the IT Equipment Utilization, P is PUE, E_{IT} is the energy used by the IT components of the datacenter, E_T is the total energy used by the datacenter.

We observe that even slight changes in the Power Usage Effectiveness metrics significantly impact the datacenter Performance Efficiency factor. To illustrate this, we use the example of the hyperscale datacenter presented in Section 2.1.1. We analyze the increase in the DCPE factor between the PUE of $x_1 = 1.58$ and $x_2 = 1.25$. We determine a 26.98% improvement of the DCPE factor.

$$U_{IT} \qquad \text{(IT Equipemnt Utilization)} \tag{12}$$

$$d_1 = \frac{U_{IT}}{x_1} = \frac{1}{1.58} = 0.63 \qquad \text{(DCPE for PUE of } x_1 = 1.58\text{)} \tag{13}$$

$$d_2 = \frac{U_{IT}}{x_2} = \frac{1}{1.25} = 0.80 \qquad \text{(DCPE for PUE of } x_2 = 1.25\text{)} \tag{14}$$

$$p_i = \frac{|d_1 - d_2|}{d_1} = 26.98\% \qquad \text{(Performance Improvement)} \tag{15}$$

2.2 CO2 Metrics

PUE is an excellent metric to quantify ICT infrastructure's performance and energy efficiency. However, PUE does not consider the energy efficiency of applications and workloads [84] and overlooks the type of energy used [59]. While there is a correlation between a datacenter's power draw (i.e., the energy consumed) and the CO2 emissions, several other factors influence the amount of CO2 emitted. Determining the CO2 footprint of the datacenter under a specific workload is an environment-critical, yet not trivial, challenge. Many datacenters use energy from the grid, generated through various sources, with various environmental impacts (e.g., solar, wind, coal). In some cases, energy used from renewable sources, such as wind or solar, can emit up to 20x less CO2 compared to traditional energy sources, such as coal [59, 32].

Niewenhuis et al. presents two types of CO2 emissions in the datacenters: *i) the embodied* carbon footprint and *ii) the operational* carbon footprint. Embodied carbon denotes the manufacturing and production results emissions. Operational Carbon Footprint is the CO2 emissions caused by energy usage during datacenter operations. This work proposes a simulation-based solution to alleviating the concerning and deepening environmental problem of CO2 emissions, focusing on the operational carbon footprint [59].

2.2.1 Carbon Metrics

The *Carbon Intensity* of an energy source defines the amount of CO2 emitted per unit of energy used [59]. The measurement unit in the international system is $[gCO_2/kWh]_{S.I.}$. Datacenters utilize energy from the grid [59]; the energy is often provided by multiple sources with distinct Carbon Intensities [59]. Therefore, the Carbon Intensity of the grid is calculated by adding up the Carbon Intensity of each source, proportional to the amount of energy consumed (Equation 16).

$$\mathrm{CI}_g = \sum_{s \in S} \mathrm{CI}_\mathrm{s} \cdot \frac{E_\mathrm{s}}{E_\mathrm{g}} \qquad [gCO_2/kWh]_{S.I.} \tag{16}$$

where CI_g is the Carbon Intensity of the grid, CI_s is the Carbon Intensity of the source, E_s is the energy from a specific source, E_g is the total grid consumption, s is the selected source, S the set of all available energy sources [59].

The *Carbon Emissions* of the grid fluctuates depending on the geographical location (Figure 4), time of the day (Figure 5), temperature, weather conditions, et cetera. The amount of green energy delivered peaks during the day, while during the night, energy from "grey" sources (e.g., coal) is predominantly used [2].

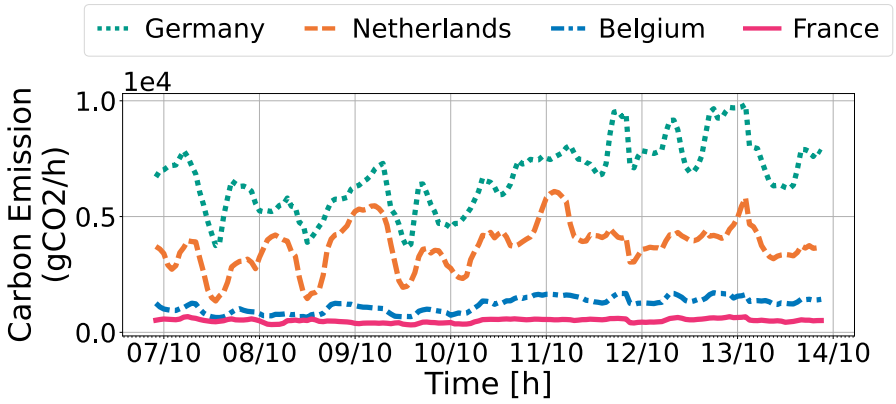


Figure 4: CO2 emission fluctuation, location-dependent. Taken from [59].

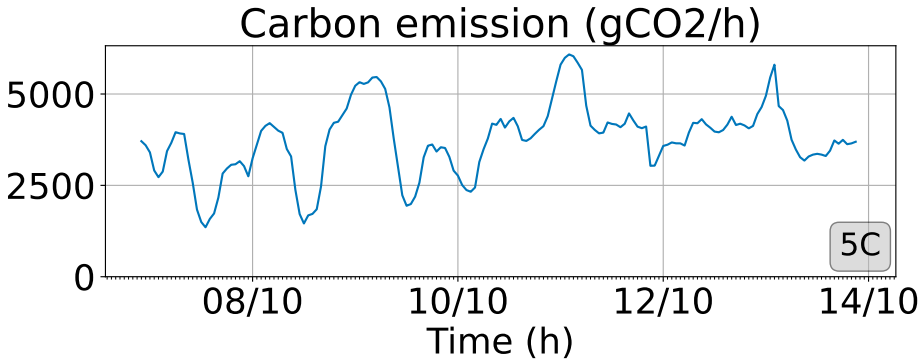


Figure 5: CO2 emission fluctuation, over time. Taken from [59].

The *Operational Carbon Footprint* denotes the CO2 emitted when the system is running. The *Operational Carbon Footprint* can be computed using Equation (17).

$$\mathrm{C}_{op} = \mathrm{CI}_d \cdot E_{op} \qquad [gCO_2]_{S.I.} \tag{17}$$

where C_{op} is the Operational Carbon Footprint, CI_d is the Carbon Intensity of the datacenter $[gCO_2/kWh]_{S.I.}$, E_{op} is the operational energy of the datacenter $[kWh]_{S.I.}$ [59].

In this work, we employ simulation based on multiple models to predict the *Operational Carbon Footprint* of various configurations of datacenters, under distinct workloads and scenarios.

2.3 Datacenter Simulation Frameworks

Datacenters serve as vital cloud infrastructure, playing a crucial role in the digital society by serving stakeholders from industry, government, and academia [7, 6, 56, 39]. Extensive research has been conducted

in this field, including analyzing and predicting data traffic evolution, developing datacenter simulators, and proposing novel scheduling techniques. In this subsection, we present the data traffic trends (Section 2.3.1), which increased by one order of magnitude within the last decade; this data, is handled, created, transferred, and reproduced via massive-scale ICT infrastructure, which is in a continuous expansion and growth [38, 39, 37]. The current state-of-the-art consists of simulating before building; we further expand on datacenter simulation frameworks in Section 2.3.2.

2.3.1 Data Traffic Trends

Reinsel et al. analyzed data traffic trends, presenting a one-order-of-magnitude increase to 163ZB by 2025, compared to just 16ZB in 2016. These data include 25ZB of critical information and 4ZB of hypercritical data, directly impacting users' health, life, commercial air travel, military security, and numerous other situations. In addition, by 2025, approximately 75% of the global population is estimated to be connected to the Internet. The research carried out by Reinsel et al., as part of an IDC White Paper sponsored by Seagate, underscores the vital importance of establishing reliable and efficient datacenters, scalable to billions of people and tens of billions of devices [64].

2.3.2 OpenDC Simulation Instrument

Iosup et al. analyzed existing datacenter simulators, highlighted, and addressed simulation challenges by introducing OpenDC 1.0 [39], succeeded by OpenDC 2.0 [56], introduced by *Mastenbroek et al.*. OpenDC is an open-source platform for modeling, simulation, and experimentation with cloud datacenters. OpenDC 2.0 addresses multiple key challenges:

1. contains models for emerging technologies, such as serverless computing and machine learning workloads running in datacenters;
2. contains models for CO2 emission predictions and models for energy usage predictions, calibrated with real-life data;
3. provides an intuitive interface with enhanced visualization and interaction tools, supporting various input/output formats and metrics. OpenDC 2.0 facilitates the process of designing and sharing (parts of) complex datacenters;
4. provides both a GUI and JSON interfaces towards accommodating a wide range of stakeholders, including experts and general users;

OpenDC 2.0 is a pioneering and re-engineered iteration of the 1.0 prototype, becoming the first simulator to integrate serverless and machine-learning execution while leveraging discrete-event simulation. This simulator integrates a model for the TensorFlow ecosystem and primarily employs Kotlin as the main programming language for the codebase. The authors compare the developed datacenter simulation concepts and architecture with *i) Mathematical Analysis*, albeit faster, too high-level for the processes from a datacenter and with *ii) Real-world experimentation*, which yields accurate results, is non-trivial to run at a large scale due to high energy footprint and extensive waiting times.

With highly precise and accurate simulations, open-source nature, and a wide variety of distinct models used in simulations, OpenDC has proven results through multiple peer-reviewed, award-winner, top-tier publications [59, 34, 39, 38, 56, 55, 57, 7, 5, 6]. We identify OpenDC framework and the related work as highly relevant for this research. We use OpenDC to implement and test the pioneering ICT concepts of Multi-Model and Meta-Model simulation.

2.3.3 Reference Architecture

Andreadis et al. propose a reference architecture for datacenter schedulers [7]. Though indubitably powerful and indispensable, existing scheduling systems employ complex designs and diverse approaches, thus becoming difficult to comprehend and compare. Moreover, scheduler stages are commonly underspecified, further deepening existing challenges. The authors map fourteen schedulers, encompassing academic and industry solutions and establishing a solid foundation for understanding and comparing various scheduling

approaches. They analyze schedulers developed before and after 2010, as well as Google's Borg scheduler, offering valuable insights into industry-leading, large-scale systems. Future work involves mapping more schedulers, integrating emerging paradigms, and conducting interviews with scheduler developers [7].

2.4 Terminology

A workload denotes the set of computational tasks provided to a system. In this work, a workload includes jobs given to the datacenter for computation.

A trace is a fine-grained record of operations within a system. In this work, a trace represents a workload submitted to the datacenter, monitored at a constant time granularity. Traces provide details on the computational demand over time, providing a granular view of resource usage. Traces are crucial for driving simulations and allow a replay of real-world scenarios to predict system behavior, energy consumption, and CO2 emissions.

A model is an empirical prediction system that analyzes, combines, and computes atomic input elements to produce a comprehensive (sometimes exhaustive) output. We use models to predict real-world workloads run on ICT infrastructure with various specifications modeled by users. Models help understand and optimize resource allocation, workload management, and monitoring of certain overall performance metrics, such as energy consumption and CO2 emissions.

The export rate of the simulator represents the granularity at which the instrument samples and exports simulation data. For example, an export rate of 30 seconds will lead to 2 exported samples per minute. In this work, we address simulations with different sample rates, towards analyzing various metrics (e.g., performance) of the researched and developed tools.

Bitbrains-small[1] is a workload trace published by Solvinity spanning one month of operation across 1,250 virtual machines (VMs) [55]. This is a small-sized workload trace, sampled at 300 seconds, leading to a total of 8,683 timestamps, and occupies 2.2 MB of storage on the disk. The data is saved in *parquet* format.

SURF-LISA[2] is a workload trace, published by SURF, and spanning 7 days of computation jobs, run on SURF Lisa cluster, a High-Performance Computing datacenter in the Netherlands, consisting of 277 physical machines. The workload consists of 7,850 jobs with jobs duration ranging from minutes to several days [59]. This is a small-to-medium-sized trace, sampled at 30-second intervals and occupying 7.8 MB of storage on the disk. The data is saved in *parquet* format.

SURF-SARA[3] is a workload trace sampled from the Dutch national HPC and e-Science support center over 7 days, at a sampling rate of 30 seconds. The workload consists of 7,850 jobs, ranging from less than 30 seconds to over 5 days and run on a maximum of 2,592 central processing units (CPUs). This small trace occupies 6.1 MB of storage on the disk. The data is saved in *parquet* format.

ENTSO-E[4] is an open-source CO2 emission trace published by the ENTSO-E Transparacy Platform. For all the workload traces used in this work, we used CO2 traces samples from *ENTSO-E*, and adjusted for the time period. For example, *Bitbrains-small* is a workload run in 2012, while *SURF-LISA* and *SURF-SARA* were run in 2022. Thus, we use CO2 emission traces for the corresponding time period, with day accuracy, for 2012 and 2022, ranging from the starting timestamp of the first job and the ending timestamp of the last job in the workload.

OpenDC framework predicts simulation scenarios based on peer-reviewed models calibrated and trained with ground-truth, real-life data from BitBrains and SURF. Figure 6 provides a high-level architecture of the OpenDC framework (Figure 6); based on an input, the simulator predicts the overall energy consumed by the Central Processing Units (CPUs), using one or more CPU Power Models, further utilized by the Energy Model. The CO2 model predicts using the output of the energy model and further processes the output (prediction) data in files.

[1] https://github.com/Radu-Nicolae/opendc/tree/local-master/resources/bitbrains-small
[2] https://github.com/Radu-Nicolae/opendc/tree/local-master/resources/surf-lisa/trace
[3] https://github.com/Radu-Nicolae/opendc/tree/local-master/resources/surf-sara
[4] https://www.entsoe.eu/

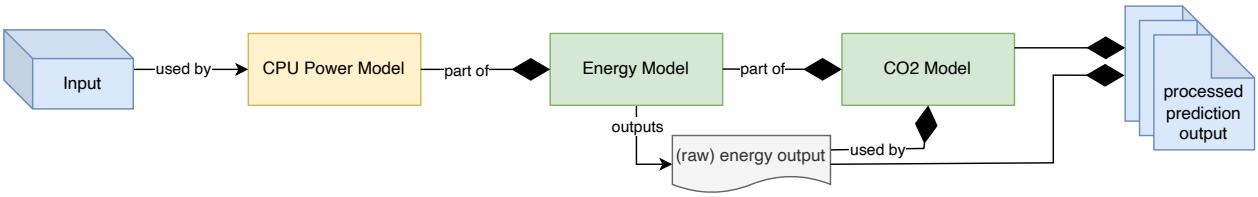


Figure 6: Prediction Architecture in OpenDC framework.

2.5 Energy Usage Models

Over the past decades, the number of datacenters has grown exponentially [68], responding to the high demand for (cloud) computing [54, 18]. Predicting the capacity of datacenters is a critical yet non-trivial simulation challenge [56, 7, 6]. Over the past decades, numerous datacenter simulators with various power models have been proposed, implemented, and evaluated by simulating workloads on different configurations for various use cases.

He et al. surveyed community-wide-recognized simulators based on a well-established survey technique [34]. The authors selected and implemented in OpenDC [56] nine power models, used for energy predictions. Further, they evaluated the built infrastructure against real-life data and proved the validity of the simulations.

2.5.1 Survey Method

He et al. established a well-defined survey workflow for selecting relevant energy models. Firstly, they selected *keywords* (❶), such as *"power provision model," "energy modeling,"* and *"datacenter energy simulation,"* to explore existing literature. Then, they searched literature (❷) via two main academic search engines (❷′, ❷″) and selected only articles with more than 500 citations (❸). Following, they checked whether the found modeling theories had been used to predict energy consumption in datacenter simulators (❹). Lastly (❺), they selected only the models matching both the criteria from ❸ and ❹.

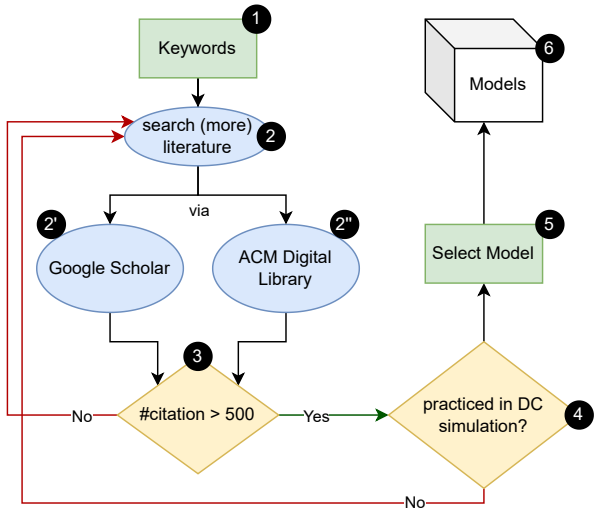


Figure 7: Energy Models Survey Workflow. Adapted from [34].

.

2.5.2 Selected Power Models

ConstantPowerModel	$P(c) = c$	(18)
LinearPowerModel	$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})u$	(19)
SquarePowerModel	$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})u^2$	(20)
CubicPowerModel	$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})u^3$	(21)
SqrtPowerModel	$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})\sqrt{u}$	(22)
MsePowerModel	$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})(2u - u^r)$	(23)
InterpolationPowerModel	$P(u) = P(u_1) + (P(u_2) - P(u_1))\frac{u - u_1}{u_2 - u_1}$	(24)
AsymptoticPowerModel	$P(u) = P_{\text{idle}} + \frac{(P_{\text{max}} - P_{\text{idle}})}{2}\left(1 + u - e^{-u/\alpha}\right)$	(25)
AsymptoticPowerModelDvfs	$P(u) = P_{\text{idle}} + \frac{(P_{\text{max}} - P_{\text{idle}})}{2}\left(1 + u^3 - e^{-u^3/\alpha}\right)$	(26)

where P_{idle}, P_{max} are the power used in idle and full capacity states, u is the CPU utilization, u_1 and u_2 are utilization points for interpolation, e is Euler's Number, α is the utilization fraction at which the host becomes asymptotic, r is the calibration parameter.

He et al. selected and integrated into OpenDC nine power models, synthesized from simulation-related literature published in Computer Systems venues in the past two decades [11, 13, 14, 15, 25, 31, 32, 35, 42, 44, 47, 49, 51, 53, 60, 77, 78, 79, 34].

Calheiros et al. proposed CloudSim, a widely-used, self-contained toolkit for cloud computing modeling and initial performance testing. CloudSim can support large-scale simulation environments and offers highly customizable settings for various cloud components, such as virtual machine allocation rate to physical machines. CloudSim's open-source nature allowed researchers to analyze, extract, and implement various models used for power consumption prediction in other simulation frameworks. CloudSim-Plus, introduced by *Filho et al.*, provides a superior application codebase using better software engineering patterns, practices, and recommendations. However, no conceptual difference exists between the energy prediction models from CloudSim[13] and CloudSim-Plus[67].

Equations (19), (20), (21), (22), (24) represent five of the energy models of CPU consumption, provided by CloudSim, and integrated into OpenDC.

Fan et al. proposed the Mean Square Error (MSE) power model (Equation (23))[24]. Although simple, this model has been widely adopted in simulating energy consumption in datacenters [34]. *Fan et al.* demonstrated the sub-/super-linear correlation between the power and the CPU frequency, also reflected in the MSE model. This model has also been integrated into OpenDC.

In this work, we focus on models (19), (20), (21), (22), due to their simplicity and common usage, with proven accuracy and precision, matching high-quality industry standards [13, 67, 34].

2.6 CO2 Emissions Models

Niewenhuis et al. proposed FootPrinter [59], a "first-of-its-kind tool that supports datacenter designers and operators in assessing the environmental impact of their datacenter." As part of their research, engineering, and evaluation, matching the state-of-the-art AtLarge Design Process [40], the authors proposed and integrated in OpenDC a model able to predict the CO2 emissions of the simulated infrastructure.

FootPrinter model employs in its simulation peer-reviewed energy models, integrated in OpenDC, and expanded in Section 2.5. We present in Figure 6 the overall simulation structure of OpenDC, and the embed of the energy models into the CO2 prediction models. *Niewenhuis et al.* calibrated the researched CO2-emissions model with data from the ENTSO-E Transparency Platform. The efforts have been concertized in a research paper, part of a top-tier conference. This confirms the validity of the experimentation results, and the accuracy of the developed model.

2.7 Single-Model Simulation in ICT

To meet the unprecedented high demand for cloud computing, numerous simulators have been proposed during the past decades [39, 56, 60, 13, 51, 78, 49, 67]. While each simulator has its own properties, posing advantages and drawbacks, the core simulation architecture is consistent among all the simulation frameworks proposed so far.

In Figure 8, we present a simplified, high-level perspective of the architecture employed by current state-of-the-art simulators. We present a critical background concept, which suggests simulating using only one simulation model. In Steps (A), (B), and (C), the user sets the experiment, precisely a datacenter configuration, workload, and traces, leading to the ready-state of the setup (F). The simulator runs the configured experiment (G), using a single simulation model (H) and then obtaining raw simulation results (I). After the simulation, the tool processes the simulation results (J) and generates output.

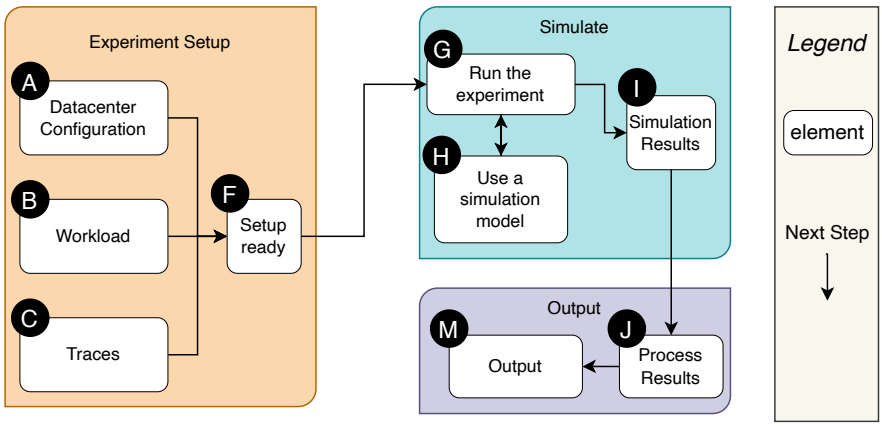


Figure 8: Simuation using a single model. The current state-of-the-art.

Simulators relying on idiosyncratic models in their simulations are prone to errors when edge cases are encountered. Throughout this paper, we propose novel simulation concepts that contrast with the existing architecture of the simulators (Figure 8). In Section 3.4, we present an architecture that uses multiple models in the simulation process and compare it against the existing architectures, which rely only on singular models.

2.8 Multi-Model Simulation in Other Sciences

The use of predictive models has become increasingly important in various scientific disciplines. Researchers proposed simulation instruments leveraging the multiple models to recognize individual models' limitations in capturing edge cases. While this approach has been successfully applied in disciplines such as Virology [21, 63], Environment [43], Ecology [43], and Weather Prediction [65, 46], simulation leveraging multiple models has never been applied in Computer Science.

2.8.1 Multi-Model in Environmental Sciences

A study by *Myhre et al.* investigated the geographical distribution of environment-harmful emissions, e.g., nitrogen oxides, sulfur dioxide, black carbon, and organic carbon. The authors used seven global models, sourced from the EU project ECLIPSE, all fed with the same datasets and unified under a single representation. Alongside presenting the individual predictions of each model, the researchers also developed a new model that aggregates and averages the predictions of all other models. This approach mitigates the effects of erroneous individual model deviations and provides more robust and comprehensive predictions [58].

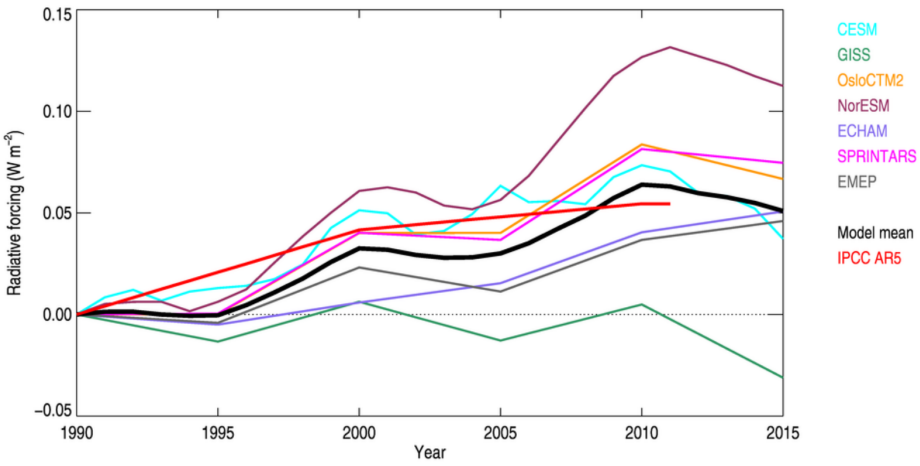


Figure 9: Multi-Model Simulation for "Radiative Forcing (Wm^{-2}) of the direct aerosol effect over the period 1990-2015 given seven models (legend lists the models); the multi-model mean is shown in black and the estimate provided in IPCC is included in red." Figure from [58].

The authors use as a benchmark for comparison the IPCC AR5 model, presented by the Intergovernmental Panel on Climate Change [26]. The IPCC AR5 is considered to be a state-of-the-art simulation model in the field of environmental sciences, with highly accurate and precise simulations [26, 58]. When comparing the derived model (mean of the other models) with IPCC AR5, the derived model's overall predictions are closest to those of the IPCC AR5 model, significantly closer than the predictions of the individual models.

2.8.2 Multi-Model in Covid-19 Pandemic Prediction

A study by *Cramer et al.* presents a notable usage of the multi-model approach, merging the Covid-19 models to gain deeper insights into the progression of the pandemic. The authors developed *"The COVID-19 Forecast Hub."* Although initially launching with only 10 models in April 2020, by May 2022, over 93 primary models were integrated into the tool. This web application enables the selection of multiple models and overlays their predictions in a single visualization, facilitating the identification of the most prevalent prediction (Figure 10, Figure 11). The study's primary objective is to provide a comprehensive, easy-to-understand visual prediction of how the pandemic will evolve while presenting multiple model predictions.

The built prototype allows users to choose simulation models from a set. A user can select one or multiple models overlapping the simulation graph. We tested the prototype by selecting two models (Figure 10), and by selecting seven models (Figure 11). The interface provided insights into how the output can be given and helped us design our Multi-Model simulator. The output also gave the US government insights on how the pandemic will evolve and, therefore, support officials in proactively planning and responding to the pandemic on national and regional scales [21].

2.8.3 Seasonal Influenza

Before the COVID-19 pandemic, in 2018, a notable study conducted by *Nicholas et al.* explored the application of multiple models and data sources to accurately forecast the progression of seasonal influenza in the US. The authors compared the predictions with the real-world epidemic data and assessed the accuracy and precision of the models using a standardized evaluation framework. The study identified Delphi-Stat as the most effective model; Delphi-Stat employs an ensemble approach that combines and weighs models from the Delphi group.

Albeit already valuable, the authors outlined that there is still an extensive, non-trivial effort needed to be made for model enhancement; the authors also underscored the substantial challenge posed by the low-quality data, as the presence of gaps in the datasets impacted the performance of these models [63].

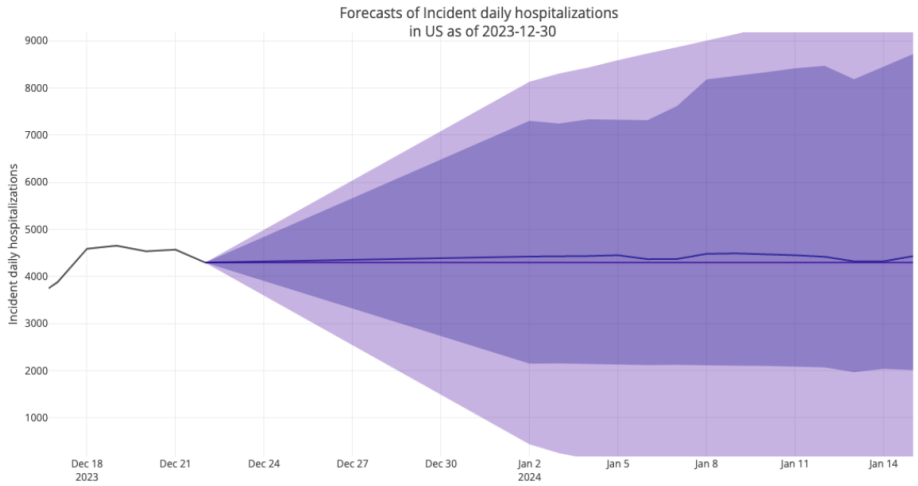


Figure 10: The Forecast Hub's COVID-19 prediction with two selected models. (Source [20]).

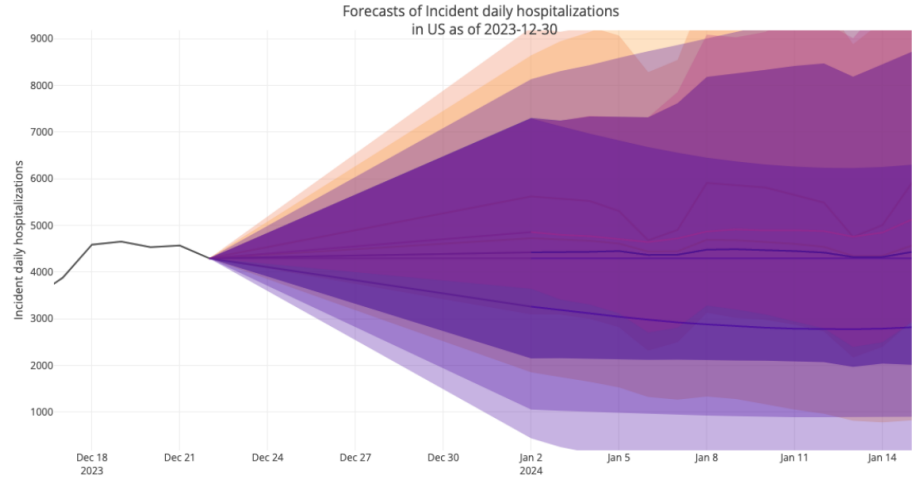


Figure 11: The Forecast Hub's COVID-19 prediction with seven selected models. (Source [20]).

3

Design of a Multi-Model Simulation Approach for Datacenters

In this chapter, we address the first research question (RQ1) by presenting the design of a simulation system that leverages the results of multiple models into a unified tool.

3.1 Overview

We design the Multi-Model simulator, the first tool to leverage multiple simulation models in ICT. Throughout this chapter, we match the state-of-the-art AtLarge Design Process [40]. Our contribution in this chapter is seven-fold:

- We establish the functional and non-functional requirements for the Multi-Model in Section 3.2.
- We analyze conceptual design choices for the Multi-Model integration in a black-boxed datacenter simulator, in Section 3.3.
- We then propose a high-level design for the Multi-Model architecture in Section 3.4.
- We present the detailed design of the Multi-Model in Section 3.5.
- We analyze how the aggregation of chunks of data influences the Multi-Model granularity, visual comprehension, and performance in Section 3.6.
- We address, individually, each established requirement in Section 3.7.
- We make the Multi-Model tool publicly available on Github[5].

We summarize the contributions of this chapter in Section 3.8.

3.2 Requirements Analysis

In this section, we determine the functional and non-functional requirements that the Multi-Model should address. This matches stage 1 of the AtLarge Design Process [40].

3.2.1 Functional Requirements

Main Functional Requirement (MFR): Leverage simulation output into a unified tool.

(**FR1**) **Support comprehensive data visualization.**
The system should enable users to visualize data predicted by the models comprehensively. The system should support time series, cumulative plots, and cumulative time series plots from which users can choose. The system should also allow users to scale data to the appropriate magnitude of the unit prefix of measurement (e.g., 1,073,741,824 units would be converted to 1.073 giga-units). The scaling feature can be enabled/disabled by the user. Without FR1, Multi-Model cannot leverage simulation output into a visual unified tool (part of MFR).

[5] `https://github.com/Radu-Nicolae/opendc/tree/local-master`

(**FR2**) **Enable single-, multi-, and any-metric simulation.**
The system should allow users to select which metrics to analyze using the Multi-Model. The system should enable users to run single- and multi-metric-based simulation analyses. Besides, the system should provide universality (FR3) and enable users to run any-metric analysis (i.e., we do not restrict users to analyzing only pre-defined metrics). Without the possibility of choosing which metrics to analyze, the Multi-Model simulator would not be able to efficiently, computationally effectively, and rapidly provide simulation analysis for very large-scale simulation (NFR1). Without allowing any-(custom-)metric analysis, we reduce the tool's universality (FR3) and limit the simulator's functionality to only specific metrics.

(**FR3**) **Facilitate tool universality and integration with any datacenter simulator.**
The system should integrate with any datacenter simulator without modifying the simulator's core architecture. The Multi-Model should be applied as a top layer and incorporated into users' existing workflows with minimal effort. Integrating the Multi-Model within an existing datacenter should only involve parsing the current simulator's output and providing input to the Multi-Model as required in the documentation. Without FR3, we hinder the adoption of the Multi-Model tool and concept.

(**FR4**) **User can select the outputting granularity.**
The system should allow users to select the window size (i.e., convolution size) at which the Multi-Model analyses and predicts. This feature raises trade-offs between the window size, the analysis time, and the granularity of the prediction. Without FR4, users could not receive comprehensive visual output (FR1) and could not adjust the analysis based on the allocated simulation-analysis time (NFR1).

3.2.2 Non-Functional Requirements

In addition to the functional requirements, we determine three non-functional requirements for the Multi-Model:

(**NFR1**) **Provide in-meeting, near-interactive, same-day simulation results.**
Cloud infrastructure currently operates at an unprecedented scale [55]. The system should run efficiently, output the simulation results on time, and support very large-scale cloud environments and workloads. The Multi-Model output process (i.e., compute, plot, save the plot to disk) should not take longer than running the simulation itself with the OpenDC simulator. NFR1 should be met on datasets of at least $100,000$ samples on a regular-user machine (i.e., not a supercomputer). $100,000$ samples represent 347 computing days at the industry-standard sampling rate of 5 minutes [57, 6]. Without NFR1, the Multi-Model cannot be reasonably used in interactive settings or for large-scale infrastructure.

(**NFR2**) **Facilitate reproducible science and experimentation.**
The results produced by the Multi-Model should be fully reproducible, and the system should minimize the effort of the users who attempt to reproduce results. The experiment setups, scripts, and results should be publicly available. Without NFR2, we weaken the confidence in the results presented in this work.

(**NFR3**) **Leverage the simulation data of at least 4 individual models.**
The simulation output of individual models should be leveraged into a unified simulation tool. The Multi-Model should be able to efficiently leverage up to 4 models and comprehensively visually output their predictions into unified visuals. Albeit the defined benchmark of 4 models, the Multi-Model should be able to scale up to more leveraged models. However, this may break other functional and non-functional requirements (e.g., clear, comprehensive outputted plots, rapidly computed results of the Multi-Model). Without NFR3, the Multi-Model could not serve the user at the full potential provided by the employed simulation concept.

3.3 Conceptual Design Choices

In this subsection, we describe the conceptual design choices of M3SA and the adopted system architecture. In Section 3.3.1 and Section 3.3.2, we identify two high-level system designs; in Section 3.3.3, we compare the two proposed architectures based on universality (FR3) and theoretical performances (NFR1). The other requirements (e.g., reproducible science) can be facilitated with either of the proposed architectures.

3.3.1 Simulate First, Compute Later (SFCL)

In this subsection, we identify and analyze a design choice in which all user-selected individual models are run first, the results are saved in external files, and the Multi-Model is computed using the output files. We term this architecture *SFCL* (simulate first, compute later). *SFCL* has a minimal impact on the core architecture of the simulator and is applied as a top layer.

We visually illustrate SFCL in Figure 12. The simulation is set up in step (A). In the simulation process, the first model is run (B), and the simulation results are saved on the disk in output files (C). Then, the simulator checks for remaining models to be run (D). If more pending models are found, the instrument runs simulations until all the user-requested models are covered.

Following, a Multi-Model is initialized in step (E), and the output data of the individual simulation models is leveraged in the Multi-Model sequentially, as illustrated by steps (F),(G),(H), and (I). After all the simulation data is appended, the Multi-Model is formed (J), and output is created (K). In the output process, the Multi-Model plots, following the user customization (from (A)).

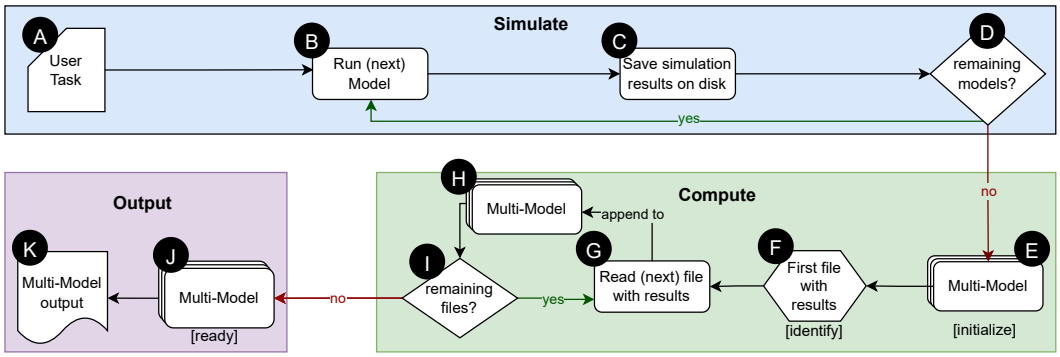


Figure 12: SFCL (Simulate First, Compute Later) High-Level Architecture.

3.3.2 Compute While Simulating (CWS)

In this subsection, we identify and analyze a design choice where each model the user selected is run, and its results are appended to the Multi-Model. We regard this as sequential, where the Multi-Model is gradually built after every simulation is finished. We term this architecture *CWS* (compute while simulating). *CWS* has a major impact on the core architecture of the simulator, as it needs to be integrated within the core simulation components.

We visually illustrate *CWS* in Figure 13. The user defines a *User Task* (A) for the simulation instrument, and the simulation process begins. In this architecture and the *Multi-Model* (B) is initialized in the setup process. Following, the first model predicts (C), and simulation results are outputted (D); the model's predictions are appended to the *Multi-Model* (E).

After the first model is appended to the Multi-Model, the system checks if there are any remaining models to be simulated (F); if more models are pending, they are run, and their results are appended sequentially to the Multi-Model. After the last model is run, its results are further appended to the Multi-Model, and

ultimately the *Multi-Model* (G) is ready. In the output process ((H)), the Multi-Model plots, following the user input from Step (A).

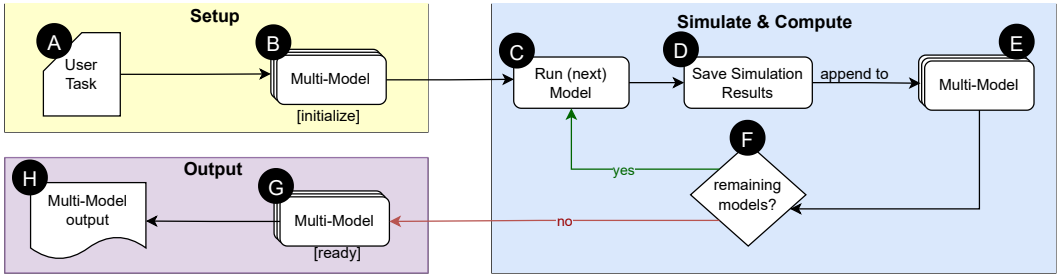


Figure 13: CWS (Compute While Simulating) High-Level Architecture.

3.3.3 SFCL compared to CWS

In this subsection, we compare *SFCL* and *CWS* and then analyze their properties against the functional requirements.

The main conceptual difference between *SFCL* and *CWS* is the stage of the simulation when data is appended to the Multi-Model. In *"Simulate First, Compute Later,"* the individual models' data is appended to the Multi-Model only after all the individual models finish computing. In contrast, in *"Compute While Simulating,"* data is appended to the Multi-Model immediately after an individual model finishes computing.

In terms of universality (FR3), we observe that *SFCL* requires minimal integration effort because it functions as a top layer, applied on top of the existing codebase of the simulator. The integration process of *SFCL* requires provisioning the Multi-Model with the format of input needed, as the integration documentation[6] requires. Therefore, no changes are necessary to the core architecture of the simulator for *SFCL*. In contrast, *CWS* design entails altering the simulator's core components; in the *CWS* architecture, after each simulation, the simulator needs to pause the simulation process, append to the Multi-Model, and then resume. Considering these, we regard *SFCL* as addressing the universality functional requirement (FR3) better than *CWS*.

In terms of performance (NFR1), *SFCL* and *CWS* provide similar theoretical performance, depending on the implementation. However, as an in-system-deeply-embedded tool, *CWS* is more flexible and open to performance enhancement. In contrast, *SFCL*, applied only as a top layer, cannot facilitate in-simulator embed, which makes the performance optimization process more challenging. Considering these, we regard *CWS* as more embedded within the simulator and, hereby, simpler to improve the tool's performance and obtain a simulator able to *"provide in-meeting, near-interactive, same-day simulation results"* (NFR1).

We determine a trade-off between the level of integration and the ease of improving the simulator's performance; the more integrated the tool, the easier it is to enhance performance, yet the more challenging it is to re-use the tool on a different simulator.

Considering the analysis from the previous paragraphs, we regard *SFCL* as an architecture open to universality, simple to apply to any simulator (FR3). Albeit the efficiency improvement challenges presented, we consider *SFCL* more suitable than *CWS* for this work, as long as the non-functional requirement NFR1 of computing, plotting, and outputting the Multi-Model in maximum the amount of time allocated for the simulation, is met. Therefore, we choose *SFCL* as the architecture for the Multi-Model and for the holistic system.

[6] `https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/M3SA-integration-tutorial.md`

3.4 Overview of Multi-Model

In this subsection, we present the conceptual and technical differences between the current state-of-the-art ICT simulators (expanded in Section 2.7) and the proposed state-of-the-art, *the Multi-Model vision.*

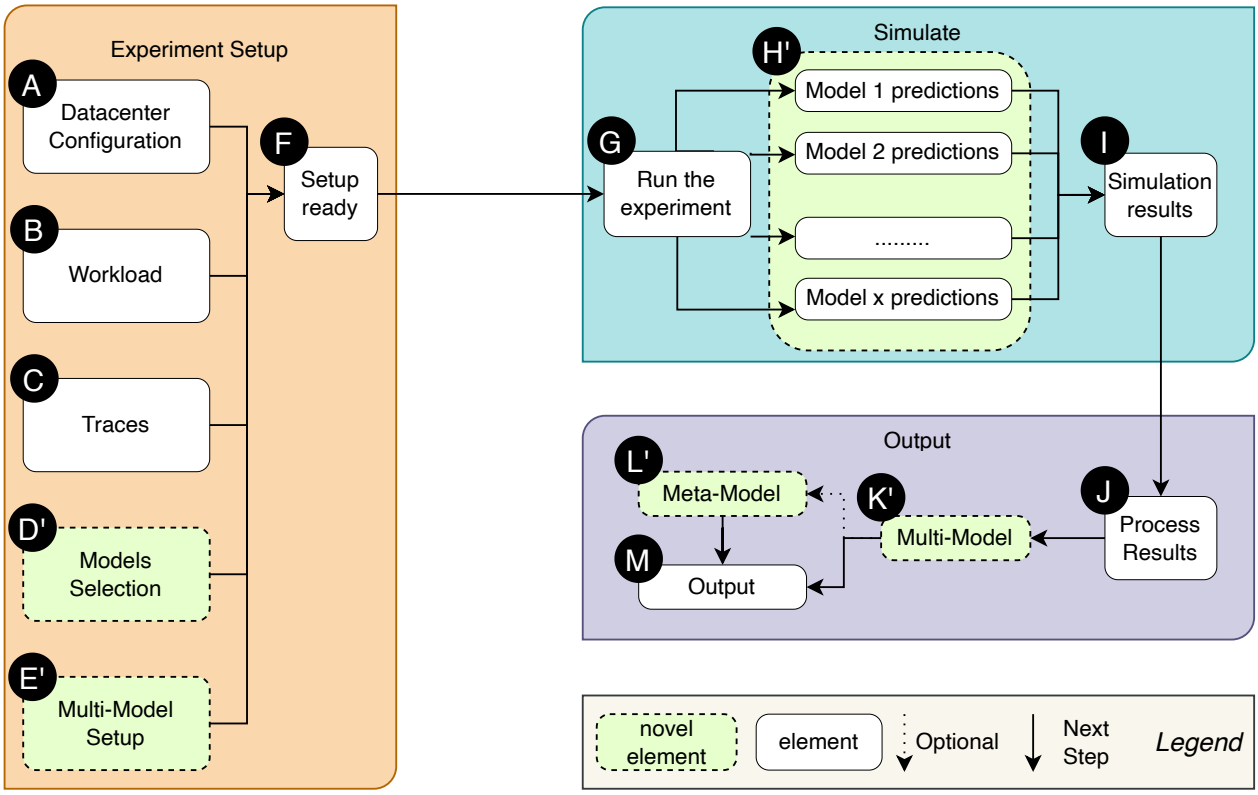


Figure 14: Simulation using multiple models. The proposed state-of-the-art.

We outline the differences between the existing and the proposed simulation architectures in Figures 8 and 14. For a clearer comparison, we label equivalent components with the same letter. In the *Experiment Setup* component, steps A, B, and C are similar for both architectures. In the Multi-Model vision, in the experiment setup stage, the user additionally selects the desired models for simulation (D') and sets up the Multi-Model (E').

The core difference between the two simulation architectures is represented in the *Simulation Stage*, between steps H and H'. In the existing architectures, the experiments (G) use a single simulation model (H). In the envisioned architecture, the experiment (G) uses multiple simulation models, run independently one from another, as depicted in step H'; step H' is a novelty introduced by the Multi-Model vision.

In both architectures, simulation results are generated and processed for output. However, the Multi-Model architecture also includes outputting the Multi-Model (step K'). Optionally, a Meta-Model can be computed (step L') on the user's choice. Lastly, the results are outputted in both architectures in step M.

3.5 The Multi-Model Process

At the core of the Multi-Model is a novel simulation concept, which leverages the results of individual simulation models into a unified, discrete simulation tool with detailed, granular modeling of energy operations and sustainability. In Figure 15, we depict an overview of the Multi-Model architecture, integrated and

applied to a black-boxed simulator as a top layer. Following the AtLarge Design Process [40], we construct the Multi-Model and M3SA architecture iteratively. This process begins with bootstrapping the creative process (stage 3), after which we focus on the high-level and low-level design (stage 4) [40, 55].

In this subsection, we provide a high-level integration of the Multi-Model upon a black-boxed simulator to emphasize the universality of our work (FR3) and present each relevant component depicted in Figure 15.

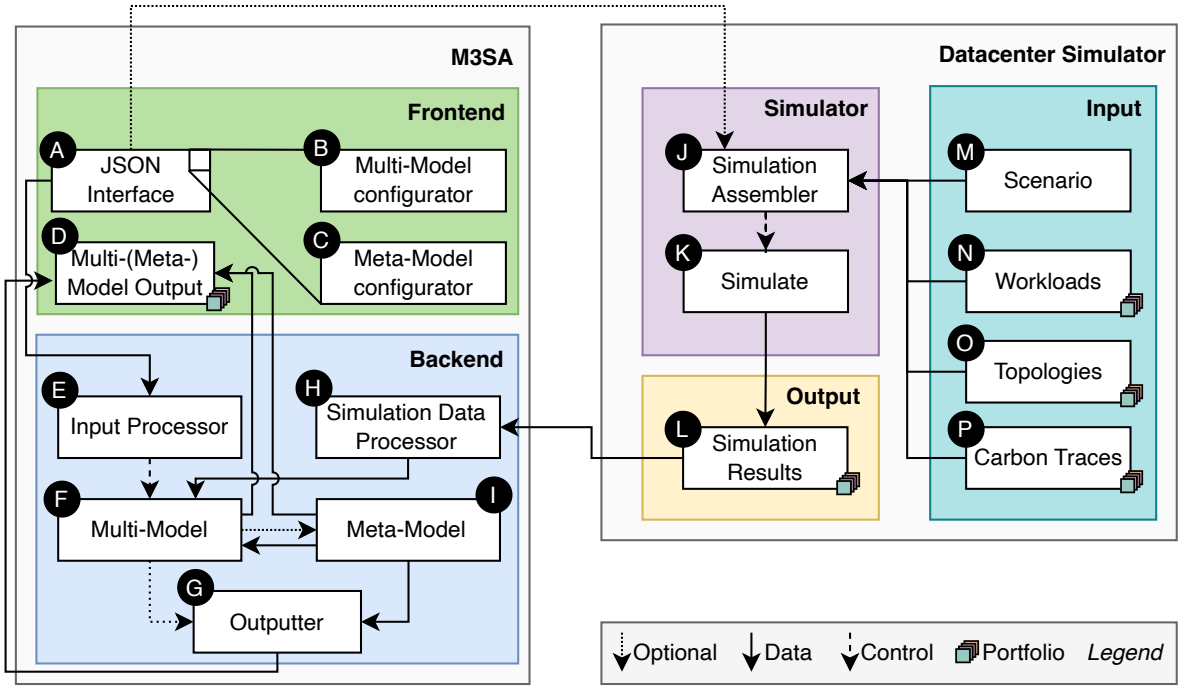


Figure 15: High-level architecture of M3SA integration within a datacenter simulator.

In the input box, users configure the simulation by providing a *Scenario* (M), a *Workload* to be executed (N), one or more *Topologies* (O) (the infrastructure configuration), and a *Carbon Trace* (P). This represents the simulation setup, which is further redirected to the simulator. The input data is assembled by a *Simulation Assembler* (J). The core-simulation process begins as part of the *Simulate* (K) component. The *Simulation Results* (L) are further saved in one or more files as part of the output process. The black-box nature of this high-level architecture facilitates the universality of our work and integration with any datacenter simulator (FR3).

The *JSON Interface* (A) represents the interaction component between the user and the simulator. The user can configure M3SA before or after the individual models' simulation. Via the *JSON Interface* (A), users can construct the *Multi-Model* (B) and *Meta-Model* (C), and establish functionality, properties, and appearance. We expand the Meta-Model in Section 4. In step A, users can choose and customize the type of plot outputted by the Multi-Model (FR1), select the measured metrics (FR2), and select the granularity of the output (FR3). Users can leave empty placeholders, and default values are used.

Next, the data received through the *JSON Interface* (A) is handled by the *Input Processor* (E). This component parses the user input and sets up the back-end components. After this step, the *Simulation Data Processor* (H) retrieves the outputted simulation data, reads, and links the files with the M3SA's Backend component. This step is a crucial layer between the outputted simulation data and the M3SA process, critical in addressing the main functional requirement (MFR) of *leveraging simulation output into a unified tool.*

After reading the raw simulation data, the *Multi-Model* component (F) processes the data using the user's input (or default values, if no input is provided) and builds a Multi-Model at the selected granularity. After

the computation, the Multi-Model generates and saves plots as configured by the user.

Optionally, the user can select to compute a Meta-Model. If the user chooses this option, component Ⓘ is activated, generating a new model. The Meta-Model predicts based on the predictions of the individual models. The *Meta-Model* (Ⓘ) is built on top of the *Multi-Model* (Ⓕ) and computed only after the Multi-Model is assembled. The process of Meta-Model simulation will be further expanded in Section 4.

The *Outputter* component (Ⓖ) is used by the system to plot data of the Multi-Model or to both plot and save meta-simulation data, the user-selected output folder.

In this work, the Multi-Model builds upon OpenDC, an open-source platform for cloud datacenter simulation, built through 7+ years of development and operation [39, 55, 56]. This enables the Multi-Model simulator to leverage OpenDC's capabilities for datacenter modeling and allows other users of OpenDC to benefit from Multi-Model's functionalities. We apply this tool to OpenDC in Section 5. Although built upon OpenDC, the Multi-Model can be applied as a top layer to any datacenter simulator that follows the integration requirements.

3.6 Window-Size Analysis

We define a *window* as an aggregation of chunks of data into a singular data entry using a pre-defined function. A window functions similarly to a one-dimensional convolutional layer, with a size indicated by a given parameter. The window-size is an essential variable in addressing the visual comprehension of the Multi-Model (FR1), providing time-efficient output (NFR1).

In Figure 16, we present how a window of size is applied on n data entries. In this example, the window has a size of 5. Hence, each group of 5 consecutive data entries is fed into an aggregation function F. The function's output is then appended in an output array with the size of $\lceil n/5 \rceil$. We round the division results to the next integer for cases where the number of data entries is not a multiple of the window size.

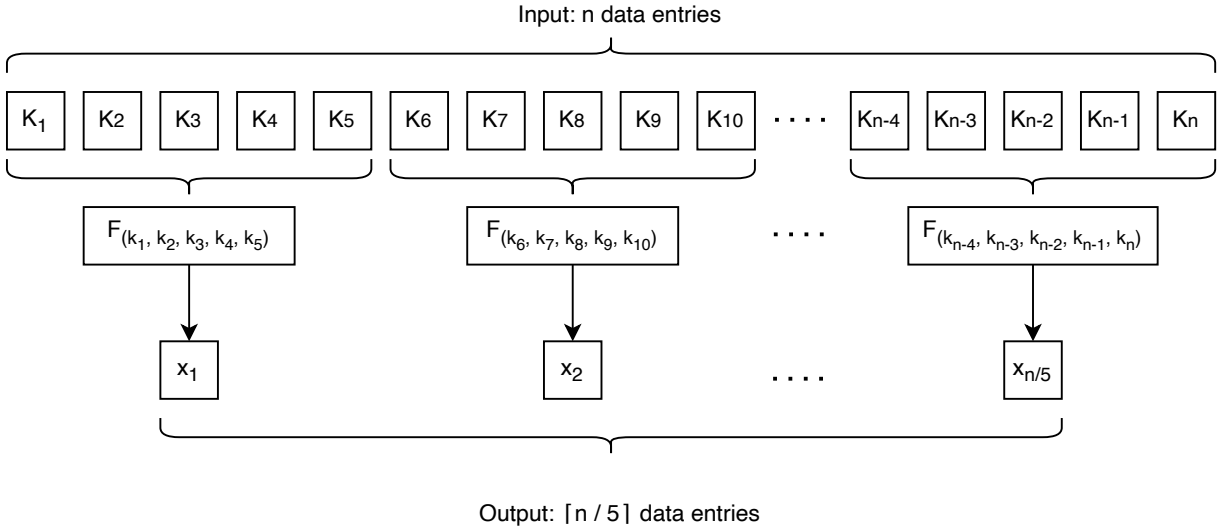


Figure 16: Window of size 5 applied on n data entries, using function F as aggregation function.

3.6.1 Trade-off between window-size and visual comprehension

In Figure 17, we depict four plots on 5,000 data entries processed by the Multi-Model, using four window sizes. In Figure 18, we visually represent 2,500 prediction data entries processed by the Multi-Model with the same window sizes as in Figure 17, leveraged in the same graph.

Experiment 1 configuration

Figure 17 and Figure 18 are generated using the output data of Experiment 1[7], which simulated a medium-scale facility under the SURF-SARA workload, using OpenDC simulator. The facility contains 16 Clusters and 32 hosts per cluster. Each host contains 1 TB of memory and 1 Intel Xeon Sapphire Rapids, with 32 cores operating at 2.6 GHz. Figure 17 and Figure 18 plot the first 5,000 and 2,500 data entries, respectively.

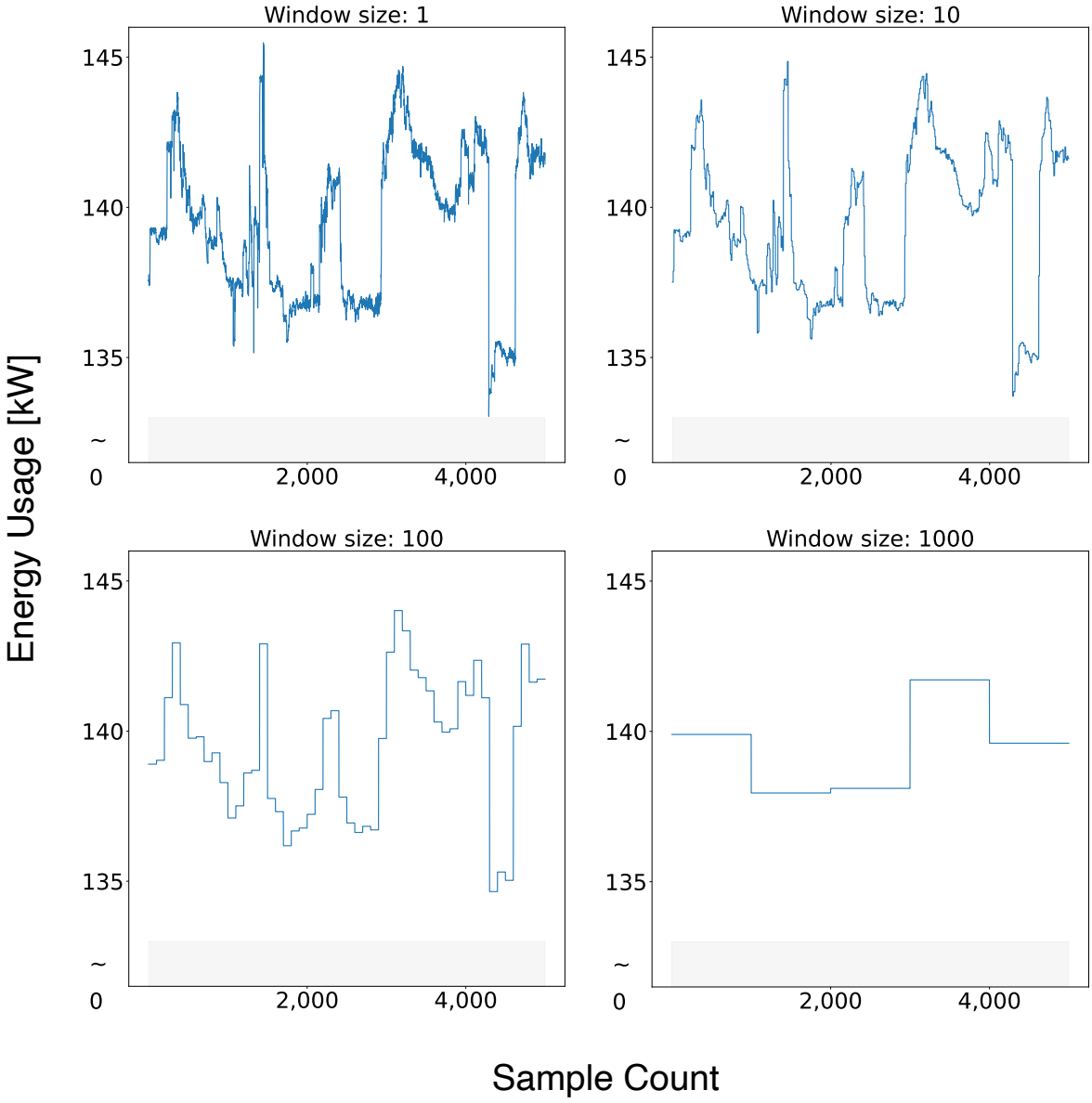


Figure 17: 5,000 simulation data entities plotted with window sizes of 1, 10, 100, and 1,000.

In Figure 17, we observe the visual differences between data processed with window sizes of 1 (i.e., raw, unprocessed data), 10, 100, and 1,000. We observe a trend of increased data noise as the window size is smaller. In contrast, we remark a clearer, more visually comprehensive plotting when the granularity of data is smaller. However, a window size of 1,000 for a dataset of 5,000 samples outputs only five samples, which is insufficient for accurately understanding the simulation results.

[7] https://github.com/Radu-Nicolae/opendc/tree/9-experiments/demo/experiments/experiment-1-window-analysis

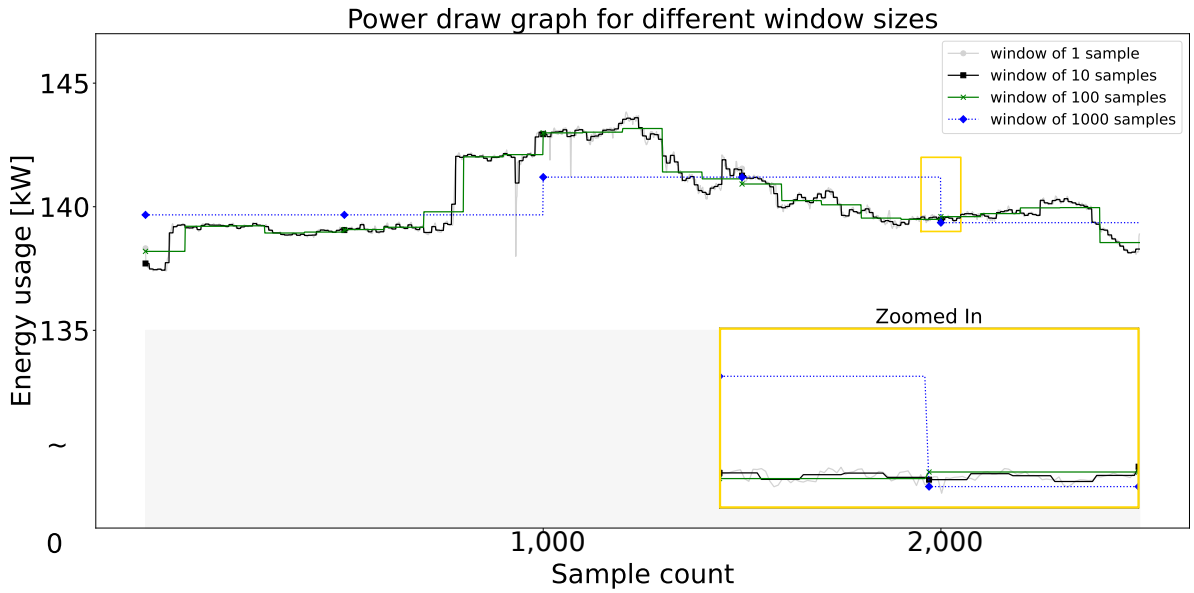


Figure 18: 2,500 simulation data entities, processed with 1, 10, 100, and 1,000 window sizes, leveraged in the same graph.

Figure 18 provides an alternative visual window sizes comparison by juxtaposing the data processed with aggregating 1, 10, 100, and 1,000 samples for a dataset of 2,500 samples. We zoom in for 100 data entries and attach the zoomed-in plot in the bottom right of the figure.

3.6.2 Trade-off between window-size and Multi-Model generation time

Besides the trade-off between the window size and the graph's visual comprehension, we identify a trade-off between the size of the window and the time required to compute, plot, and output the Multi-Model. In this section, we analyze the trade-off window-size—time to generate the Multi-Model.

To facilitate reproducible science (NFR2), we make the experiment configuration, results, and analysis publicly available on the GitHub repository[8] of the Multi-Model. We run the Multi-Model on datasets of various sizes, containing simulation data from the same simulation as in Section 3.6.1. The experiments are run on the same machine without external user tasks running in parallel.

Experiment 2 configuration
We ran this experiment on an Apple MacBook Pro, 2023, 16-inch, M2 pro chip, connected to power. We simulate a medium-scale facility under the SURF-SARA workload. The facility contains 16 Clusters and 32 hosts per cluster. Each host contains 1 TB of memory and one Intel Xeon Sapphire Rapids, with 32 cores operating at 2.6 GHz. We run the same simulation using OpenDC, using four different power models, namely *"sqrt"* (equation (22)), *"linear"* (equation (19)), *"square"* (equation (20)), and *"cubic"* (equation (21)). Under the same simulation, we used four different sampling rates (i.e., the granularity at which the state of the infrastructure is checked), which resulted in data files of varying size magnitudes. These Multi-Model configurations generate time series plots.

[8] `https://github.com/Radu-Nicolae/opendc/tree/9-experiments/experiments/experiment-2-window-performance-analysis`

	Multi-Model Time [s]			
	d=2,016	d=10,080	d=20,160	d=201,600
w_1	0.8 s	4.4 s	9.8 s	264.7 s
w_{10}	0.4 s	2.6 s	5.9 s	187.8 s
w_{100}	0.4 s	2.5 s	5.5 s	220.0 s
w_{1000}	0.4 s	2.6 s	6.0 s	238.3 s

Table 1: Comparison of Multi-Model computing, plotting, and outputting times for different window sizes. d represents the number of samples in the simulated dataset. w_1, w_{10}, w_{100}, w_{1000}, represent windows of sizes 1, 10, 100, and 1,000. The Multi-Model Graph Computing Time is measured in seconds [s], and time series plots are generated.

We notice a significant performance improvement when using a window size of 10 compared to no window (i.e., a window size of 1) in Multi-Model computation, plotting, and outputting time. We observe a trend of less Multi-Model generation time for window sizes of 10 and 100 compared to the time required to generate Multi-Models for window sizes of 1 and 1,000. Computing a Multi-Model with a window size of 1,000 requires aggregation of a large number of data chunks, which need to be stored in memory, summed up, and then divided; this results in slower operation than smaller window sizes. In contrast, not applying any window size keeps the dataset raw, which results in plotting a very large number of data entries; this results in slow operation as well.

We conclude that the trade-off between granularity and performance is valuable when choosing a window size between 1 and 10, but not worth it for higher magnitudes in terms of performance. However, regarding visual comprehension, users may find larger window sizes more suitable despite losing granularity.

3.7 Requirement Addressal

In this section, we present how this work addresses each functional and non-functional requirement.

For FR1 and FR2, we use data from a simulation of a small-scale facility run under the SURF-SARA workload and using the ENTSO-E carbon trace. The facility contains 4 clusters, with 8 hosts per cluster. Each host contains 64 GB of memory and one Intel Xeon Sapphire Rapids CPU with 32 cores operating at 2.6 GHz. We run the same simulation using OpenDC, using four different power models, namely *"sqrt"* (Equation (22)), *"linear"* (Equation (19)), *"square"* (Equation (20)), and *"cubic"* (Equation (21)). The experiment (labeled Experiment 3) is publicly available on Github[9].

3.7.1 Support comprehensive data visualization. (FR1)

The system supports comprehensive data visualization and enables users to visualize data predicted by models in plotted graphs. Users have extensive customization options for the Multi-Model output, such as selecting the window size, type of plot, vertical and horizontal axis limits, units of measurement, labels, and titles. The system supports time series, cumulative, and cumulative time series plots, allowing users to select their preferred plot type.

We meet the functional requirement of supporting time series, cumulative, and cumulative time series plots, from which the user can select. We present all three supported plots in Figure 19. We present, per plot, the consequences of different window sizes, which can be selected by the user, towards meeting the needs of the project (e.g., visual constraints and time limits).

In Figure 19, we visually represent the results of Experiment 3, using window sizes of 1, 10, and 100, and plotting as time series plot, cumulative plot, and cumulative time series plot. The time series plot displays the energy consumed at each infrastructure sampling, while the cumulative plot shows the total amount of energy consumed by the infrastructure. The cumulative time series plot represents the total energy consumed by the datacenter at various granularities during the experiment.

[9] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-3-multi-model-all-plots`

Users can also select the unit and the data scaling magnitude. In Figure 19, we show the difference in scale between the cumulative time series plots (measured in megawatts) and the time series plots (measured in kilowatts). In the time series plots, we observe a decrease in data noise as the window size increases. We apply no window for the cumulative plots (hence no aggregation function), which minimizes the computation efforts for the Multi-Model. We emphasize that the cumulative data remains unchanged, independent of the window size. For cumulative time series graphs, we observe a staircase-like pattern, where the length of each step increases with the window size. We also observe that the granularity is so high for graphs depicting data in windows of 1 and 10 chunks that the plotted data appears as a line.

The Multi-Model provides extensive customization options, all of which are documented as part of the Multi-Model documentation[10]. We also provide a schema for the input JSON file[11].

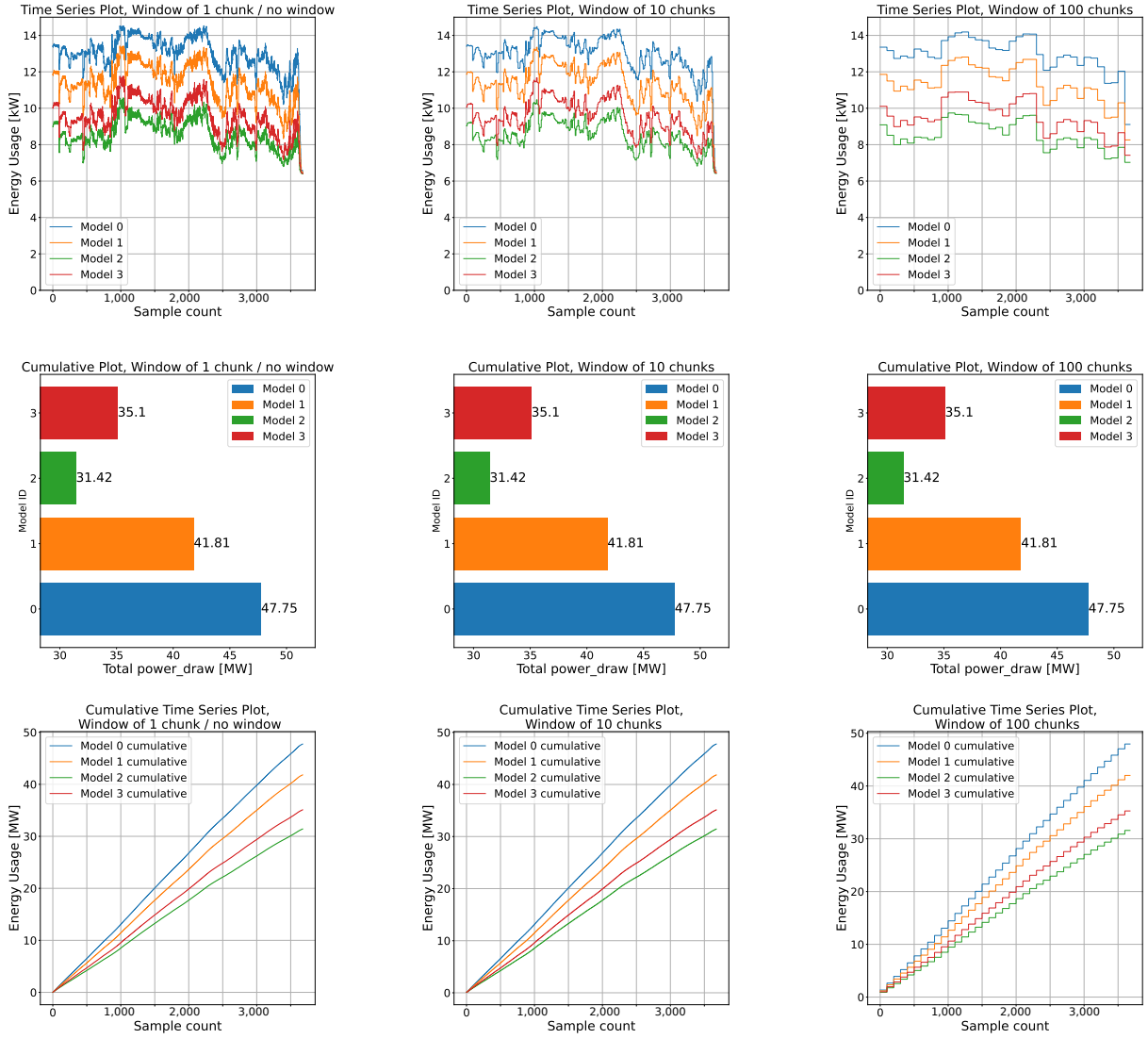


Figure 19: Multi-Model-supported plots, with 1, 10, and 100 window sizes.

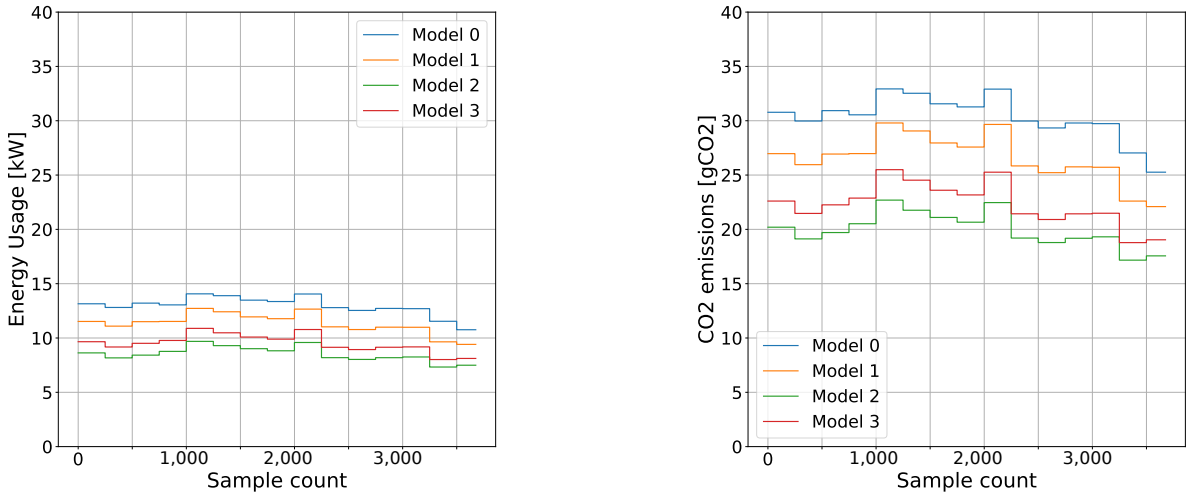
[10] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/Input/MultiMetaModel.md

[11] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/Input/MultiMetaModelSchema.md

3.7.2 Enable single- and multi- and any-metric simulation. (FR2)

Datacenter simulation is a complex task that inputs, analyses, and predicts numerous metrics [55]. We identify and address the requirement of analyzing only the user-selected metrics instead of conducting multi-model simulations for all available metrics. We design and develop the Multi-Model simulator, which can simulate individual metrics chosen by the users. This ensures that the simulator can perform both single- and multi-metric simulations.

We present the Multi-Model's capability of conducting single-metric analysis by plotting outputted simulation data as part of Experiment 3[12]. For instance, if a user desires to analyze the energy usage of the infrastructure, they can select this specific metric as the input. After simulating using multiple models, the system generates Figure 20(a), representing a single-metric (energy usage) simulation. Then, the user also wants to analyze the CO2 emissions of the infrastructure under the given workload. The user runs the simulation and analysis for CO2 emission, which results in Figure 20(b). With this step, the single-metric-based simulation transitions into a multi-metric-based simulation (energy usage and CO2 emissions).



((a)) Energy usage analysis.

((b)) CO2 emissions analysis.

Figure 20: Single- and multi-metric simulation capabilities.

In this work, we build the Multi-Model on top of the OpenDC simulator [56], but our tool's capabilities are not limited to integration with a single simulator. The Multi-Model allows users to analyze any metric as long as the metric is present as a column name in the analyzed dataset. The technical design of the Multi-Model ensures that it is not constrained to specific metrics but rather functions as an any-metric analyzer.

3.7.3 Facilitate tool universality and integration with any datacenter simulator. (FR3)

We identify and address the critical functional requirement of providing a tool designed towards universality, which can be effortlessly integrated within existing ICT simulation instruments. Aligned with our research philosophy, we aim to maximize our contribution to the scientific community and promote the adoption of our proposed simulation concept and tool.

In Section 3.3, we compare the Multi-Model's two possible high-level integration architectures. We identify a design choice where the tool would be deeply embedded in a simulator codebase, which we term *CWS* (Compute While Simulating), and expand in Section 3.3.2. We identify this design as highly accessible to performance enhancement towards optimization. However, this design choice alienates the re-usability of the

[12] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-3-multi-model-all-plots`

tool as applied to other simulation instruments. In Section 3.3.1, we identify and describe an alternative design choice in which the tool can be used as a top layer and applied to any ICT simulator that respects the bridging requirements. This architecture makes performance enhancement more challenging due to the degree of (non-)integration of the Multi-Model with other simulation instruments. In Section 3.3.3, we compare the two proposed high-level architectures and decide on adopting an *SFCL* architecture in our work.

Alongside the architectural aspects of the Multi-Model simulator, we address the need to provide extensive documentation aligned with the industry, state-of-the-art standards for Kotlin [45] and Python [81] code, the programming languages used for developing the Multi-Model. This step is critical to maximizing the life expectancy of the codebase.

Furthermore, we provide detailed documentation on the Multi-Model simulator's input requirements and capabilities, publicly available on Github[13], as well as a JSON schema validator[14] for the tool's input. Lastly, we provide an integration guide[15], of the Multi-Model, with any datacenter simulator.

3.7.4 User can select the outputting granularity. (FR4)

We identify and address the requirement of allowing users to select the granularity at which the Multi-Model computes, analyzes, and predicts. We refer to this outputting granularity as "window-size," a one-dimensional convolutional layer applied using an aggregation function. We expand the concept of windowed aggregation in Section 3.6.

Users can select the window size as part of the Multi-Model setup. This represents the number of data chunks present per window. To simplify the process for the user, we provide this as a JSON file, in which users adjust the window size only by specifying either window size 1 (i.e., no window applied) or any distinct, positive integer. In Listings 1 and 2, we present a sample from the input interface, highlighting the ease of changing the window size from the user's perspective. The feature of selecting the window size addresses FR4, FR1, and NFR1.

Listing 1: Selection of 25-chunk window.

```
{
    "multimodel": true,
    // setups
    "window_size": 25,
    // other setups
}
```

Listing 2: Selection of 250-chunk window.

```
{
    "multimodel": true,
    // setups
    "window_size": 250,
    // other setups
}
```

3.7.5 Provide in-meeting, near-interactive, same-day simulation results. (NFR1)

We identify and address the performance non-functional requirement of providing in-meeting, near-interactive, same-day simulation results. NFR1 establishes the performance threshold for computing the Multi-Model to be less than the time required for performing the simulation. We develop the Multi-Model tool as efficient, performance software with minimal redundancy.

In Experiment 2[16], (detailed in Section 3.6.2), we analyze the time required to run the simulation, with the datacenter sampling rate of 3 seconds, 6 seconds, 30 seconds, 60 seconds, and 300 seconds. This led to output datasets of 201,160 samples, 100,800 samples (NFR2), 20,160 samples, 10,080 samples, and 2,016 samples. After obtaining the simulation results, we computed Multi-Models with window sizes of 1, 10, 100, and 1,000 chunks. Then, we plotted a time series plot, a cumulative plot, and a cumulative time series plot for each window size; the experiment configuration, setup, inputs, and outputs can be found on GitHub.

[13] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/Input/MultiMetaModel.md

[14] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/Input/MultiMetaModelSchema.md

[15] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/M3SA-integration-tutorial.md

[16] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-2-window-performance-analysis

Due to GitHub file size limitations, the simulation data for the sample rate of 3 seconds is not available on Github but can be computed locally by following the steps in the reproducibility document associated with the simulation.

In Table 2, we present the direct proportionality relation between the outputting granularity, the simulation duration, and the output file size.

s_r	300	100	30	6	3
s_t	9 seconds	28 seconds	58 seconds	218 seconds	592 seconds
d_s	2,016 samples	10,080 samples	20,160 samples	100,800 samples (NFR1)	201,600 samples

Table 2: Comparison of the time and disk space required by OpenDC to simulate, using various exporting granularities. s_r represents the sampling rate of the simulator, s_t represents the simulation time (in seconds) using four models. d_s represents the number of entries in the dataset.

In Table 3, we display the outcomes of Experiment 2, along with the time taken by the simulator to run the simulations. We highlight that, for any window size, the Multi-Model computes, plots, and outputs in less than half of the simulation time, even when running on a regular-user machine (i.e., not a supercomputer), specifically an Apple MacBook Pro, M2, 2023, 16GB RAM, M2 pro chip.

This aligns with the thresholds set by NFR1, as the Multi-Model analyzes, on average, in 35.8%, 19.9%, 10.0%, 9.2%, and 4.7% of the benchmark-established time, for datasets of 201,160 samples, 100,800 samples, 20,160 samples, 10,080 samples, and 2,016 samples. We hence prove that we meet NFR1 and compute the Multi-Model in 19.9% of the requirement-set threshold. Additionally, despite NFR1's requirement to meet the threshold for up to "only" 100,000 values (equivalent to 347 days of simulation at the industry standard export rate of 300 seconds), we extend the simulation to 201,600 samples and still meet the NFR1 criteria.

		$d_s = 201,600$	$d_s = 100,800$	$d_s = 20,160$	$d_s = 10,080$	$d_s = 2,016$
Time Series	w_1	264.7 s	44.9 s	9.8 s	4.4 s	0.8 s
	w_{10}	187.8 s	42.9 s	5.9 s	2.6 s	0.4 s
	w_{100}	220.0 s	46.3 s	5.5 s	2.5 s	0.4 s
	w_{1000}	238.0 s	42.1 s	6.0 s	2.6 s	0.4 s
Cumulative	w_1	152.0 s	47.2 s	4.5 s	1.9 s	0.4 s
	w_{10}	150.4 s	43.6 s	4.7 s	1.8 s	0.4 s
	w_{100}	158.4 s	42.2 s	4.1 s	1.8 s	0.3 s
	w_{1000}	138.1 s	45.8 s	4.6 s	1.9 s	0.3 s
C.T.S.	w_1	254.7 s	41.1 s	7.5 s	3.6 s	0.6 s
	w_{10}	174.4 s	43.3 s	5.6 s	2.6 s	0.4 s
	w_{100}	184.3 s	41.4 s	5.7 s	2.5 s	0.4 s
	w_{1000}	175.4 s	41.2 s	5.9 s	2.7 s	0.4 s
	Simulation Time	592.0 s	218.0 s	58.0 s	28.0 s	9.0 s

Table 3: Time Comparison, for computing, plotting, and saving the Multi-Model. d_s is the dataset size. w_1, w_{10}, w_{100}, w_{1000}, are window sizes of 1, 10, 100, and 1,000. "C.T.S." abbreviates "Cumulative Time Series".

3.7.6 Facilitate reproducible science and experimentation. (NFR2)

We identify and address the non-functional requirement of facilitating reproducible science and experimentation, improving the credibility of this work's presented results. We provide open-source software, publicly available on GitHub[17]. This way, we improve the chances of having our work further adopted or expanded by other researchers towards facilitating reproducible and evolutionary science.

We facilitate reproducible experimentation by publishing our experiments on the GitHub Repository of the project, in the *"experiments"* folder[18]. In this folder, we provide, for each experiment, the input configuration (*"inputs"* folder), the reproducibility file (*"reproducibility.md" file*), which presents the steps of reproducing the experiment, and the results by the experiment (*"outputs"* folder). Due to GitHub limitations, we cannot host files larger than 100 MB. However, these large files can be generated locally by running the experiment and following the steps presented in the reproducibility file.

3.7.7 Leverage the simulation data of up to 4 individual models. (NFR3)

We identify and address the non-functional requirement of comprehensively leveraging simulation data of up to 4 individual models while still meeting the other functional and non-functional requirements. In Figure 19, from Section 3.7.1, we provide nine plots, in which we leverage four simulation models, per plot, and comprehensively visually output their predictions into unified visuals. In Section 3.7.5, we run Experiment 2, in which we simulate using four distinct simulation models, and we show that we meet the NFR1 regarding the performance of the Multi-Model simulator. We hence prove that the Multi-Model simulator meets the NFR3 by efficiently and visually leveraging simulation data of up to 4 individual simulation models.

However, we do not restrict the Multi-Model to any pre-defined maximum number of models, but we do not guarantee the visual comprehensiveness and performance of a Multi-Model that simulates using more than four models.

In Experiment 4[19], we simulate the same infrastructure as in Experiment 3, simulated with 8 distinct power models, under the SURF-SARA workload. The infrastructure is small-scale, with 4 clusters and 8 hosts per cluster, each host containing 64 GB of main memory and an Intel Xeon Sapphire Rapids, 32 cores, operating at 2.6 GHz. In Experiment 4, we run the models *"sqrt"* (equation (22)), *"linear"* (equation (19)), *"square"* (equation (20)), and *"cubic"* (equation (21)), with 2 different CPU configurations. In configuration 1, the CPU consumes 350W, with a maximum power of 500W and an idle power of 200W. In the second configuration, the CPU consumes 330W, with a maximum power of 450W and an idle power of 150W. We plot the results in Figure 21.

[17] `https://github.com/Radu-Nicolae/opendc/tree/local-master`
[18] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments`
[19] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-4-more-than-4-models`

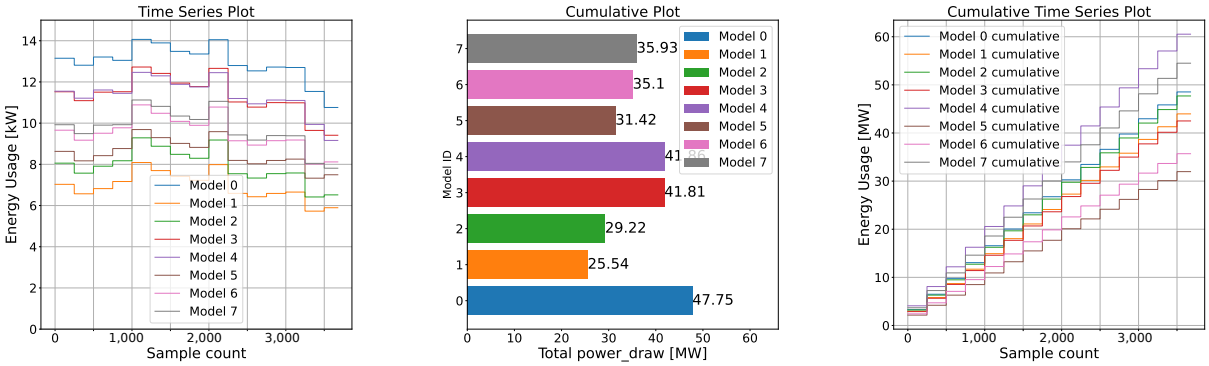


Figure 21: Multi-Model simulation of 8 models, in 3 plotting types, of the energy consumption of the workload SURF-SARA, as part of Experiment 4.

- Model 1 - *"sqrt"*, (equation (22)), CPU configuration 1.
- Model 2 - *"linear"*, (equation (19)), CPU configuration 1.
- Model 3 - *"square"*, (equation (20)), CPU configuration 1.
- Model 4 - *"cubic"*, (equation (21)), CPU configuration 1.
- Model 5 - *"sqrt"*, (equation (22)), CPU configuration 2.
- Model 6 - *"linear"*, (equation (19)), CPU configuration 2.
- Model 7 - *"square"*, (equation (20)), CPU configuration 2.
- Model 8 - *"cubic"*, (equation (21)), CPU configuration 2.

3.8 Discussion

We proposed in this chapter the Multi-Model simulator, a tool that leverages the individual simulation models into a unified tool. This addresses RQ1. The Multi-Model is designed as a top layer towards universality and integration with any datacenter simulator; in this work, we integrate the Multi-Model and apply it to the OpenDC simulator, leveraging its existing feature set and extending it in critical areas. The explorative capabilities of the Multi-Model, combined with support for both long-term and short-term, performant, visually comprehensive, and highly customizable modeling, enable integral analysis of datacenters, using multiple simulation models.

4

The Meta-Model Vision

In this chapter, we address the second research question (RQ2) by presenting a novel ICT simulation concept. We propose the *Meta-Model*, a tool able to predict using other models' predictions and alleviate the characteristic errors of individual models, thus strengthening the credibility of simulation results.

4.1 Overview

We design the Meta-Model simulator, the first tool to facilitate datacenter simulation based on other models' simulation results. Throughout this chapter, we match the state-of-the-art AtLarge Design Process [40]. Our contribution in this chapter is seven-fold:

- We establish the functional and non-functional requirements for the Meta-Model in Section 4.2.
- We analyze conceptual design choices for the Meta-Model integration within the Multi-Model simulator in Section 4.3.
- We envision the meta-simulation concept, its embedding within a system, and its integration upon a generic simulator in Section 4.4.
- We propose, define, and analyze three accuracy metrics used for quantifying the quality of the Meta-Model, in Section 4.5.
- We provide functions used by the Meta-Model for computation, in Section 4.6.
- We address, individually, each established requirement in Section 4.7.
- We make the Meta-Model tool publicly available on GitHub[20].

We summarize the contributions of this chapter in Section 4.8.

4.2 Requirements Analysis

In this section, we determine the functional and non-functional requirements that the Meta-Model should address. This section matches stage 1 of the AtLarge Design Process [40].

4.2.1 Functional Requirements

Main Functional Requirement (MFR): Predict using the other models' predictions.

(**FR1**) **Support prediction based on other models' predictions.**
Singular-model-based simulation is a critical yet non-trivial technique widely employed by any-scale ICT infrastructure simulators [56]. Albeit still valuable, we argue that singular models are often imprecise, inaccurate, biased, and prone to errors when edge cases are encountered [21]. The Meta-Model simulator should predict using the outputs of individual simulation models. The Meta-Model

[20] `https://github.com/Radu-Nicolae/opendc/tree/local-master`

should be able to predict at a user-established granularity (FR4) using a user-selected/default prediction function (FR5, NFR3). Without FR1, the Meta-Model cannot conduct its core functionality of simulating using other models' simulations.

(**FR2**) **Support comprehensive data visualization.** *(similar to Multi-Model FR2)*
The system should enable users to comprehensively visualize data predicted by the Multi-Model and Meta-Model in the same plotting entity (FR4). The Meta-Model simulator should support all the Multi-Model supported types of plots, specifically time series plots, cumulative plots, and cumulative time series plots, from which the users can choose (FR4). The system should allow users to scale data to the appropriate magnitude of the unit prefix, alike the Multi-Model (e.g., 1,073,741,824 units would be converted to 1.073 giga-units). The scaling feature can be enabled and disabled by the user (FR4). Without FR1, the Meta-Model cannot provide comprehensive simulation output into a visual unified tool.

(**FR3**) **Enable single-, multi-, and any-metric simulation.** *(similar to Multi-Model FR3)*
The system should allow users to select which metrics to predict using the Meta-Model. The system should allow users to run single- and multi-metric-based simulation analyses. Besides, the system should provide universality (FR6) and enable users to run any-metric analysis. Without the possibility of choosing which metrics to analyze, the Meta-Model simulator would not be able to efficiently, computationally effectively, and rapidly provide simulation analysis for very large-scale simulation (NFR1). Without allowing any-(custom-)metric analysis, we reduce the tool's universality (FR6) and constrain the simulator's functionality to only predefined metrics.

(**FR4**) **Integrate within the Multi-Model simulator.**
The Meta-Model should be integrated into and within the Multi-Model and provided as an embedded system. This would reduce the complexity and overhead of the software design and offer more room for performance enhancement. Furthermore, this architecture would facilitate simple tool engineering, maintenance, and future engineering. Despite embedding the Meta-Model into the Multi-Model simulator, the user should be able to choose whether to compute only a Multi-Model, both Multi- and Meta-Model, or neither. Without FR4, we limit the performance and the maintainability of the tool.

(**FR5**) **Facilitate tool universality and integration with any datacenter simulator.** *(similar to Multi-Model FR5)*
The system should integrate with any datacenter simulator without modifying the simulator's core architecture. The Meta-Model should be applied as a top layer and incorporated into the users' existing workflows without minimal effort. Integrating the Meta-Model and, hence, also the Multi-Model (FR4), within an existing datacenter simulator should only involve adjusting the simulator's output to match the requirements of M3SA, expanded in the tool's documentation. Without FR5, we hinder the adoption of the Meta-Model tool and concept.

(**FR6**) **Users can select the outputting granularity.** *(similar to Multi-Model FR4)*
The system should allow users to select the window size (i.e., convolution size) at which the Meta-Model analyses and predicts. This feature raises trade-offs between the window size, meta-simulation time, and prediction accuracy. Without FR6, users could not receive comprehensive visual output (FR2) and could not adjust the analysis based on the allocated simulation-analysis time (NFR1).

(**FR7**) **Provide multiple meta-simulation functions.**
The system should contain predefined meta-functions, under which the Meta-Model aggregates predictions of the individual models. Aggregation functions are expected to perform differently regarding time and accuracy. Without FR7, our tool's meta-simulation flexibility is limited, which can result in accuracy loss and performance issues.

4.2.2 Non-Functional Requirements

In addition to the functional requirements, we determine four non-functional requirements for the Meta-Model:

(**NFR1**) **Provide in-meeting, near-interactive, same-day meta-simulation results.**
Datacenter operators must maintain efficient, reliable, and high-speed operation at an unprecedented scale [8, 10]. To ensure the wide adoption of our tool, the Meta-Model must run efficiently, output the simulation results on time, and support very large-scale cloud environments and workloads. The Meta-Model output process (computing, plotting, and saving in various formats) should not take longer than the process of simulation, executed with OpenDC, a tool able to simulate 70x-330x faster than widely-used simulators, such as CloudSimPlus [55]. NFR1 should be met on sets of at least $100,000$ values on a regular-user machine (i.e., not a supercomputer). $100,000$ values represent 347 computing days at the industry-standard sampling rate of 5 minutes [57, 6]. Without NFR1, the Meta-Model cannot be reasonably used in interactive settings or for large-scale infrastructure.

(**NFR2**) **Provide meta-functions with an accuracy higher than the average individual model accuracy.**
Although valuable and widely used, individual model simulation can be biased and prone to errors when encountering edge cases. We identify a similar potential problem, yet at a lower magnitude, also in the Meta-Model. The Meta-Model should provide multiple meta-functions used to compute the meta-simulations. Furthermore, the system should provide three evaluation metrics and compare the accuracy of each meta-simulation function against the ground truth. Without NFR2, the Meta-Model may not achieve the desired accuracy, would not explore alternative (perhaps with higher accuracy) meta-simulation functions, and would restrict users' simulation capabilities.

(**NFR3**) **Facilitate reproducible science and experimentation.**
The results produced by the Meta-Model should be fully reproducible, and the system should minimize the effort of the users who attempt to reproduce results. The experiment setups, scripts, and results should be publicly available. Without NFR3, we weaken the confidence in the results presented in this work.

(**NFR4**) **Leverage the simulation data of up to 16 individual models.**
The simulation output of the individual models should be aggregated and used to compute the Meta-Model. The system should support leveraging data of up to 16 singular models while meeting all other functional and non-functional requirements. However, the system should not limit the number of models to be leveraged by the Meta-Model and allow the user to leverage as many individual models; yet, the system should not be held responsible if more than 16 models are used for meta-simulation and one or more requirements would not be met anymore. Without NFR4, the Meta-Model could not serve the user at the full potential enabled by the Meta-Model vision.

4.3 Meta-Model Design Overview

In this subsection, we present the integration of the Meta-Model, as embedded with the Multi-Model simulator. The tool is designed to be applied to any simulator that follows the integration guidelines. We build this subsection on top of Section 3.3 and Section 3.4.

In Section 3.3, we discuss two potential architectures the Multi-Model can adopt. One architecture involves computing the Multi-Model while simulating, which requires integrating the Multi-Model tool within the core codebase of the simulator; we term this architecture *Compute While Simulating* (*CWS*). In contrast, the other architecture consists of computing the Multi-Model only after the simulation, hereby presenting an architecture in which the core functionality of the simulator is not altered; we term this architecture *Simulate First, Compute Later* (*SFCL*). We analyze both design choices, argue advantages and disadvantages against the established requirements, and build the Multi-Model following the *SFCL (Simulate First, Compute Later)* design. In Section 3.4, we envision a simulation architecture able to leverage multiple simulation models in the prediction component, as well as conduct post-simulation analysis, to compute a Multi-Model and a Meta-Model.

We further expand the *SFCL* architecture to accommodate the Meta-Model's functionality and features. We propose such an architecture in Figure 22. The user sets the simulation in step Ⓐ. In steps Ⓑ, Ⓒ, Ⓓ, the

simulation is run for all the selected models. The Multi-Model process happens in steps E,F,G,H, and I, leading to results in the "ready" state (J).

The Meta-Model process starts in step L, where the Meta-Model is built using the *Multi-Model* (L). At this stage, the Multi-Model has already leveraged and computed the individual models into a unified tool. Further, the *Meta-Model* is initialized (M), and a new model is computed as part of the meta-simulation step (N). This involves aggregating data chunks from all the leveraged models at the same index using a meta-simulation function; we further expand the process of Meta-Simulation in Section 4.4. After step P is completed, the meta-simulation data is appended to the Multi-Model as an individual model, yet with the properties of a Meta-Model in step L.

When the Meta-Model is ready (P), the system plots the results in steps Q and K. The Meta-Model simulation results are outputted alongside the Multi-Model simulation, yet a unique, constant ID differentiates the Meta-Model. Besides the visual plotting, the Meta-Model also saves its simulation data into a parquet file, which can be further used in specific simulation and analysis (step R).

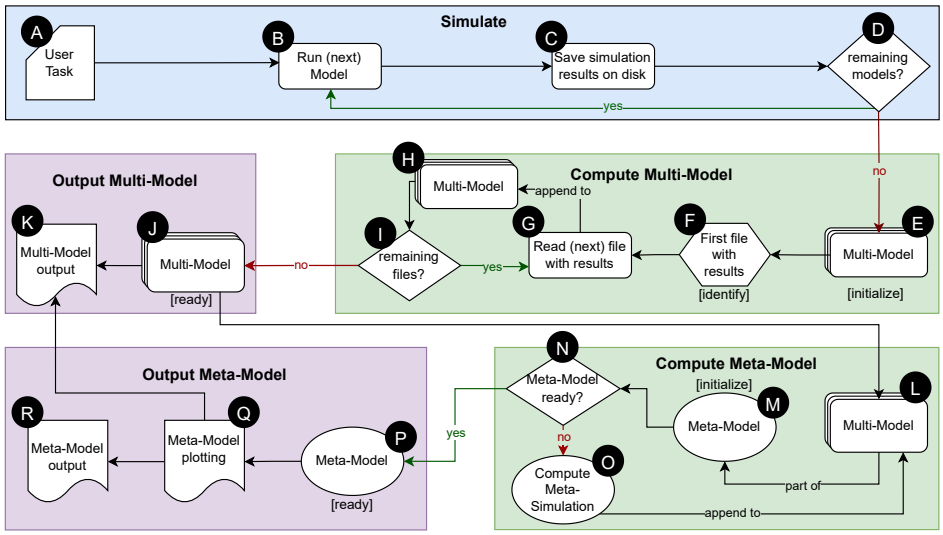


Figure 22: High-Level Architecture of M3SA.

4.4 The Meta-Model Process

Meta-simulation is a novel simulation concept in ICT that proposes leveraging multiple models into a unified tool (Multi-Model) and creating a new simulation model based on the predictions of the individual models. In Section 4.4.1, we expand the concept of meta-simulation and explain why we regard this technique as more reliable than the current state-of-the-art simulation. In Section 4.4.2, we expand a functional architecture of the Meta-Model embed within the Multi-Model. In Section 4.4.3, we present how the new system, containing both the Multi-Model and the Meta-Model, can be integrated within a black-boxed simulator to ensure the universality of the tool. Following the AtLarge Design Process [40], we construct the Meta-Model and M3SA architecture iteratively. This process begins with bootstrapping the creative process (stage 3), after which we focus on the high-level and low-level design (stage 4) [40, 55].

4.4.1 The meta-simulation concept

The meta-simulation concept represents a novelty in the ICT field and proposes the computation of a new simulation model using the simulation results of individual models. We represent this process, on a general case, in the Figure 23.

A number m of model predictions are inputted, each with various sizes. In this example, we consider that the minimum length of each model's prediction is of n data entries. However, models with more or less than n numbers of data entries can exist (e.g., Model 2, Model 3). To ensure a non-biased meta-simulation model, the Meta-Model contains the minimum number of entries per model, in this case, n data entries.

In Figure 23, we regard the model data entries as a table (i.e., a two-dimensional array), starting from the index of 1, both column- and row-wise. The contents of each column are inputted into a meta-simulation function, denoted in this example as F. Function F, receiving the input of c_i, feeds all the elements of column i into the function and outputs a single value. For example, if the function F is the mean of the chunks, alike the Meta-Model-equivalent used in Environmental Sciences [58] (expanded in Section 2.8.3), function F would compute the mean of the elements $K_{i_1}, K_{i_2}, K_{i_3}, ..., K_{i_m}$, where i represents the column index, and m is the total number of individual models.

When aggregating simulation models, various challenges may arise. For example, some models may generate predictions of different sizes due to various reasons. One potential reason is the scheduling system used by the model, which can run a task for shorter or longer periods of time. Another potential reason could be the misalignment of prediction timestamps due to differences in simulator design.

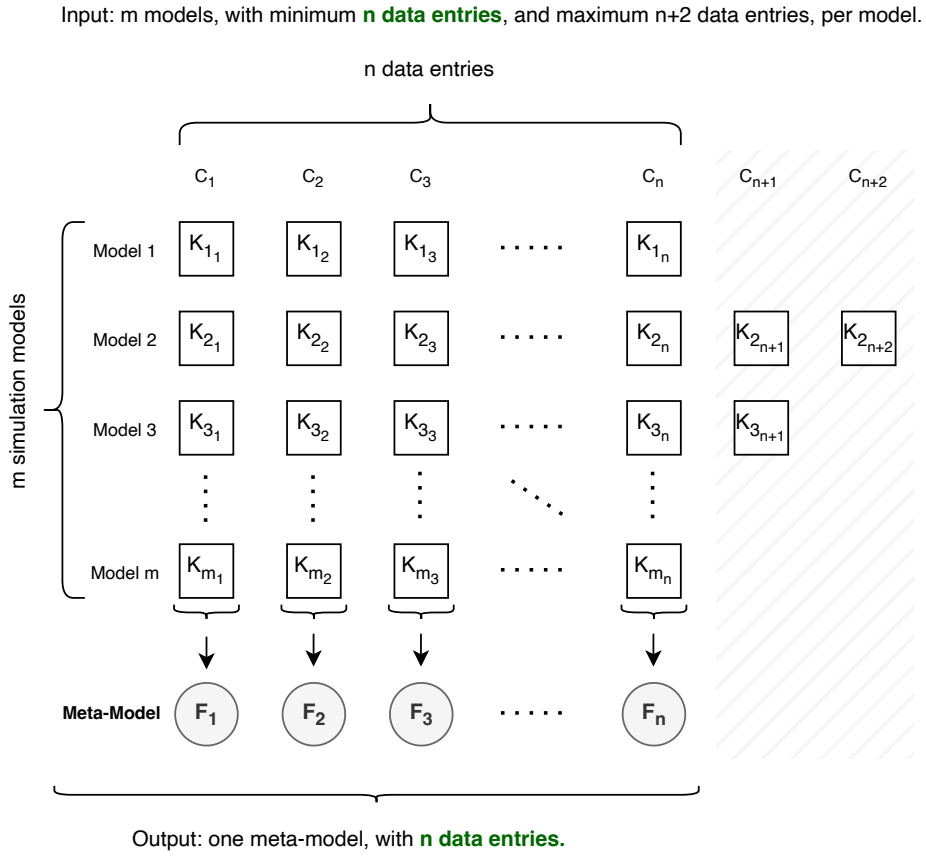


Figure 23: Meta-Model simulation concept.

We identify individual simulation models as valuable yet prone to errors when encountering edge cases; we alleviate such errors with the Meta-Model. The approach of employing a simulation model based on multiple models reduces or eliminates the biases of idiosyncratic models, thanks to the meta-simulation function. The degree of error alleviation depends on two significant aspects. Firstly, a varied length of models' predictions can lead, for the longest models, to have Meta-Model predictions using only some, or one, model's prediction; for example, in Figure 4.4.1, function F, would input only two samples for C_{n+1}, and only one sample for

C_{n+2}. We tackle this challenge by computing the Meta-Model using only the minimum length of models' data entries, in this case, n. Secondly, the meta-function influences the accuracy of the Meta-Model. A highly inclusive function F, such as the mean of the chunks, can be biased by outliers, while a less inclusive function, such as the median, would be less influenced.

4.4.2 Meta-Model embedded within the Multi-Model

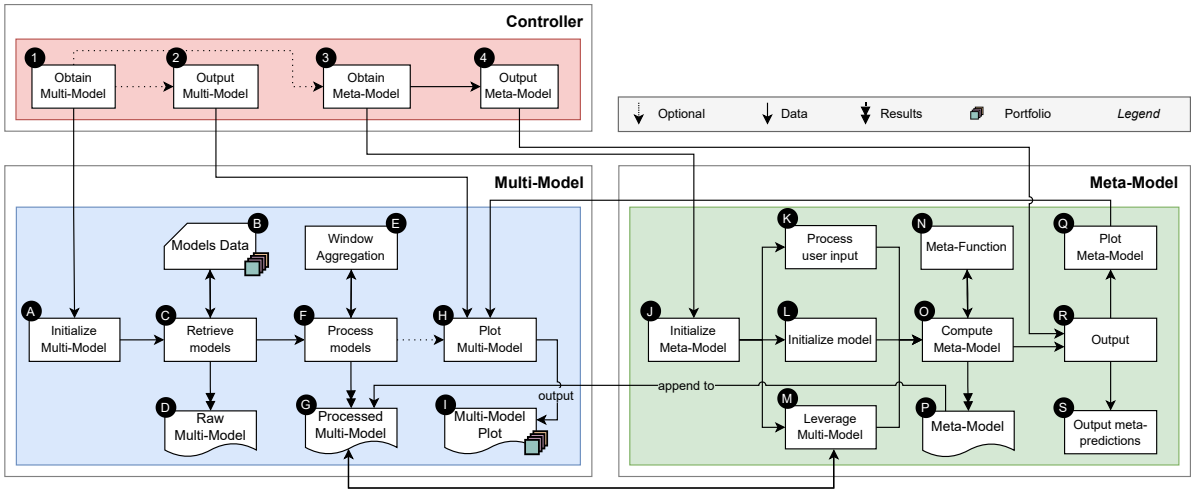


Figure 24: Meta-Model embedded within the Multi-Model.

In Figure 24, we present the overall architecture of the Meta-Model functionality, as embedded with the Multi-Model. The system is orchestrated by the *Controller*, which facilitates the creation and plotting of the Multi-Model (❶, ❷). After obtaining a Multi-Model, an optional Meta-Model can be computed and outputted (❸, ❹).

With a focus on stage ❸, the process of obtaining a Meta-Model starts from Ⓙ, where an instance of the Meta-Model class is initialized. The initialization process involves three sub-processes. First, in *Process user input* (Ⓚ), the Meta-Model configuration is parsed from the JSON interface. Secondly, in *Initialize model* (Ⓛ), an empty instance of the model class is created, later to be populated with the meta-predictions. Thirdly, in *Leverage Multi-Model* (Ⓜ), the Meta-Model integrates the processed state of the Multi-Model. Step Ⓜ is crucial for the performance of the Meta-Model, avoids redundant computation, and enhances performance.

After the initialization process, the predictions of the Meta-Model are generated. Step Ⓞ involves computing a Meta-Model using the predictions of individual models and the *Meta-Function* (Ⓝ) selected by the user. We further expand on the Meta-Model concept and computation in Section 4.4.1. Upon completion, the fully computed *Meta-Model* (element Ⓟ) is appended to the *Processed Multi-Model* component (Ⓖ). This step is crucial for the plotting process.

With a focus on stage ❹, if a Meta-Model is generated, its meta-predictions are automatically outputted. However, the Multi-Model is not automatically outputted for performance reasons and minimized redundancy; the user may choose to build a Multi-Model only as part of the Meta-Model, instead of as a separate entity. The *Output* process (Ⓡ) involves *Outputting meta-predictions* (Ⓢ) in a data file and plotting the *Multi-Model* (Ⓠ). The meta-simulation data is outputted in parquet format due to the scalability, efficient storage, and high compatibility of this storage format [74]. The plotting of the Multi-Model (Ⓠ) leverages the plotting functionality of the Multi-Model. In this step, the Multi-Model contains both the regular simulation models

and the meta-simulation model, with different identifiers and plots their simulations in the same graph. A plot is outputted (Ⓘ) and appended to the plot portfolio.

4.4.3 Meta-Model integrated within a black-boxed simulator

In this section, we present the Meta-Model integration within a black-boxed simulator. The Meta-Model is deeply embedded in the *Multi-Model* and is provided as a system. The system is designed and developed towards universality (FR5), where the provided tool can be applied to any simulator, serving as a top layer.

We present our vision of the system in Section 3.5. For reading convenience, we repeat Figure 15, as Figure 25.

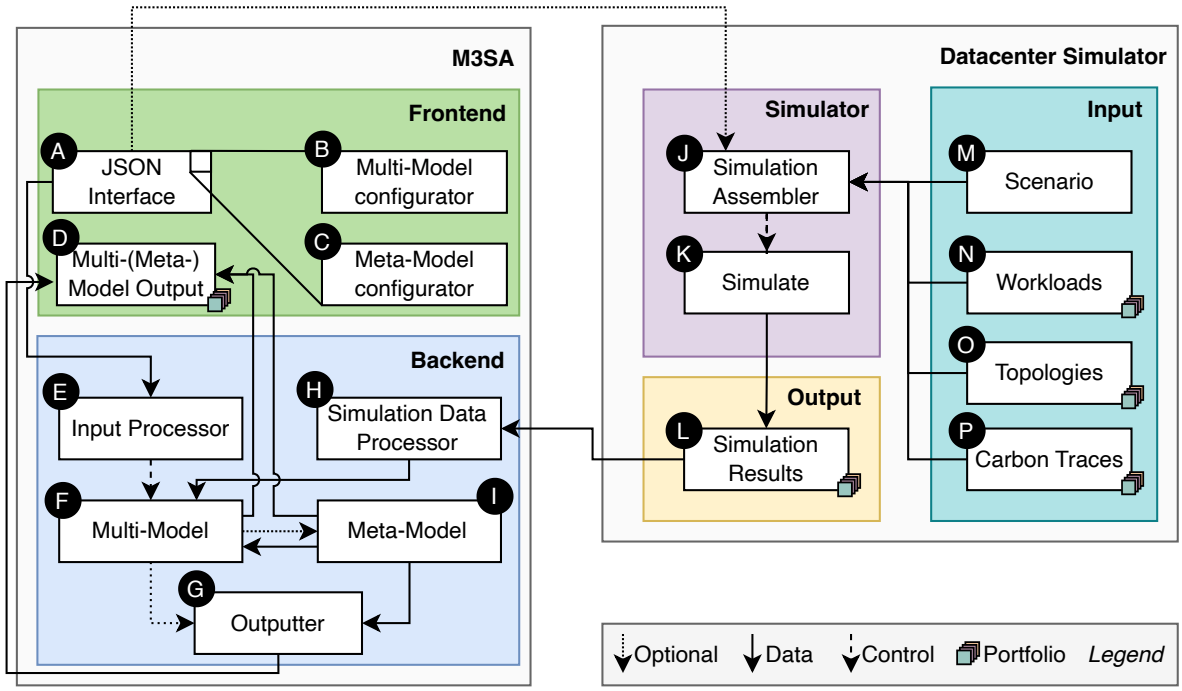


Figure 25: High-level architecture of M3SA integrated within a datacenter simulator.

4.5 Accuracy

In this section, we define accuracy metrics, which we use to quantify simulation results for the rest of this work. We use this conceptual definition and the developed accuracy analyzer tools in Section 5, where we integrate M3SA upon OpenDC and analyze idiosyncratic models' accuracy and Meta-Model's accuracy against real-life data.

To quantify the accuracy of models' predictions, we compare the power draw of a workload determined by the simulator to the real-world power draw of the same workload. We develop a tool in which the user inputs the path to the real-world data and a Multi-Model containing individual models, including a Meta-Model, and receives an accuracy report generated with three different metrics.

The first metric is the Mean Absolute Percentage Error ($MAPE$), a commonly used measure of the accuracy of simulation methods due to its intuitive interpretation of relative error [59, 23]. $MAPE$ is a relative error measure that uses absolute values to keep the positive and the negative errors from canceling one another out [59, 61]. $MAPE$ is calculated using equation (27), where n represents the number of samples, R is the real-world data, S is the simulation data, and i is the sample index.

$$MAPE\ [\%] = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{R_i - S_i}{R_i}\right| \cdot 100 \tag{27}$$

The second metric of accuracy estimation is the Normalized Absolute Differences (NAD), which measures the deviations between the simulated and the real-world data. NAD provides a cumulative measure of absolute differences and is widely used for identifying the total error of the prediction, divided by the sum of the ground truth [66, 59]. NAD is calculated using equation (28), where n represents the number of samples, R is the real-world data, S is the simulation data, and i is the sample index.

$$NAD\ [\%] = \frac{\sum_{i=0}^{n}|R_i - S_i|}{\sum_{i=0}^{n} R_i} \cdot 100 \tag{28}$$

The third metric of accuracy estimation is the Root Mean Square Logarithmic Error ($RMSLE$), adapted from RMSE [16], which penalizes underestimates differently than overestimates, using a variable established by the user. $RMSLE$ is especially effective in simulation contexts where both the scale and the direction of error are critical [83]; for example, building an over-capable-medical-purposed ICT infrastructure, which consumes more energy while being idle, but is able to serve the patients, is less harmful than building an under-capable-medical-purposed ICT infrastructure prone to critical ICT failures and hence hospitals shutdowns. Similar large-scale, critical failures can occur when overestimates are more detrimental than underestimates. We provide a hyperparameter, which can be changed by the user, depending on the over-under estimating ratio. $RMSLE$ is calculated using equation (29), where n represents the number of samples, R is the real-world data, S is the simulation data, i is the sample index, and α is the hyperparameter setting the ratio between underestimates and overestimates; $\alpha \in [0, 1]$, where $\alpha < 0.5$ means that overestimations are penalized more, $\alpha > 0.5$ means that underestimations are penalized more, while $\alpha = 0.5$ means that underestimations and overestimations are equally penalized.

$$RMSLE\ [\%] = \sqrt{\frac{1}{n}\sum_{i=0}^{n}\left(\alpha \cdot \log(R_i + 1) - (1 - \alpha) \cdot \log(S_i + 1)\right)^2} \cdot 100 \tag{29}$$

4.6 Meta-Functions

In this section, we present two meta-functions employed in meta-simulations.

We define a *meta-function* as the function the Meta-Model uses to aggregate data chunks and generate meta-simulation data. The computation and aggregation direction of the meta-function can be in various shapes and forms; for example, in Figure 26, the meta-function is applied vertically. However, the concept also allows the meta-function to be applied horizontally and, even in both vertical and horizontal directions, applied concurrently. The function can employ either mathematical/statistical computation or machine-learning techniques. In this work, we utilize mathematical/statistical functions that are applied vertically.

In Figure 26, extracted from Section 4.4.1, we illustrate the concept of meta-simulation. We expand this simulation concept in Section 4.4.1. Function F, which takes the column at index k as input, represents the meta-function.

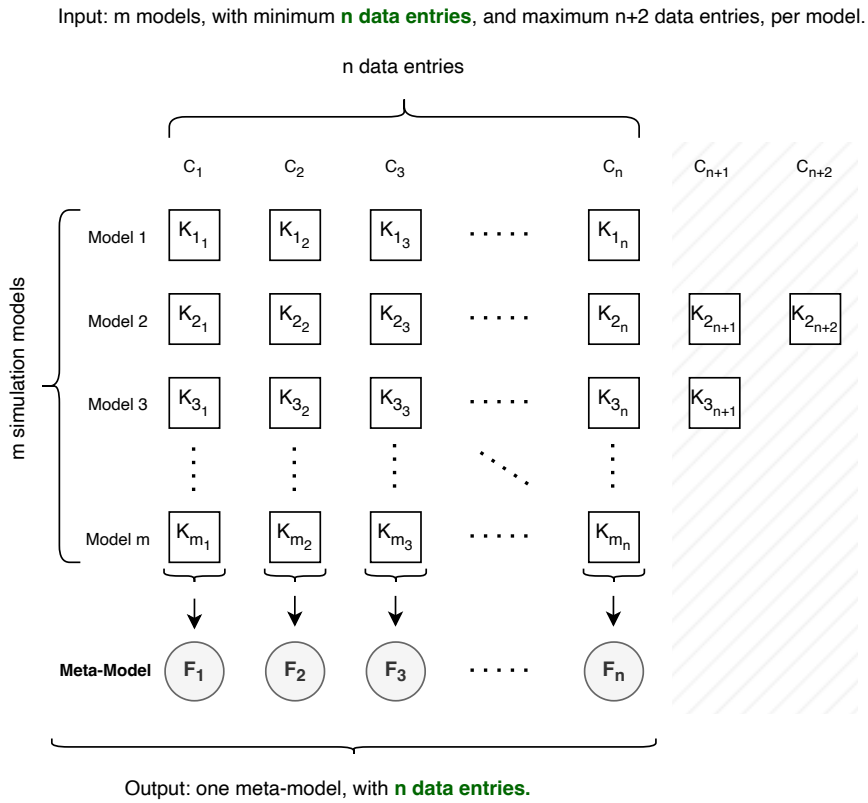


Figure 26: Meta-simulation concept.

4.6.1 Mean as a meta-function

Employed with notable results in various sciences in simulations using aggregated individual models [33, 58], *mean* is a trivial, computationally lightweight, and powerful aggregation function [80]. *Mean* is a statistical/mathematical function applied vertically and consists of summing up the data entries and dividing by the number of entries.

Advantages of Mean
Mean has been widely adopted due to its simplicity and ease of comprehension. Due to its simplicity, *mean* makes the results and the aggregation process clear to various stakeholders. The computation is lightweight and involves only the summation of a small number of values (equivalent to the number of analyzed models) followed by division.

Moreover, the *mean* proves useful in reducing the data noise and partially mitigating the effects of outliers. Additionally, the *mean* yields central values useful for further statistical analysis, such as variance and standard deviation calculations, making it a valuable tool for preliminary and concluding data analysis [80].

Disadvantages of Mean
The Achilles heel of the *mean* is its sensitivity to outliers. While the *mean* alleviates outliers, extreme values (one or more orders of magnitude difference) can skew the *mean* significantly, making it an unreliable metric in certain cases.

Simplicity represents a major advantage of the *mean* function, yet it comes with trade-offs. One of the disadvantages is the assumption of data symmetry around a central value, which may not always hold in the simulation field and can lead to potential misinterpretations. Another disadvantage is the lack of depth of this technique, which can lead to an inaccurate granular decision of the most accurate simulation entry [80].

4.6.2 Median as a meta-function

Similar to the *mean* function, the *median* is a simple, computationally lightweight, and powerful aggregation function. *Median* is a statistical/mathematical function applied vertically. The *median* function involves sorting a dataset and identifying the middle value. For a (sorted) dataset with an odd number of entries, where the indexing starts from 0, and s is the size of the dataset, the median value is located at index $\lfloor s/2 \rfloor$. For a (sorted) dataset with an even number of entries, the median value is the average of the values located at indices $((s/2) - 1)$ and $(s/2)$ [80].

Advantages of Median
The main advantage of the *median* is its robustness to outliers, hence not skewed by extreme values; this property makes the *median* ideal for datasets susceptible to outliers or with non-symmetric distributions, such as sets of models prone to extreme errors. In skewed distributions, the *median* provides a reliable central value, making the *median* particularly useful for analyzing datasets with unevenly distributed data. These represent a major advantage of the *median* over the *mean* [80].

The *median* is one of the simplest aggregation functions to understand and implement, making the results and the process clear to various stakeholders. Additionally, the *median* shows great stability under sample augmentation; expanding the dataset while keeping the standard deviation consistent (i.e., not expanding with extreme values) tends not to affect the *median* as dramatically as it might affect the *mean*. This feature can be critical when aggregating a large number of models.

Disadvantages of Median
Despite its robustness, usefulness, and popularity, the *median* also poses drawbacks. Similar to the *mean*, a main trade-off is between the *median*'s simplicity and inclusivity. The *median* does not consider external factors while computing, ignoring much of the available data [80]. Besides, the *median* can be less sensitive to changes in the data distribution that do not affect the midpoint. This can be useful in various scenarios, yet it can become an issue when underestimating the importance of subtle shifts in models' predictions.

While somewhat computationally effective, for very large datasets, *median* can be more computationally expensive than calculating other simple statistical methods (e.g., *mean*) because the data needs to be sorted. For this work, the computational effectiveness does not pose a drawback. Still, for Meta-Models using thousands or higher magnitudes of single models, the *median* can become computationally complex. Also, for datasets with mostly extremely low and extremely high values, the *median* would reintroduce sensitivity to outliers, leading to a possibly even worse sensitivity than the *mean*.

4.6.3 Accuracy Evaluation

We evaluate the accuracy of the meta-functions presented above using three statistical analysis metrics: Mean Absolute Percentage Error (*MAPE*), Normalized Absolute Differences (*NAD*), and Root Mean Square Logarithmic Error (*RMSLE*). We present and expand on *MAPE*, *NAD*, and *RMSLE* in Section 4.5.

In Experiment 5[21], we run 8 simulations, each with distinct prediction models, and analyze their accuracy against the *MAPE*, *NAD*, and *RMSLE* (lower values are better). We present the analysis results in Table 4 and emphasize the accuracy of each individual model against the accuracy of the Meta-Model, computed with various meta-functions.

We observe that the Meta-Model using the *mean* meta-function achieved a 42.5% improvement in *MAPE* compared to the average individual model, a 44.3% improvement in *NAD* compared to the average individual model, and a 37.3% improvement in the *RMSLE* factor. We also observe that the *mean* meta-function performs better than the *median* meta-function in *MAPE* and *NAD* measurements, while the *RMSLE* is relatively similar for an $\alpha = 0.5$ (i.e., both too high and lower values are penalized equally). However, we observe that the Meta-Model, with any function, predicts at a lower accuracy than the best-individual model (i.e., Model 0).

[21] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-5-accuracy-evaluation`

Entity	MAPE [%]	NAD [%]	RMSLE [%]($\alpha = 0.5$)
Individual Model 0	3.544	3.508	2.048
Individual Model 1	13.829	13.223	6.913
Individual Model 2	16.866	16.281	8.123
Individual Model 3	3.568	3.649	2.416
Individual Model 4	28.574	28.109	12.654
Individual Model 5	11.421	11.595	6.478
Individual Model 6	8.384	8.533	4.885
Individual Model 7	22.24	21.707	10.234
Best Individual	3.544	3.508	4.448
Average Individual	13.428	13.451	6.467
Meta-Model (mean)	7.789	7.46	4.163
Meta-Model (median)	8.591	8.271	4.448
Real-World	0.000	0.000	0.000

Table 4: Accuracy Evaluation of individual models and Meta-Models aggregated with *mean* and *median*.

4.7 Requirements Addressal

In this section, we present how this work addresses each functional and non-functional requirement, set in Section 4.2.

4.7.1 Support prediction based on other models' predictions. (FR1)

Simulations based on individual models are widely employed in ICT infrastructure [56]. Although valuable, we argue that the idiosyncratic models are biased and prone to errors when encountering edge cases. Throughout Section 4, we envision a concept and a tool able to predict using other models' predictions; we term this concept *meta-simulation*, and we term the tool *Meta-Model*. The Meta-Model can predict using the simulation output of other models, as described in Figure 23. The Meta-Model leverages the simulation results of individual models, as we depict in Figure 24.

We design and develop the Meta-Model towards universality and scalability while maintaining high flexibility. In Section 4.4, we present the meta-simulation process, from the initial state, the user input, to the terminal state, the system's output. The process starts with the user's input via a JSON interface. The user can customize the Multi-Model and the Meta-Model using this interface, where they can select the meta-function (FR5, NFR3) and the outputting granularity (FR4) by adjusting the window size. We exemplify the simplicity of this setup in Listing 3 and Listing 4. In Listing 3, the Meta-Model is computed at a granularity of 1, using a *median* meta-function. In Listing 4, the granularity and the meta-function are changed, and the Meta-Model is computed at a granularity of 10 using a *mean* meta-function.

Listing 3: M3SA setup 1.

```
{
    "multimodel": true,
    "metric": "power_draw",
    "window_size": 1,
    // other Multi-Model setups
    "metamodel": true,
    "meta_function": "median",
    // other Meta-Model setups
}
```

Listing 4: M3SA setup 2.

```
{
    "multimodel": true,
    "metric": "power_draw",
    "window_size": 10,
    // other Multi-Model setups
    "metamodel": true,
    "meta_function": "mean",
    // other Meta-Model setups
}
```

4.7.2 Support prediction based on other models' predictions. (FR2)

The Meta-Model system supports comprehensive data visualization by employing and adapting the plotting functionality of the Meta-Model system. We demonstrate the plotting capabilities of the system in Figure 27, where we plot the results of Experiment 6[22], as time series, cumulative, and cumulative time series plots.

In Experiment 6, we simulated a real-life infrastructure from SURF, with 277 hosts, each containing 128 GB of main memory and a CPU with 16 cores, operating at 2.1 GHz. We run the simulations with four distinct models. The Meta-Model has the identifier 101 and uses the *"mean"* meta-function, applied on individual models, processed with a window size of 100. The infrastructure is simulated under the SURF-SARA workload.

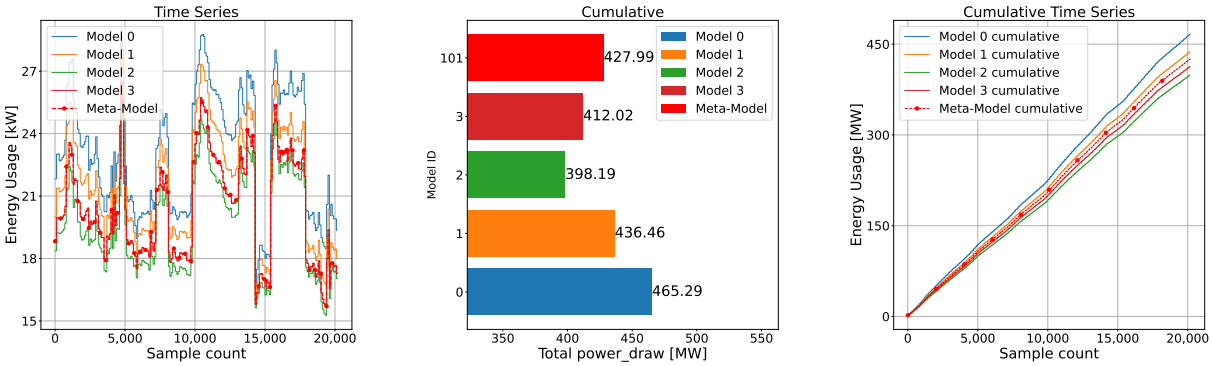


Figure 27: Meta-Model plotting, against individual models, in three types of plots. Meta-Model uses meta-function *mean* and window size 100. Model 1 uses *"sqrt"* model (equation (22)). Model 2 uses *"linear"* model (equation (19)). Model 3 uses *"square"* model (equation (20)). Model 4 uses *"cubic"* model (equation (21)).

In Figure 27, we present the core plotting capabilities of the Meta-Model, yet not exhaustively. The Meta-Model, built on top of and embedded within the Meta-Model, supports extensive customization, such as the plot title, axis values (minima, maxima, dynamically computed ticks), a unit of measurement, data scaling to the selected unit, et cetera. An exhaustive list of the input capabilities can be found in the *Input Documentation*[23]. In the cumulative plot, we show the capability of scaling the values, where the data is measured in megawatts, compared to the time series plot, where the data is measured in kilowatts.

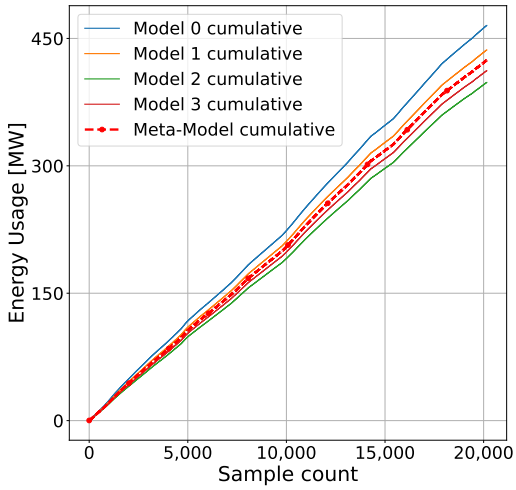
We ensure a good user experience (also) by simplifying the input process. We provide extensive documentation for both the Multi-Model and the Meta-Model, as well as a JSON schema validator.

4.7.3 Enable single-, multi-, and any-metric simulation. (FR3)

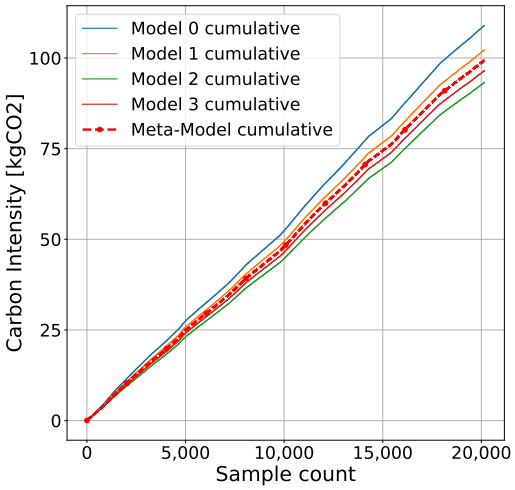
The Meta-Model allows single-metric and multi-metric simulation analysis. Additionally, we do not restrict users to only a predefined set of metrics but allow the analysis of any custom metric as long as the input requirements are followed.

We run Experiment 6[24] and analyze the total energy consumed by the infrastructure (single-metric simulation). Then, we also analyze the total amount of (cumulated) CO2 emissions. Analyzing more than one metric makes the analysis a multi-metric analysis. We plot the results of four individual models and a cumulative Meta-Model in Figure 28.

[22] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-6-metamodel-plotting
[23] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/Input/MultiMetaModel.md
[24] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-6-metamodel-plotting



((a)) Total Energy Consumed, by Sample Count.



((b)) Total CO2 Emissions, by Sample Count.

Figure 28: Single- and multi-metric simulation capabilities.

Applying this simulator as a top layer on a black-boxed simulator, we do not set pre-defined metrics for measurement. We allow custom metric selection, hence allowing any-metric meta-simulation.

4.7.4 Integrate within the Multi-Model simulator (FR4)

True to our scientific philosophy, we research and engineer toward *Massivizing Computer Systems*, focusing on scalability, performance, universality, and state-of-the-art standards. We integrate the Meta-Model within the Multi-Model and provide them as a unitary system; this design choice reduces the complexity of the codebase architecture and enhances the code re-usability, meeting the software design industry standards [62]. In Figure 30, extracted from Section 4.4.2, we present the overall architecture of the system, focusing on the embedding of the Meta-Model with the Multi-Model.

The Meta-Model reuses core data and functionality from the Multi-Model, such as the input parsing, processed and assembled models, and plotting functionality. This design avoids redundant code repetition. Moreover, this design avoids the redundant computation of the code in cases where the user computes both a Multi-Model and a Meta-Model by integrating the Multi-Model properties and data (e.g., processes and assembled individual models) into the Meta-Model. Additionally, a deep interconnection of the two components of the system allows for performance enhancements. We explain, in detail, the system's architecture in Section 4.4.2.

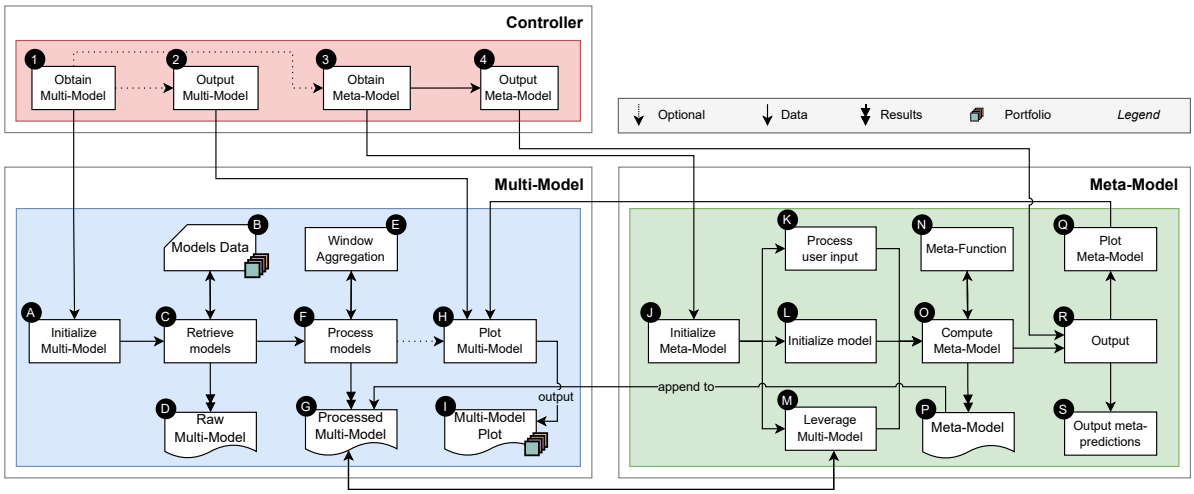


Figure 29: Meta-Model embedded within the Multi-Model.

Despite the deep embedding of the Meta-Model into the Multi-Model, the tool allows users to choose whether to compute only a Meta-Model, both a Meta-Model and a Meta-Model, or none (i.e., disable the tool). If a Meta-Model is selected for computation, part of the Multi-Model's functionality needs to be used, such as the individual model aggregation and the chunk aggregation (i.e., windowing). In Figure 30, we show the decision chart of which system components are computed.

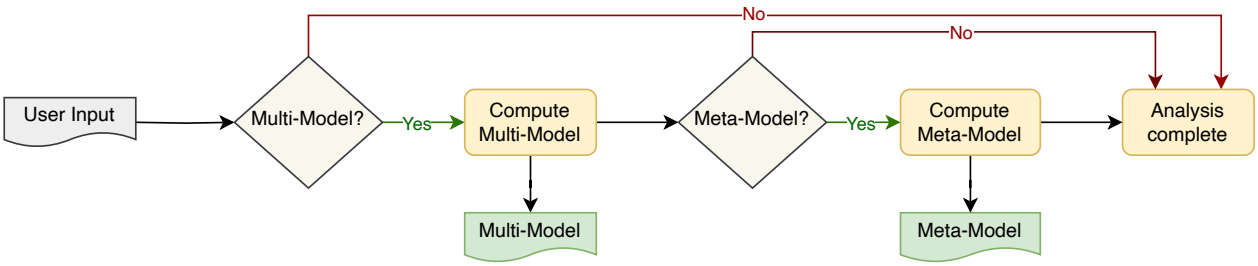


Figure 30: Decision flowchart for computing Multi-Model, Meta-Model, both, or none, based on user input.

In Listings 5, 6, 7, and 8, we present the four possible inputs from the JSON interface. In Listing 7, we present an edge case in which the user selects only the computation of the Meta-Model and disables the computation of the Multi-Model; since a Meta-Model cannot exist without a Multi-Model, the system will give a warning, and no computation will be performed.

Listing 5: Both enabled.

```
{
    "multimodel": true,
    "metamodel": true,
    // other setups
}
```

Listing 6: Only Multi-Model enabled.

```
{
    "multimodel": true,
    "metamodel": false,
    // other setups
}
```

Listing 7: Only Meta-Model enabled (N/A).

```
{
    "multimodel": false,
    "metamodel": true,
    // other setups
}
```

Listing 8: Both disabled.

```
{
    "multimodel": false,
    "metamodel": false,
    // other setups
}
```

4.7.5 Facilitate tool universality and integration with any datacenter simulator. (FR5)

We design and build the meta-simulation system aligned with our scientific philosophy of providing open-source code that is simple to reuse and expand. We identify tool universality as critical for our tool to become a state-of-the-art simulation concept widely employed in ICT simulations.

The architecture of the Meta-Model is closely followed and embedded into the architecture of the Multi-Model. In Section 3.3, we analyze the design choices of the Multi-Model system, compare the found architectures, and choose the architecture that aligns best with the universality requirement. We further adopt the same architecture for the Meta-Model and embed this component within the Multi-Model component, thus providing M3SA as a unitary system (FR4). In Section 4.4, we present the integration of the M3SA with a black-boxed, generic simulator. We emphasize the "top layer" behavior of the M3SA, which can be applied to any simulation tool that respects the input requirements from the documentation[25].

Moreover, although designed for ICT simulations and integrated into OpenDC (Section 5), M3SA can be integrated within any simulator from any field, which provides evolutionary predictions and follows the integration requirements.

4.7.6 Users can select the outputting granularity. (FR6)

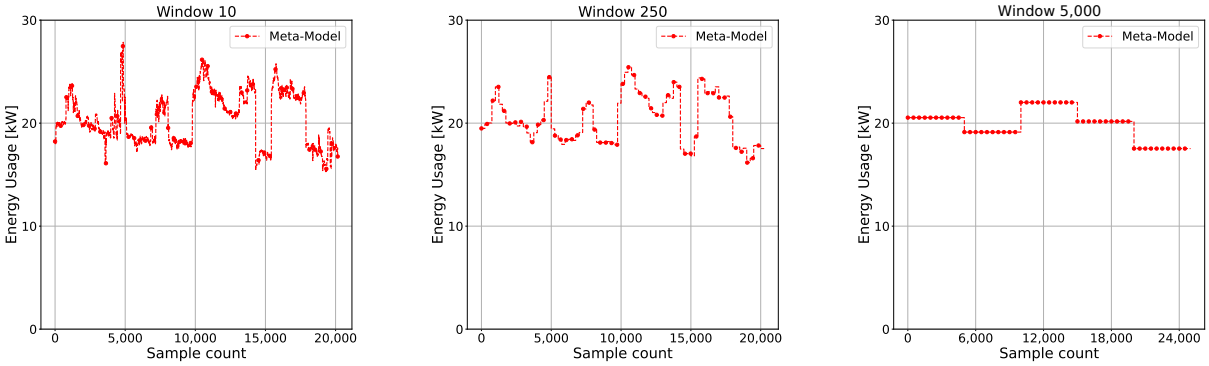
The system allows users to select the outputting granularity at which the Meta-Model analyzes and predicts. We facilitate this functionality through the input interface, where the user can adjust the window size. In Listing 9, we present the relevant component of the input interface, in which the user can change the window size by inputting any integer value x, where $x \geq 1$.

Listing 9: Window adjustment.

```
{
  "multimodel": true,
  "metamodel": true,
  "window_size": x,
  // other setups
}
```

The size of the window influences various properties of the Meta-Model, such as visual comprehension (e.g., noise, visual accuracy), meta-simulation time, and prediction accuracy. We identify and analyze the trade-off between the size of the window and the meta-simulation time in NFR1 and the trade-off between the size of the window and the accuracy in NFR2. In Figure 31, we present the Meta-Model generated from the predictions of Experiment 6, with window sizes of 10, 250, and 5,000.

[25] https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/M3SA-integration-tutorial.md



((a)) ((b)) ((c))

Figure 31: Three Meta-Models computed with window sizes of 10, 250, and 5000.

4.7.7 Provide multiple meta-simulation functions. (FR7)

We identify the requirement of providing flexibility and allowing users to meta-simulate using the user-chosen meta-function. In Section 4.6, we identify two meta-functions: *mean*, *median*.

We further detail each meta-function, presenting advantages and disadvantages when used for ICT simulations. Furthermore, we evaluate the accuracy of the functions against real-world data from SURF, simulating real-world infrastructure under the SURF-SARA workload. We observe that the accuracy of each meta-function is approximately 40% higher than the average model prediction for metrics such as *MAPE*, *NAD*, and *RMSLE*. We conduct accuracy analyses on the proposed meta-functions in Section 4.6.3 and Section 4.7.9.

4.7.8 Provide in-meeting, near-interactive, same-day meta-simulation results. (NFR1)

We identify and address the performance non-functional requirement of providing in-meeting, near-interactive, same-day, meta-simulation results.

We set the threshold for the Meta-Model to process, meta-simulate, and output in less than the simulation time. NFR1 mandates meeting this requirement on a regular-user machine (i.e., not a supercomputer) for simulations of up to 100,000 samples, and using OpenDC [56] simulator; 100,000 samples is the equivalent of 347 computing days, at the industry-standards sampling rate of 5 minutes [57, 6]. We identify this as a critical requirement of the Meta-Model, which should be computationally efficient for both small and massive-scale meta-simulations.

In Experiment 7[26], we evaluate the performance of the Meta-Model towards meeting NFR1. We run the experiment on an Apple MacBook Pro, M2, 2023, 16GB RAM, M2 pro chip, without any user tasks running in the background. We simulate the energy consumption of a real-life cluster from SURF, with one host containing 128 GB of main memory and 277 CPUs, with 16 cores operating at 2.1 GHz. We simulate using 16 different models, using various energy configurations and four power models (sqrt, linear, square, and cubic). We use various export rates and obtain datasets of 2,016, 10,080, 20,160, 100,800, and 201,600 samples. Then, we analyze the performance of the Meta-Model, with 4 window sizes and all the available meta-functions. We test the outputting of the Meta-Model with the time series, cumulative, and cumulative time series graphs, as well as outputting the Meta-Model predictions in a parquet data file.

This leads to a total of 180 Meta-Models analyzed. More precisely, for each of the 5 datasets, for each of the 4 window sizes, for each of the 3 plot types, and for each of the 2 meta-simulation functions, we analyze the time required to process, generate, and plot the corresponding Meta-Model. The experiment input and output are publicly available on GitHub, except for the very large simulation files (≥100 MB), which cannot

[26] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-7-metamodel-performance-analysis`

be hosted due to platform limitations. The entire experiment can be reproduced by following the instructions in the reproducibility file.

We plot the experiment results in Table 5 and prove that the Meta-Model meets NFR1.

The meta-simulation process involves assembling the models and applying the windowed aggregation function. Then, the Meta-Model creates a novel model by applying the assigned meta-simulation function to the processed data. Lastly, in the output process, the Meta-Model packs and saves the meta-simulation data into a parquet file and then plots it.

We identify a large gap between the meta-simulation and the actual simulation time. In all the run meta-simulations, the Meta-Model meta-predicted in less than 25% of the time limit set by the NFR1. For example, for window sizes of 2,016, the Meta-Model is computed, in the worst-case scenario, in less than 10% of the computational time set by NFR1. Although this gap becomes smaller as the dataset size increases, the Meta-Model still meets the threshold set by NFR1 by analyzing, in the worst-case scenario, in less than 25% of the allocated time. We observe a comparable, almost identical performance between the *mean* and the *median*.

We observe comparable simulation times between the window sizes. This is because, for smaller window sizes, the process of chunk aggregation consumes fewer resources (or is even skipped if the window size is 1), but this involves more computation for the Meta-Model when the meta-function is applied and when the results are saved and plotted. For a larger window size, the process of chunk aggregation consumes more resources, but this leads to fewer data entries on which the Meta-Model meta-simulates, as well as fewer data entries to be saved and plotted. Therefore, we conclude that the window size does not (significantly) influence the performance of the Meta-Model.

			$d_s = 201,600$	$d_s = 100,800$	$d_s = 20,160$	$d_s = 10,080$	$d_s = 2,016$
***Mean* Meta-Function**	**Time Series**	w_1	186.2 s	85.4 s	15.7 s	4.5 s	0.9 s
		w_{10}	181.8 s	83.4 s	11.4 s	4.3 s	0.8 s
		w_{100}	196.4 s	82.5 s	12.0 s	4.2 s	0.8 s
		w_{1000}	202.9 s	85.3 s	12.4 s	4.1 s	0.7 s
	Cumulative	w_1	199.9 s	88.4 s	12.2 s	4.0 s	0.8 s
		w_{10}	209.9 s	87.4 s	12.5 s	4.1 s	0.8 s
		w_{100}	202.4 s	86.8 s	12.4 s	4.2 s	0.7 s
		w_{1000}	200.1 s	84.4 s	13.0 s	4.1 s	0.8 s
	C.T.S.	w_1	199.0 s	84.4 s	9.9 s	4.1 s	0.7 s
		w_{10}	191.6 s	86.0 s	9.5 s	3.9 s	0.7 s
		w_{100}	195.1 s	84.9 s	9.7 s	3.9 s	0.7 s
		w_{1000}	192.7 s	85.0 s	8.8 s	3.9 s	0.8 s
	Simulation Time		944.0 s	592.2 s	58.4 s	28.0 s	9.1 s

			$d_s = 201,600$	$d_s = 100,800$	$d_s = 20,160$	$d_s = 10,080$	$d_s = 2,016$
***Median* Meta-Function**	**Time Series**	w_1	202.0 s	86.8 s	10.1 s	4.3 s	0.8 s
		w_{10}	206.2 s	92.5 s	10.9 s	4.3 s	0.8 s
		w_{100}	194.0 s	83.5 s	11.1 s	4.1 s	0.7 s
		w_{1000}	194.5 s	82.9 s	10.0 s	4.1 s	0.7 s
	Cumulative	w_1	193.5 s	83.6 s	9.9 s	4.0 s	0.7 s
		w_{10}	179.1 s	82.1 s	9.4 s	3.9 s	0.7 s
		w_{100}	179.0 s	80.9 s	9.9 s	3.9 s	0.7 s
		w_{1000}	185.6 s	81.6 s	9.4 s	4.0 s	0.7 s
	C.T.S.	w_1	175.4 s	80.4 s	9.7 s	4.0 s	0.7 s
		w_{10}	185.3 s	81.0 s	9.5 s	3.8 s	0.7 s
		w_{100}	181.7 s	82.3 s	9.5 s	3.9 s	0.7 s
		w_{1000}	178.1 s	82.5 s	9.5 s	3.8 s	0.7 s
	Simulation Time		944.0 s	592.2 s	58.4 s	28.0 s	9.1 s

Table 5: Time comparisons for assembling, computing, and outputting (plotting and saving) the *Meta-Model*. d_s represents the dataset size, w_1, w_{10}, w_{100}, w_{1000}, represent window sizes of 1, 10, 100, and 1000, C.T.S. abbreviates "Cumulative Time Series".

4.7.9 Provide three meta-functions, with an accuracy higher than an average individual model accuracy. (NFR2)

We identify the main requirement of the Meta-Model to have a higher accuracy than the average individual simulation model. In Section 4.5, we present three commonly used accuracy metrics: Mean Absolute Percentage Error (*MAPE*), Normalized Absolute Difference (*NAD*), and Root Mean Square Logarithmic Error (*RMSLE*). In Section 4.6, we expand the *mean* and *median* meta-functions; we then compute Meta-Models using *mean* and *median* meta-functions, aggregating 8 individual models, and evaluate the *Meta-Models'* accuracy against real-world data.

To address the accuracy component of NFR2, we design Experiment 8[27], in which we propose 16 individual power models (NFR4), with various configurations of the modeling functions *"sqrt"* (equation (22)), *"linear"* (equation (19)), *"square"* (equation (20)), and *"cubic"* (equation (21)). We evaluate the accuracy of each model using real-world data from SURF. We simulate an infrastructure from SURF, with 277 hosts, 128 GB of RAM per host, and a CPU with 16 cores operating at 2.1 GHz, under the SURF-SARA workload. Then, we build two Meta-Models, using the *mean*, respectively the *median* meta-functions, and analyze the accuracy of the Meta-Models against the same real-world data. We further compare the accuracy of the Meta-Models against the average individual model (NFR2). We additionally compare the accuracy of the Meta-Models against the most accurate individual model and against the first quartile. We present the results in Table 6.

We observe that both Meta-Models meet the non-functional requirement of having more accurate simulations than the average individual model for each of the three metrics. Therefore, NFR2 is met for 16 models (NFR4). Moreover, the Meta-Model using *median* as the meta-function performs better than the first quartile, as of *MAPE* and *RMSLE* metrics and is almost identical for *NAD* metric. We observe comparable accuracy between the best Meta-Model and the best individual model; however, the best individual model predicts at a higher accuracy. The meta-predictions using the *median* meta-function have better accuracy than the meta-prediction using the *mean* meta-function.

Analyzing the results from Table 6 (Section 4.7.9) and Table 4 (Section 4.6.3), we emphasize the direct correlation between the overall accuracy of the individual models and the accuracy of the corresponding Meta-Model. The accuracy of the individual models can have a higher impact than the meta-function on the accuracy of the Meta-Model. For example, in Experiment 5 from Section 4.6.3, the Meta-Model using the *mean* predicts a higher accuracy than the Meta-Model using the *median*. In Experiment 8, the *median* has better accuracy than the *mean*.

[27] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-8-metamodel-accuracy-analysis`

Entity	MAPE [%]	NAD [%]	RMSLE [%] ($\alpha = 0.5$)
Individual Model 0	3.701	3.642	2.155
Individual Model 1	13.143	12.969	7.243
Individual Model 2	5.396	5.240	3.099
Individual Model 3	10.937	10.828	5.473
Individual Model 4	11.833	11.780	6.638
Individual Model 5	14.184	14.178	6.734
Individual Model 6	17.198	17.198	9.734
Individual Model 7	24.316	24.258	14.173
Individual Model 8	19.186	19.077	10.904
Individual Model 9	3.792	3.713	2.240
Individual Model 10	18.192	18.161	8.434
Individual Model 11	27.227	27.191	16.116
Individual Model 12	5.616	5.488	3.184
Individual Model 13	3.756	3.817	2.495
Individual Model 14	7.205	7.116	3.862
Individual Model 15	20.245	20.269	11.577
Average Model (NFR2)	12.679	12.539	7.128
1st quartile Model	4.418	4.172	3.141
Best Model	3.701	3.642	2.155
Meta-Model Median	4.170	4.194	2.609
Meta-Model Mean	5.351	5.372	3.244

Table 6: Accuracy analysis of individual- and meta- models.

4.7.10 Facilitate reproducible science and experimentation. (NFR3)

We identify and address the non-functional requirement of facilitating reproducible science and experimentation, improving the credibility of this work's presented results. Similar to NFR2 of the Multi-Model, we make the built tool, experimentation setup, and results publicly available on GitHub[28]. This way, we improve the chances of having our work further adopted and/or expanded by other researchers toward facilitating reproducible and evolutionary science.

We facilitate reproducible experimentation by publishing our experiments on the GitHub Repository of the project, in the *"experiments"* folder[29]. In this folder, we provide, for each experiment, the input configuration (*"inputs"* folder), the reproducibility file (*"reproducibility.md" file*), which presents the steps of reproducing the experiment, and the results by the experiment (*"outputs"* folder). Due to GitHub limitations, we cannot host files larger than 100 MB. However, these large files can be generated locally by running the experiment and following the steps presented in the reproducibility file.

4.7.11 Leverage the simulation data of up to 16 individual models. (NFR4)

We identify and address the non-function requirement of leveraging simulation data of up to 16 individual models in the meta-simulation process while meeting all the system's functional and non-function requirements, especially the performance requirements set by NFR1 and the accuracy requirements set by NFR2.

In NFR1, we analyze the performance of 180 Meta-Models with various configurations, all of which use 16 individual models in their meta-simulation process. In Table 3, we present the experiment results, proving that NFR1 is met for 16 models under all the analyzed Meta-Model configurations.

In NFR2, we analyze the accuracy of 16 individual models and the accuracy of two corresponding Meta-Models. Both Meta-Models meet the accuracy requirement of NFR2, proving that NFR2 is met for 16 models, for both the *mean* and *median* meta-functions. Moreover, in Table 4, we present the accuracy of

[28] `https://github.com/Radu-Nicolae/opendc/tree/local-master`
[29] `https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments`

two Meta-Models computed on the predictions of 8 individual models; both Meta-Models have a higher prediction-accuracy than the average simulation model, hence meeting NFR2 also for less than 16 individual models.

Therefore, all the requirements, including the performance and the accuracy non-functional requirements, are met for up to 16 models, hereby meeting NFR4. However, we do not restrict the Meta-Model to any predefined maximum number of individual models, but we do not guarantee the functional and non-functional requirements if the threshold of 16 models is overtaken.

4.8 Discussion

In this chapter, we proposed the Meta-Model simulator, a meta-simulation tool that predicts using the predictions of other models. This addresses RQ2. The Meta-Model is a tool developed as a top-layer, embedded into the Multi-Model, and provided together as a unitary system with high functional customizability. We design the tool towards performance, accuracy, customizability, universality, and integration within any datacenter simulator; in Section 5, we integrate M3SA into the *OpenDC* simulator, leveraging and extending the current feature set in critical simulation areas.

5

Integration and Evaluation of M3SA

In this chapter, we address the third research question (RQ3) by integrating M3SA into a peer-reviewed, top-tier datacenter simulator and evaluating the overall instrument's performance, accuracy, usability, and universality.

5.1 Overview

We conduct a comprehensive evaluation of M3SA, developing a working software prototype and using four different evaluation methodologies. Our contribution in this chapter is six-fold:

- We present a working software prototype of M3SA, implementing key features of the design and integrate such a software tool on top of OpenDC platform. We expand the process in Section 5.2.
- We evaluate the performance of the integrated system (the engineered prototype of M3SA applied on OpenDC) in Section 5.3.
- We validate the accuracy of the integrated system in Section 5.4.
- We evaluate the usability of the system by presenting use-case scenarios of two different stakeholders in Section 5.5.
- We evaluate the universality of M3SA, in Section 5.6.
- We make M3SA, both independent and integrated with OpenDC, available on GitHub.

We summarize our contribution and discuss potential threats to validity in Section 5.7.

5.2 Engineering and Integration of a Software Prototype

In this section, we describe the software engineering processes used to develop the prototype (Section 5.2.1). Then, we describe the extensions and the improvements to OpenDC's simulator regarding input/output and simulation tracking (Section 5.2.2). We then describe the linking layers between the simulator and M3SA, enabling the holistic simulation analysis to run with a single click (Section 5.2.3). Lastly, we present the structure of the reproducibility capsule (Section 5.2.4).

We present a full overview of the technologies adopted by M3SA in Figure 32.

5.2.1 Software Engineering Process of M3SA

We employ the industry-best software development technologies and practices in the engineering process of M3SA and OpenDC. The main codebase of M3SA is written in Python, the second most used programming language, due to its extensive support for various libraries and frameworks [70]. The main codebase of OpenDC is written in Kotlin, a modern and fast-growing programming language, fully interoperable with Java, and already widely adopted by large companies [41, 55].

We adhere to high development standards by employing continuous integration (CI) [27] techniques running for each change, automated test suites, and static code analysis tools (e.g., linting) to spot common

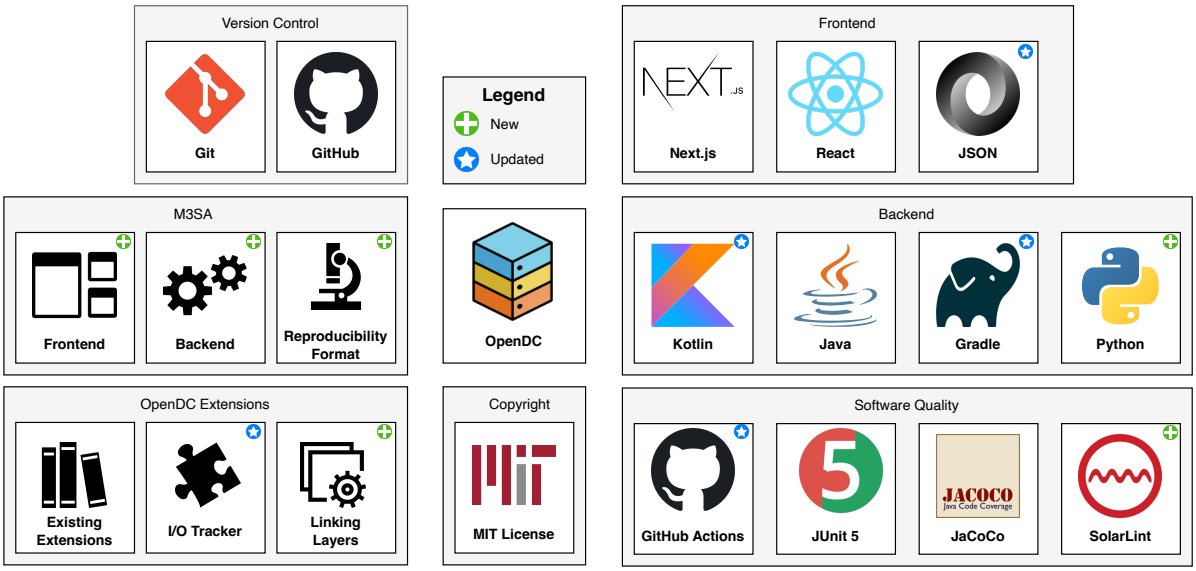


Figure 32: Technologies adopted by M3SA, as integrated in OpenDC.

mistakes and mandate the adherence to best engineering practices. We promote software quality by using GitHub actions, linking standardized GitHub issues with branches, and following patterns and naming conventions. Similarly, pull requests and commit messages follow industry-standard formats employed in large-tech companies, such as Google [29, 30]. Although this approach increases the overall burden of engineering and maintenance, it also ensures high-quality tools, cross-component compatibility, and the long life of the system as OpenDC evolves [62, 55].

5.2.2 Simulation tracking

Reducing the user's overhead is an important feature of the overall simulation system, aiding the user in focusing extensively on the simulation files. Storing local information about the simulations is especially helpful in experiment reproducibility or when reusing large-scale simulation data; this feature prevents redundant re-running of experiments, which can sometimes be computationally expensive in terms of time and resources.

We identified a lack of information on the experiment configuration in the output files, requiring users to document the input details manually. To address this, we enhanced the internal codebase of OpenDC to automatically track simulations by ID, storing the configuration details in a tracker file associated with the experiment. The tracker file can store information about one or more simulations. This feature streamlines the experimentation process, reproducibility, and simulation data re-usability. Moreover, the existence of a tracker file that is automatically generated makes OpenDC more open to the adoption of further extensions.

5.2.3 Applying M3SA on top of OpenDC

We have designed M3SA towards universality, as a tool simple to integrate with any cloud-infrastructure simulator (Multi-Model FR3, Meta-Model FR5). Despite the "top-layer" design of our tool, minimal integration steps are still needed, as presented in the integration documentation[30].

Adapting the Backend

Step 1 from the integration document suggests adapting the simulator output format, structure, and names

[30] `https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/M3SA-integration-tutorial.md`

for the existing folders and files. This step is critical for M3SA to identify the prediction data. To match this structure, we adapt the backend of OpenDC by creating new functionality, which also entails adapting cascading breakdowns of the API for various elements (e.g., methods, classes, parameters).

Adapting the simulation file format
Step 2 from the integration document presents the file format regarding rows and columns and the storage file type. We use *parquet*, a scalable, efficient, and high-compatible storage format widely used in the industry [74]. This format is required for M3SA to parse the data needed for further computation. OpenDC already matches the format required by M3SA. Hence, no integration efforts were needed for this step.

Adapting for one-click simulation and analysis
Step 3 of the integration document presents the configuration for M3SA to be run, as decoupled from the simulator. In the optional step 4, we present a more in-depth integration alternative, in which M3SA is more embedded in the simulator; step 4 presents the simulator and M3SA as a holistic system, in which the simulation and analysis process occur unitary, with the click of one button.

To implement such a system, we adapt the main simulation commander of OpenDC. The command-line interface OpenDC supports flags for specifying paths or setting up the degree of parallelism. We add a new flag, which, only if enabled, activates an adaptation layer between OpenDC and M3SA. Furthermore, we build such an adaptation layer that links the OpenDC codebase, written in Kotlin, with M3SA codebase, written in Python. The linking process involves setting up the script languages and the path to the main function of the tool, as well as providing paths, as parameters, to the output folder and M3SA setup file. By accepting the paths as parameters, M3SA can be used both coupled to a simulator or decoupled.

Adapting the Frontend parser
To improve the usability of the OpenDC, we adapt the infrastructure from computing using only one simulation model to using multiple simulation models concurrently. For this, we adjust the front-end parser to accept multiple simulation models, topologies, workloads, carbon traces, and allocation policies. This enables the simulator to be suitable for this research, oriented towards simulation leveraging multiple models, but also suitable for future researchers, multi-workload analysis, multi-scheduler simulation, et cetera.

5.2.4 Reproducibility capsule

We include in our prototype a *reproducibility capsule*, which contains all the experiments presented in this paper, containing the input traces, portfolios of scenarios, and the scripts and guidelines for generating the raw experiment data, as well as the analysis for this paper. In the limit of 100 MB per file, we also provide the output files on which statistics were made. We make this capsule publicly available on GitHub, meeting the reproducible science requirements of the Multi-Model (NFR2) and the Meta-Model (NFR3). By automating this process, we ensure that the archive is correct and that we can reproduce the archive for arbitrary versions of OpenDC.

Our main findings and contributions in this section are:

MF1	M3SA enables various forms of multiple-model-based analysis of cloud infrastructure.
MF2	Multi-Model analysis provides unified ICT simulation predictions of multiple models in the same plot.
MF3	Meta-Model provides a new model, able to leverage the predictions of other models.
MF4	The researched and engineered system can be integrated with any (ICT) simulator.

5.3 Performance Validation

In this section, we analyze the performance of the holistic system involving M3SA integration with OpenDC [56].

Modern cloud infrastructure operates at an unprecedented scale [55, 28]. Cloud operators must maintain efficient, reliable, and high-speed operation of increasingly large and complex infrastructure in a time-efficient

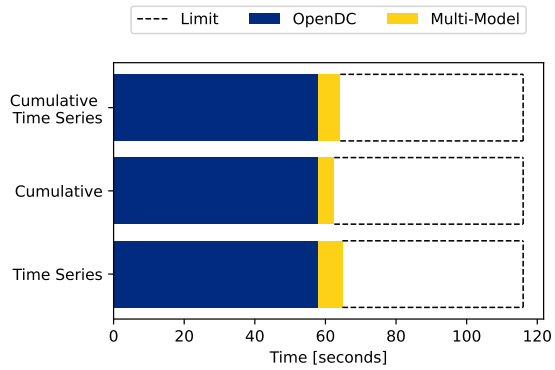
manner [8, 10]. For a ICT simulator to be useful, it must support the immense and growing scale of the modern infrastructure. We identify performance as a critical requirement for M3SA (Multi-Model NFR1, Meta-Model NFR1) and for the holistic system (M3SA-OpenDC integration).

OpenDC simulator is 70x-300x faster than CloudSim Plus [67], an ICT simulator widely used in the field, according to the peer-reviewed experimental measurements made by *Mastenbroek et al.* in [55].

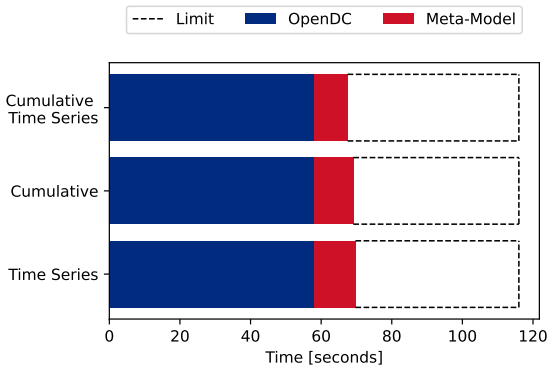
We establish the main requirement of the system to leverage models' predictions, plot the Multi-Model, and output the results in less than the simulation time of OpenDC. We also establish the main requirement of leveraging models' predictions, computing the Meta-Model, and output in two different formats, also in less time than the simulation time of OpenDC. This equivalates generating the Multi-Model, or meta-simulating a Meta-Model, in less than 70x-300x the simulation time of CloudSim Plus [67].

We analyze the performance of OpenDC-M3SA integration, using using Experiment 2[31]. In this experiment, we simulate the SURF-SARA workload, consisting of 7 days of real-world datacenter operation time.

In Figure 33, we present the performance results. OpenDC can simulate a 7-day workload in just tens of seconds, four orders of magnitude faster than the equivalent real-life operation. The simulation time of OpenDC is represented in blue in the plot. Additionally, we display the time the Multi-Model took to combine 4 simulation models, plot, and output. Similarly, we plot the time it took the Meta-Model to aggregate, compute a new model, and generate output, both as a *parquet* file and as a plot. Both the Multi-Model and the Meta-Model operate in a matter of seconds, which is one order of magnitude less than the simulation time. We also plot, with dotted borders, the thresholds set by the non-functional requirements (NFR1 of Multi-Model, NFR1 of Meta-Model).



((a)) Multi-Model Time.



((b)) Meta-Model Time.

Figure 33: Individual performance evaluation, for Multi-Model and Meta-Model, applied to OpenDC, against the performance threshold.

[31] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-2-window-performance-analysis

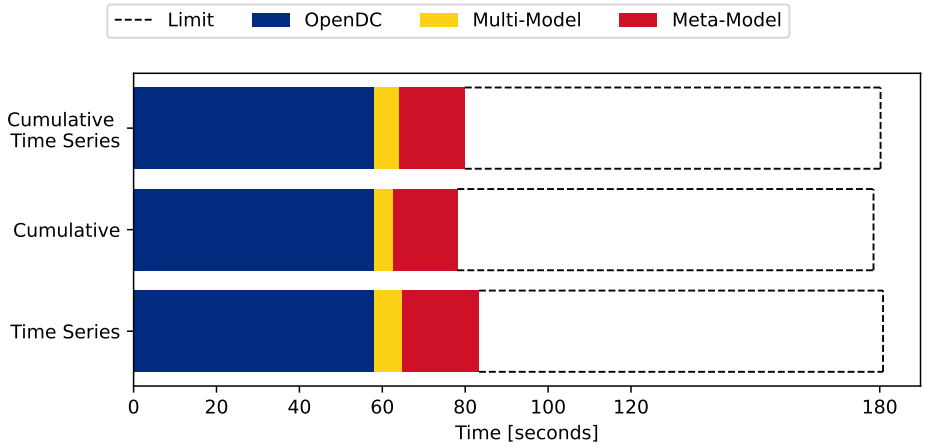


Figure 34: M3SA, applied to OpenDC, against the performance threshold.

In Figure 34, we plot the simulation time of OpenDC, together with the performance of the Multi-Model and Meta-Model, and the requirement-set time (performance) limit. The time limit for computing both a Multi-Model and a Meta-Model is twice the time needed for OpenDC to simulate. Our system computes, analyzes, and outputs faster than the time limit and utilizes less than 25% of the allocated time.

In Section 3.7.5 and Section 4.7.8, as part of Experiment 2, and Experiment 7[32], we simulate 694 computation days (the equivalent of 201,600 samples, at the industry standard sampling rate of 5 minutes), in only 15.7 minutes. We compute the Multi-Model and the Meta-Model, each, in 3-4 minutes, depending on the type of plot and the window size. All the computation has been conducted on a regular-user machine (i.e., not a supercomputer). This presents and emphasizes the efficiency of the system and the ability to provide "in-meeting, near-interactive, same-day" (meta-)simulation results, even for massive-scale infrastructure and simulation.

Our main performance findings are:

MF5	OpenDC is 70x-330x faster than a widely-used cloud simulated in the field, for the workloads analyzed by Mastenbroek et al. [55].
MF6	OpenDC can simulate 3 months of datacenter operation in a matter of seconds and 2 years in a matter of minutes, thus enabling interactive risk exploration for practitioners for both large- and massive-scale experiments.
MF7	Multi-Model leverages, computes, and outputs, in less than 25% of the simulation time of OpenDC.
MF8	Meta-Model leverages, computes, and outputs (data file and plot), in less than 25% of the simulation time of OpenDC.
MF9	M3SA, as embedded in OpenDC, operates one order of magnitude faster than OpenDC for month- and year-scale workloads.

5.4 Accuracy Validation

In this section, we discuss the validity of the results produced by M3SA, as integrated to OpenDC.

[32] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-7-metamodel-performance-analysis

5.4.1 Multi-Model Results Validity

The Multi-Model is assembled by leveraging other models' predictions into a singular tool. The validity of the Multi-Model's predictions depends directly on the validity of the individual models whose predictions it is using. OpenDC, as a peer-reviewed simulator, with top-tier related publications [56, 39, 6, 57], and awards [34, 55], employs numerous highly accurate power models. The accuracies of employed models have been tested and validated in numerous publications against real-world data [34, 55, 5]. Thanks to the mathematical libraries used in the prediction process, the precision of the models is constant.

Our main findings of the subsection are:

MF10	The validity of the Multi-Model is directly dependent on the validity of the models from the simulator the model is applied to.
MF11	The high validity of the power models employed in OpenDC implies the high validity of the Multi-Model integration with OpenDC.

5.4.2 Meta-Model Results Validity

Meta-Models apply meta-functions on multiple models' predictions, synthesizing accurate results at the user-selected granularity. In this paper, we research, integrate, and analyze two such meta-functions widely used in other sciences, yet not in ICT. We engineer an accuracy validation tool to assess the accuracy of both individual models and Meta-Models, against real-world data, using three statistical analysis metrics: Mean Absolute Percentage Error (*MAPE*), Normalized Absolute Differences (*NAD*), and Root Mean Square Logarithmic Error (*RMSLE*).

We set the non-functional requirement for the Meta-Model to be more accurate than the average simulation model (NFR2). Therefore, the error rate of any Meta-Model must be lower than the error rate of the average individual model used in the meta-simulation.

In Experiment 5[33], we compute Meta-Models based on the predictions of 8 individual models (Meta-Model NFR2) with various configurations and, thus, various accuracies. Similarly, in Experiment 8[34], we compute Meta-Models using 16 individual models (Meta-Model NFR2) with various setups and accuracy. We compute Meta-Models using the *mean* and *median* meta-function.

We plot the accuracy evaluations in Figure 35. On the horizontal axis, we plot the model ID, while on the vertical axis, we plot the corresponding accuracy of the model. We plot, with a red dotted line, the average individual model error (NFR2 of the Meta-Model) and color the area green, representing the area in which the accuracy of the Meta-Models must situate to meet the set requirements. We represent the accuracy of the computed Meta-Models with full horizontal lines; the Meta-Models are generated with the *mean*, respectively *median* meta-functions.

[33] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-5-accuracy-evaluation
[34] https://github.com/Radu-Nicolae/opendc/tree/local-master/experiments/experiment-8-metamodel-accuracy-analysis

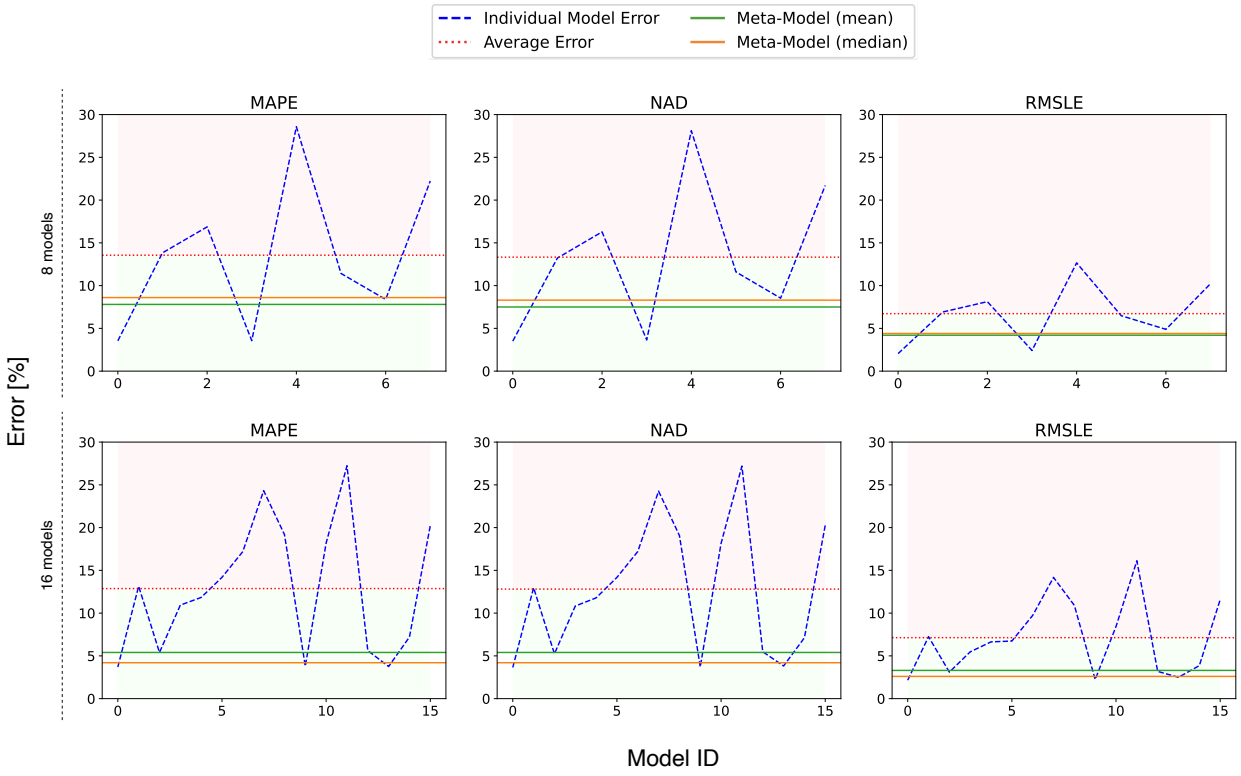


Figure 35: Accuracy validation for the Meta-Model using 8 and 16 individual models, evaluated using MAPE, NAD, and RMSLE metrics.

For all the Meta-Models leveraging both 8 and 16 models, using both *mean* and *median* meta-functions, and for each of the accuracy metrics, we identify the Meta-Model as simulating with a (sometimes significantly) smaller error rate (i.e., higher accuracy) than the average individual model.

Our main findings of the subsection are:

MF12	The validity of the Meta-Model is indirectly dependent on the validity of the models from the simulator it is applied to.
MF13	The validity of the Meta-Model is indirectly dependent also on the validity of the meta-functions used for data aggregation.
MF14	Applied on 8 and, respectively, on 16 individual models supported by OpenDC, the Meta-Model simulates a higher accuracy than the average simulation model for all the supported meta-functions.

5.5 Usability Evaluation

In this subsection, we evaluate the usability of M3SA with a hypothetical use case scenario. We identify usability as a requirement critical for the massive-scale adoption of M3SA and critical for becoming the new industry standard in ICT simulations.

Context

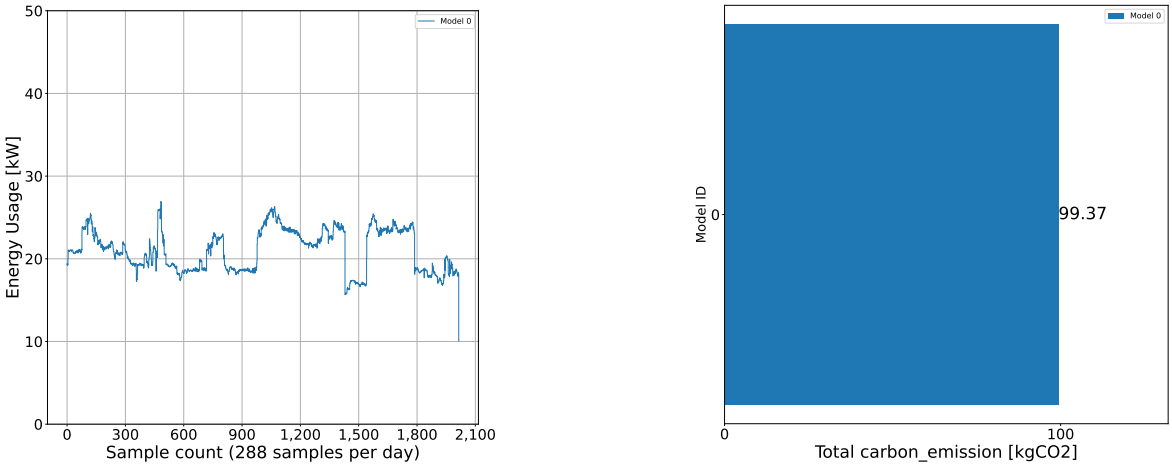
We propose a scenario in which datacenter operators want to double the scale of the infrastructure, thus allowing faster execution of larger workloads. The current infrastructure contains 140 hosts, each containing one CPU operating at 3.0 GHz and 512 GB of memory. The desired infrastructure contains 280 identical

hosts. In this use case scenario, we assume the workload used for simulation is SURF-SARA, a 1-week workload with a 300 seconds sampling rate. We make the input and output data publicly available on GitHub[35].

Due to contractual limitations, the infrastructure cannot exceed 30 KWh; otherwise, the infrastructure operators would support additional costs five times higher per exceeded KWh. Moreover, due to local regulations, the infrastructure is not permitted to exceed 100 kg of CO2 emissions per week; if this limit is not respected, the infrastructure may be temporarily or permanently shut down, depending on the severity of the exceed.

The infrastructure has a built-in monitoring system, which provides the operators with insights into various metrics, including energy usage and CO2 emissions. The system monitors the infrastructure at 300-second intervals, following the industry standards [5, 55].

Single-Model Simulation
The operators use the OpenDC simulator integrated to estimate energy usage and carbon estimations. They simulate the upgraded infrastructure, containing 280 hosts, using a single model. Although not designed for single-model predictions, M3SA supports simulation analysis (e.g., plotting) for single models. We present the simulation results in Figure 36.



((a)) Energy Usage.

((b)) Total CO2 emissions.

Figure 36: M3SA predictions for the scaled infrastructure, using a single model.

The operators observe that energy usage never exceeds the limit of 30 KW; however, although the total CO2 emissions predicted by the model are situated within the limits, the total CO2 emissions are less than 1% away from exceeding the law-mandated limits.

M3SA - Multi-Meta-Model Simulation Analyzer
To ensure more accurate simulations, the operators rerun the simulation, but this time using M3SA, with eight simulation models supported by OpenDC. They set up M3SA to compute a Multi-Model for energy usage (Listing 10) and CO2 emissions (Listing 11), with both data customization (e.g., unit scaling) and plotting customization (e.g., axis, ticks). To analyze energy consumption at a continuous sampling rate, the operators configure M3SA to output a time series graph, shown in Figure 37a. The operators select a cumulative graph for analyzing the total CO2 emissions, shown in Figure 37b.

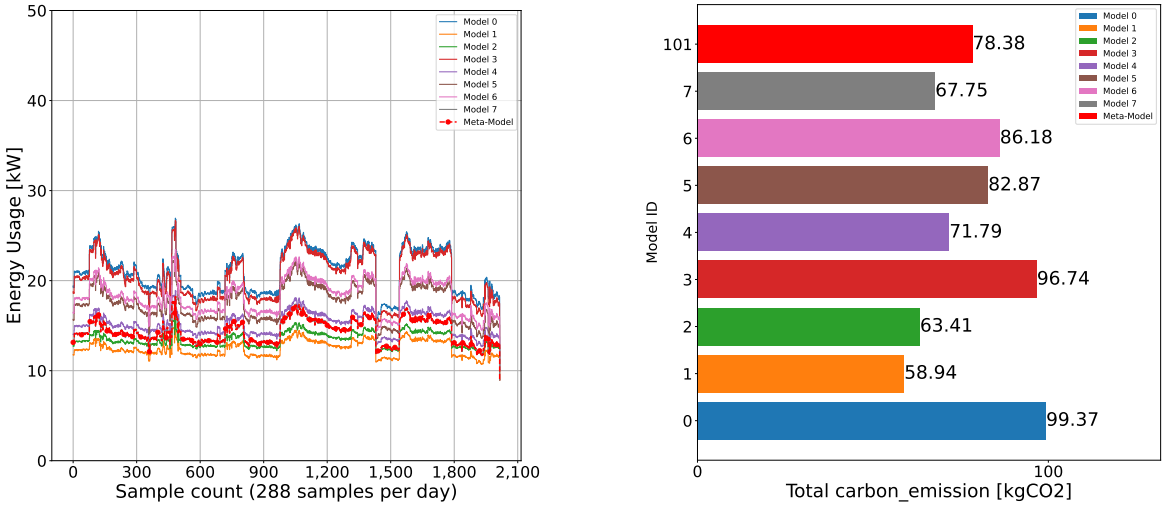
[35] `https://github.com/Radu-Nicolae/opendc/tree/27-m3sa-usability-evaluation/experiments/use-case-scenario`

Listing 10: M3SA setup energy usage.

```
{
"multimodel": true,
"metamodel": false,
"metric": "power_draw",
"window_size": 1,
"window_function": "mean",
"meta_function": "mean",
"samples_per_minute": 0.2,
"plot_type": "time_series",
"x_label": "Sample count
(288 samples per day)",
"y_label": "Energy Usage [kW]",
"y_min": 0,
"y_max": 50,
"x_ticks_count": 8,
"plot_title": "",
"current_unit": "W",
"unit_scaling_magnitude": 3
}
```

Listing 11: M3SA setup CO2 emissions.

```
{
"multimodel": true,
"metamodel": false,
"metric": "carbon_emission",
"window_size": 1,
"window_function": "mean",
"meta_function": "mean",
"samples_per_minute": 0.2,
"plot_type": "cumulative",
"x_label": "Sample count
(288 samples per day)",
"y_label": "CO2 Emissions [kgCO2]",
"x_min": 0,
"x_max": 120,
"x_ticks_count": 1,
"plot_title": "",
"current_unit": "gCO2",
"unit_scaling_magnitude": 3
}
```



((a)) Energy Usage.

((b)) Total CO2 emissions.

Figure 37: M3SA predictions for the expanded infrastructure, using M3SA.

Due to the system's performance, the operators receive the simulator's prediction for seven days of operation in less than 10 seconds and the analysis of M3SA in 0.8 seconds. This enables the operators to receive in-meeting, near-interactive, same-day predictions. The operators observe varying predictions, yet all are within the limits the infrastructure must adhere to. The operators trust the predictions of M3SA, especially the predictions of the Meta-Model, which estimates energy usage constantly under 20 KWh (instead of the 30 KWh limit) and total CO2 emissions of 78.38 kgCO2 - 22% lower than the limit.

Decisions and results

The operators decide to approve the expansion and double the scale of the infrastructure. After the expansion, the infrastructure receives a workload of similar size and complexity to SURF-SARA, executed over seven days. The energy and CO2 limits are met: the energy never exceeds 18 kWh, and the total carbon emissions

are 76 kgCO2. This would result in an error rate of the Meta-Model of around 3%.

Our main findings in this subsection are:

MF15	M3SA can be used to analyze both simulations using one model and simulations using multiple models.
MF16	All functional and non-functional requirements of M3SA contribute, some critically, to the tool's usability.

5.6 Universality Evaluation

In this subsection, we evaluate the universality of M3SA. While envisioning novel simulation concepts and engineering a tool able to materialize such concepts is a major scientific contribution, in our research philosophy, this is insufficient for universal science. We design M3SA, a tool with Multi-Model and Meta-Model simulation capabilities, for integration with any simulator.

In Section 3.3, we analyze two design choices and the trade-off universality-performance each architecture poses. We choose the architecture that best meets the universality requirements of the system (Multi-Model FR3, Meta-Model FR5) while still meeting the performance thresholds. Throughout the engineering process, we develop software that is simple to integrate, adapt, and expand.

We identify a potential architecture that receives in-code data (e.g., populated instances of classes) as challenging to port and adapt between simulators. We further identify and adopt an architecture that uses data files (the output generated by the simulator) instead of embedding a tool in the simulator's code architecture. This architecture allows M3SA to be a tool simple to port between simulators.

To simplify the adaptability process even further, we design an integration document, available on the GitHub repository of our tool[36], containing in-detail adaptability tutorial. We further expand on the integration process in Section 5.2.3. We provide clear, fully documented code, following state-of-the-art (industry) standards (Section 5.2.1), which makes the current codebase a robust foundation for further enhancement of M3SA.

Our main findings in this subsection are:

MF17	M3SA can be integrated with any simulator, yet minimal changes to the simulator need to be made.
MF18	The state-of-the-art engineering of M3SA allows future codebase expansion, as part of potential research or engineering efforts.

5.7 Discussion

5.7.1 Summary

We present a working prototype of M3SA, following the design and requirements established in Section 3 and Section 4. Our integration and experiments demonstrate that the researched and engineered system can assist complex cloud simulations and enhance prediction accuracy and comprehension. Our system is able to combine multiple simulation predictions in a unified plot and able to generate meta-simulation models using other models' predictions. M3SA analyzes one order of magnitude faster than one of the fastest existing datacenter simulators, which confirms the tool's performance. We prove the high accuracy of the proposed and implemented meta-simulation concept. Finally, we highlight the potential of such a simulation concept and tool with practical examples and emphasize the universality of M3SA, which can be applied to any simulator.

[36] `https://github.com/Radu-Nicolae/opendc/blob/local-master/site/docs/documentation/M3SA-integration-tutorial.md`

5.7.2 Threats to validity

Facilitating reproducible experiments and science, we design and use simple tests for the validity of the models integrated into OpenDC, which primarily focus on precision (accuracy is guaranteed by the math libraries used in the OpenDC project)[55]. However, the precision of the individual models, and, thus, the precision of the models' leverage within the Multi-Model and the Meta-Model, could be threatened by the impossibility of comparing directly with large-scale infrastructure. To do so, it would cost between 7 and 10 orders of magnitude more resources, both financial and climatic [38, 55].

Besides, the lack of exhaustive details about the workload traces and datacenter configuration threatens the validity of the accuracy evaluation process. M3SA applied on OpenDC may have different predictions than if it were applied on a different datacenter simulator; this could result in potential prediction discrepancies. Besides, we acknowledge that no simulator reproduces real-world scenarios with perfect precision. Specific configurations or operational contexts may limit the precision of the models employed by the simulation infrastructure.

6

Conclusion and Future Work

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

6.1 Conclusion

In this work, we investigate three main research questions concerning the analysis of ICT simulation predictions. We first describe the vulnerabilities of idiosyncratic simulation models in Chapter 1, then provide a literature survey and a comprehensive background on the topic in Chapter 2. In the remaining chapters, we propose M3SA, the Multi-Meta-Model Simulation Analyzer. In Chapter 3, we design the Multi-Model, a novel simulation concept and tool that proposes leveraging the predictions of multiple models. In Chapter 4, we design the Meta-Model, another novelty that materializes the concept of meta-simulation. In Chapter 5, we integrate M3SA into OpenDC and evaluate the overall system.

We now answer each of the research questions, in turn:

RQ1 **How to design a Multi-Model simulator that leverages the results of singular models?**
We identify a gap between the employed simulation concepts in ICT and other sciences. In Chapter 2, we conduct a literature survey on how other sciences conduct simulations by leveraging multiple models. We then conduct a literature survey on individual models and widely used, performant, and accurate ICT simulators. Then, in Chapter 3, we set requirements for the Multi-Model tool. Further, we envision the concept of using multiple simulation models for datacenters and analyze design choices to meet the set requirements. We propose the concept of windowing aggregation and analyze related trade-offs. We design the Multi-Model towards performance, universality, and usability as a tool that provides in-meeting analysis with comprehensive functionality and simple integration into any simulator. Then, we evaluate the Multi-Model against the set requirements. The Multi-Model is a critical component of M3SA.

RQ2 **How to design a Meta-Model simulator that combines the outputs of the Multi-Model for a more accurate prediction?**
We identify significant research opportunities facilitated by the Multi-Model concept. In Chapter 4, we conduct a requirement analysis of the Meta-Model concept and tool. Based on these requirements, we envision the concept of simulation using Meta-Models. We then design the Meta-Model to be embedded into the Multi-Model. Further, we present and analyze the concept of meta-simulation and meta-functions. We embed the Meta-Model in the M3SA system and design towards performance, universality, and usability. Then, we evaluate the Meta-Model against the set requirements. The Meta-Model, together with the Multi-Model, form M3SA.

RQ3 How to integrate and evaluate M3SA?
We identify the main requirement of integrating and evaluating the engineered version of M3SA and its components, the system designed in Chapter 3 and Chatper 4. In Chapter 5, we present a working software prototype of M3SA and the state-of-the-art software engineering process used in the development phase. We integrate M3SA with OpenDC and presented the process of tool integration. Then, we evaluate M3SA against performance, accuracy, universality, and usability to ensure in-meeting, real-time predictions, highly accurate predictions, integration simplicity with any datacenter simulator, and comprehensive functionality.

We release M3SA as free and open-source software for the community to use. M3SA follows modern, state-of-the-art engineering processes in the industry and produces reproducible results.

6.2 Future Work

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

1. We intend to keep improving M3SA towards becoming a production-ready system for simulation analysis of cloud infrastructure. We are continuously engineering M3SA with additional features towards extensive error handling, improved enhanced performance, and better overall user experience. Our goal is to make M3SA widely adoptable for diverse stakeholders by allowing dynamic and interactive simulation analysis of any scenario, datacenter topology, and workload.
2. We investigate new meta-functions that leverage prediction data vertically and horizontally, with a higher accuracy (i.e., less error rate) than the existing meta-function. Due to the design of the codebase, integrating new meta-functions does not affect the core functionality. This minimizes the engineering efforts of the researcher when integrating and testing meta-functions.
3. We plan to investigate efficient ways of integrating Machine Learning techniques in the process of meta-simulation towards generating meta-predictions with higher accuracy than the predictions of singular models and higher predictions than the already enabled meta-functions (e.g., mean, median).
4. Due to the enhanced performance of M3SA, we envision the tool's usage in real-time decision-making in datacenters for a wide range of stakeholders (e.g., projecting a datacenter, expanding a datacenter, scheduling large-scale workloads). We plan to investigate the usage of M3SA and OpenDC with real-life, large- and massive-scale experiments consisting of real-life infrastructure and workloads.
5. We plan to develop educative material around M3SA and OpenDC as a series of interactive workshops, seminars, and assignments that educate students on the operation datacenters, the design process, and the impact of design decisions in datacenters. Available as free software, both M3SA and OpenDC can be accessed by students who can engage in activities as part of Computer Systems courses, either at Bachelor's or Master's level. This would provide students with new perspectives, essential in a potential career towards *Massivizing Computer Systems*.

References

[1] ACM Technology Council. Computing and climate change. *ACM*, 2021.

[2] E. E. Agency. Share of energy consumption from renewable sources. 2023. Accessed: May 4, 2024.

[3] I. E. Agency. Countries, 2023. Accessed on: May 28, 2023.

[4] T. F. Allen and T. B. Starr. *Hierarchy: perspectives for ecological complexity*. University of Chicago Press, 2019.

[5] G. Andreadis. Capelin: Fast data-driven capacity planning for cloud datacenters. 2020.

[6] G. Andreadis, F. Mastenbroek, V. van Beek, and A. Iosup. Capelin: Data-driven capacity procurement for cloud datacenters using portfolios of scenarios–extended technical report. *arXiv preprint arXiv:2103.02060*, 2021.

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

[8] G. Andreadis, L. Versluis, F. Mastenbroek, and A. Iosup. A reference architecture for datacenter scheduling: Extended technical report. *arXiv preprint arXiv:1808.04224*, 2018.

[9] V. Avelar, D. Azevedo, A. French, and E. N. Power. Pue: a comprehensive examination of the metric. *White paper*, 49, 2012.

[10] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy. *Site reliability engineering: How Google runs production systems*. " O'Reilly Media, Inc.", 2016.

[11] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster computing*, 18:385–402, 2015.

[12] R. Bukht and R. Heeks. Defining, conceptualising and measuring the digital economy. *Development Informatics working paper*, (68), 2017.

[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[14] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, 2014.

[15] G. G. Castañé, A. Nunez, P. Llopis, and J. Carretero. E-mc2: A formal framework for energy modelling in cloud computing. *Simulation Modelling Practice and Theory*, 39:56–75, 2013.

[16] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[17] J. Cho, B. Park, and Y. Jeong. Thermal performance evaluation of a data center cooling system under fault conditions. *Energies*, 12(15):2996, 2019.

[18] Cisco Systems. White paper on the evolution and future of data centers. White paper, Cisco Systems, 2023. Accessed: 2023-04-30.

[19] Climate Neutral Data Centre. Home, 2023. Accessed: 2023-05-28.

[20] COVID-19 Forecast Hub. Covid-19 forecast hub, 2023. Accessed: Dec 2023.

[21] E. Y. Cramer, Y. Huang, Y. Wang, E. L. Ray, M. Cornell, J. Bracher, A. Brennen, A. J. C. Rivadeneira, A. Gerding, K. House, et al. The united states covid-19 forecast hub dataset. *Scientific data*, 9(1):462, 2022.

[22] Data Center Dynamics. Surgeries and procedures paused at wichita hospitals due to data center outage, 2023. Accessed: 2024-04-30.

[23] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, 2016.

[24] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news*, 35(2):13–23, 2007.

[25] R. Ferreira da Silva, A.-C. Orgerie, H. Casanova, R. Tanaka, E. Deelman, and F. Suter. Accurately simulating energy consumption of i/o-intensive scientific workflows. In *Computational Science–ICCS 2019: 19th International Conference, Faro, Portugal, June 12–14, 2019, Proceedings, Part I 19*, pages 138–152. Springer, 2019.

[26] J. D. Ford, W. Vanderbilt, and L. Berrang-Ford. Authorship in ipcc ar5 and its implications for content: climate change and indigenous populations in wgii. *Climatic change*, 113:201–213, 2012.

[27] M. Fowler. Continuous integration, 2006. Accessed: 2024-07-29.

[28] Google. Data center efficiency, 2023. Accessed: 2023-05-06.

[29] Google Developers. Contributing to blockly: Getting started with commits, 2024. Accessed: 2024-07-29.

[30] Google Developers. Contributing to blockly: Writing a good pull request, 2024. Accessed: 2024-07-29.

[31] T. Guérout, T. Monteil, G. Da Costa, R. N. Calheiros, R. Buyya, and M. Alexandru. Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, 39:76–91, 2013.

[32] S. K. Gupta, R. R. Gilbert, A. Banerjee, Z. Abbasi, T. Mukherjee, and G. Varsamopoulos. Gdcsim: A tool for analyzing green data center design and resource management techniques. In *2011 International Green Computing Conference and Workshops*, pages 1–8. IEEE, 2011.

[33] X. A. Harrison, L. Donaldson, M. E. Correa-Cano, J. Evans, D. N. Fisher, C. E. Goodwin, B. S. Robinson, D. J. Hodgson, and R. Inger. A brief introduction to mixed effects modelling and multi-model inference in ecology. *PeerJ*, 6:e4794, 2018.

[34] H. He. Modelling energy consumption in the opendc datacenter simulator for analyzing energy-aware cloud infrastructure, 5 2021. Honours Programme, Research Thesis.

[35] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A.-C. Orgerie, and M. Quinson. Predicting the energy-consumption of mpi applications at scale using only a single node. In *2017 IEEE international conference on cluster computing (CLUSTER)*, pages 92–102. IEEE, 2017.

[36] IEA. Data centres and data transmission networks. International Energy Agency (IEA) Report, 2022. Available online.

[37] International Energy Agency. Data centres and networks. `https://www.iea.org/fuels-and-technologies/data-centres-networks`. Accessed: July 4, 2023.

[38] A. Iosup. Massivizing computer systems. Keynote Presentation, 2021. Available online at: `https://atlarge-research.com/pdfs/pres-20210204-aiosup-massivizing.pdf`.

[39] A. Iosup, G. Andreadis, V. Van Beek, M. Bijman, E. Van Eyk, M. Neacsu, L. Overweel, S. Talluri, L. Versluis, and M. Visser. The opendc vision: Towards collaborative datacenter simulation and exploration for everybody. In *2017 16th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 85–94. IEEE, 2017.

[40] A. Iosup, L. Versluis, A. Trivedi, E. V. Eyk, L. Toader, V. van Beek, G. Frascaria, A. Musaafir, and S. Talluri. The atlarge vision on the design of distributed systems and ecosystems. *CoRR*, abs/1902.05416, 2019.

[41] JetBrains. Kotlin programming language, 2024. Accessed: 2024-07-29.

[42] G. Kecskemeti, W. Hajji, and F. P. Tso. Modelling low power compute clusters for cloud simulation. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 39–45. IEEE, 2017.

[43] J. Kim, B. P. Mohanty, and Y. Shin. Effective soil moisture estimate and its uncertainty using multimodel simulation based on bayesian model averaging. *Journal of Geophysical Research: Atmospheres*, 120(16):8023–8042, 2015.

[44] D. Kliazovich, P. Bouvry, and S. U. Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62:1263–1283, 2012.

[45] Kotlin Team. Kotlin documentation. `https://kotlinlang.org/docs/kotlin-doc.html`, 2024. Accessed: 2024-07-12.

[46] T. Krishnamurti, V. Kumar, A. Simon, A. Bhardwaj, T. Ghosh, and R. Ross. A review of multimodel superensemble forecasting for weather, seasonal climate, and hurricanes. *Reviews of Geophysics*, 54(2):336–377, 2016.

[47] K. Kurowski, A. Oleksiak, W. Piatek, T. Piontek, A. Przybyszewski, and J. Weglarz. Dcworms–a tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, 39:135–151, 2013.

[48] H. M. Ljungqvist, M. Risberg, A. Toffolo, and M. Vesterlund. A realistic view on heat reuse from direct free air-cooled data centres. *Energy Conversion and Management: X*, 20:100473, 2023.

[49] B. Louis, K. Mitra, S. Saguna, and C. Åhlund. Cloudsimdisk: Energy-aware storage simulation in cloudsim. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 11–15. IEEE, 2015.

[50] Y. Ma, X. Xie, and M. Zhang. A review of failure prediction in distributed data centers. In *Asian Simulation Conference*, pages 497–509. Springer, 2022.

[51] A. W. Malik, K. Bilal, S. Malik, Z. Anwar, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya. Cloudnetsim++: A gui based framework for modeling and simulation of data centers in omnet++. *IEEE Transactions on Services Computing*, 10(4):506–519, 2015.

[52] C. Malone and C. Belady. Proceedings of 2006 digital power forum richardson tx. *Metrics to Characterize Data Center IT Equipment Energy Use*, 2006.

[53] A. Markus, A. Kertesz, and G. Kecskemeti. Cost-aware iot extension of dissect-cf. *Future Internet*, 9(3):47, 2017.

[54] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[55] F. Mastenbroek. Radice: Data-driven risk analysis of sustainable cloud infrastructure using simulation. 2022.

[56] F. Mastenbroek, G. Andreadis, S. Jounaid, W. Lai, J. Burley, J. Bosch, E. Van Eyk, L. Versluis, V. Van Beek, and A. Iosup. Opendc 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 455–464. IEEE, 2021.

[57] F. Mastenbroek, T. D. Matteis, V. van Beek, and A. Iosup. Radice: A risk analysis framework for datacenters. *IEEE Transactions on Cloud Computing*, 2023.

[58] G. Myhre, W. Aas, R. Cherian, W. Collins, G. Faluvegi, M. Flanner, P. Forster, Ø. Hodnebrog, Z. Klimont, M. T. Lund, et al. Multi-model simulations of aerosol and ozone radiative forcing due to anthropogenic emission changes during the period 1990–2015. *Atmospheric Chemistry and Physics*, 17(4):2709–2720, 2017.

[59] D. Niewenhuis, S. Talluri, A. Iosup, and T. de Matteis. Footprinter: Quantifying data center carbon footprint. 2024.

[60] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10:185–209, 2012.

[61] Oracle. Working with planning: Mape. `https://docs.oracle.com/en/cloud/saas/planning-budgeting-cloud/pfusu/insights_metrics_MAPE.html#GUIDC33B0F01-83E9-468B-B96C-413A12882334`, 2014. Accessed: insert date here.

[62] J. Ousterhout. *A Philosophy of Software Design*. Yaknyam Press, 1 edition, 2018.

[63] N. G. Reich, L. C. Brooks, S. J. Fox, S. Kandula, C. J. McGowan, E. Moore, D. Osthus, E. L. Ray, A. Tushar, T. K. Yamana, et al. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the united states. *Proceedings of the National Academy of Sciences*, 116(8):3146–3154, 2019.

[64] D. Reinsel, J. Gantz, and J. Rydning. Data age 2025: The evolution of data to life-critical. don't focus on big data. *2*, 2017.

[65] R. Schunk, L. Scherliess, V. Eccles, L. Gardner, J. Sojka, L. Zhu, X. Pi, A. Mannucci, M. Butala, B. Wilson, et al. Space weather forecasting with a multimodel ensemble prediction system (meps). *Radio Science*, 51(7):1157–1165, 2016.

[66] M. A. Siddiqui, M. Y. Dahab, and O. A. Batarfi. Building a sentiment analysis corpus with multifaceted hierarchical annotation. *International Journal of Computational Linguistics (IJCL)*, 6(2):11–25, 2015.

[67] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire. Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)*, pages 400–406. IEEE, 2017.

[68] Statista. Data centers worldwide by country 2021. `https://www.statista.com/statistics/1228433/data-centers-worldwide-by-country/`. Accessed: July 4, 2023.

[69] Statista. Data center average annual pue worldwide, 2023. Accessed: 2023-10-01.

[70] Statista. Most used programming languages among developers worldwide, 2023. Accessed: 2024-07-29.

[71] Statista. Electricity prices in selected countries. `https://www.statista.com/statistics/263492/electricity-prices-in-selected-countries/`, 2024. Accessed: 2024-05-01.

[72] Statista. Data center average annual pue worldwide from 2008 to 2020, n.d.

[73] J. Summers, A. Kozma, et al. Holistic cooling at the world's most efficient data center. *Data Center Dynamics*, Oct 2019.

[74] The Apache Software Foundation. Apache parquet, 2024. Accessed: 2024-07-20.

[75] The Register. Google outage: internet traffic plunges 40 Accessed: 2024-04-30.

[76] The Register. Overheating data center in singapore causes widespread disruption, Nov 2023. Accessed: 2024-04-30.

[77] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun. A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. *IEEE Transactions on Automation Science and Engineering*, 12(1):153–161, 2013.

[78] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya. Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 385–392. IEEE, 2012.

[79] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *2011 IEEE International Conference on High Performance Computing and Communications*, pages 279–287. IEEE, 2011.

[80] M. F. Triola. *Elementary Statistics*. Pearson New International Edition, 12 edition, 2013.

[81] G. van Rossum. Python style guide. `https://www.python.org/doc/essays/styleguide/`, 2024. Accessed: 2024-07-12.

[82] L. D. Williams. Concepts of digital economy and industry 4.0 in intelligent and information systems. *International Journal of Intelligent Networks*, 2:122–129, 2021.

[83] Y. Zhao, Y. Wang, J. Liu, H. Xia, Z. Xu, Q. Hong, Z. Zhou, and L. Petzold. Empirical quantitative analysis of covid-19 forecasting models. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 517–526. IEEE, 2021.

[84] R. Zhou, Y. Shi, and C. Zhu. Axpue: Application level metrics for power usage effectiveness in data centers. In *2013 IEEE International Conference on Big Data*, pages 110–117. IEEE, 2013.