OpenDT: Exploring Datacenter Performance and Sustainability with a Self-Calibrating Digital Twin

Extended Technical Report

Radu Nicolae*, Jules van der Toorn*, Stavriana Kraniti*, Houcen Liu*, Alexandru Iosup+
{r.nicolae,j.vander.toorn,s.kraniti,h.liu19}@student.vu.nl
a.iosup@vu.nl
VU University Amsterdam
Amsterdam, The Netherlands

Abstract

Datacenters are the backbone of our digital society, but raise numerous operational challenges. We envision digital twins becoming primary instruments in datacenter operations, continuously and autonomously helping with major operational decisions and with adapting ICT infrastructure, live, with a human-in-the-loop. Although fields such as aviation and autonomous driving successfully employ digital twins, an open-source digital twin for datacenters has not been demonstrated to the community. Addressing this challenge, we design, implement, and experiment using OpenDT, an Open-source, Digital Twin for monitoring and operating datacenters through a continuous integration cycle that includes: (1) live and continuous telemetry data; (2) discrete-event simulation using live telemetry from the physical ICT, with self-calibration; and (3) SLO-aware and human-approved feedback to physical ICT. Through trace-driven experiments with a prototype mainly covering stages 1 and 2 of the cycle, we show that (i) OpenDT can be used to reproduce peer-reviewed experiments and extend the analysis with performance and energy-efficiency results; (ii) OpenDT's online re-calibration can increase digital-twinning accuracy, quantified to a MAPE 4.39% vs. 7.86% in peer-reviewed work. OpenDT adheres to FAIR/FOSS principles and is available at: https://github.com/atlarge-research/opendt/tree/hcp.

Keywords

OpenDT, datacenters, digital twins, simulation, calibration, performance, sustainability, energy utilization, efficiency.

1 Introduction

Our digitalized society and economy increasingly rely on digital services running in increasingly larger datacenters [7, 18, 20]. Operators use system analysis, increasingly based on simulation, for designing and operating datacenters [20, 23, 29]. Using simulation as the core of digital twins that mimic datacenter conditions, and support analysis and decision-making in datacenter operations, is an open, emerging challenge [25]. In this work, we design, prototype, and experiment using OpenDT, an Open-source Digital Twin for datacenters.

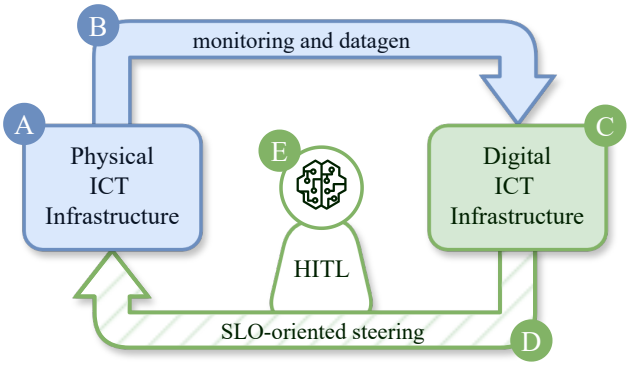

Figure 1: High-level overview of datacenter digital twinning: (1) The Physical ICT infrastructure collects telemetry data, (2) the Digital ICT infrastructure ingests telemetry and related data, then (3) mimics (twins) the operation of the Physical ICT, and (4) offers back SLO-aware adjustment feedback. A human-in-the-loop (HITL) oversees the process.

Simulation already supports datacenter operators and scientists with timely and cost-efficient experimentation and analysis [19, 23]. For example, in the European Horizon project Graph Massivizer, simulators predict speedup, failure cost, energy consumption, and CO2-emissions for massive-scale infrastructure [11, 21, 25, 29].

Although simulators are valuable for detailed, realistic analysis of datacenters under workload, in ICT, there is currently no closed-loop process that continuously ingests live telemetry, updates the state inside the simulation, and calibrates the simulator so the multi-metric analysis matches reality. Instead, these steps must be taken independently, with considerable delays and potential errors particularly at boundaries.

Figure 1 illustrates the *digital twinning process* that informed the high-level design of OpenDT. This process involves continuous communication between a physical twin (label Ⓐ) and a digital twin (Ⓒ). The physical twin generates telemetry data (Ⓑ), either obtained from the ICT infrastructure's monitoring and logging systems, or generated from it via advanced analytics or AI-based processes; telemetry data is ingested into the digital twin to be processed directly, or placed in a larger data pool. The digital twin

*These authors contributed equally to this research.
+Conceptual contribution and supervision.

Submitted for peer-review, arxiv.org version

ACM ISBN 978-1-4503-XXXX-X/2026/06
https://doi.org/XXXXXXX.XXXXXXX

uses techniques to mimic (*twin*) the operation of the ICT infrastructure, often simulation-based and leveraging SLO-oriented analysis. Closing the loop, the digital twin informs and potentially steers the physical datacenter with SLO-oriented decisions and recommendations. A *human-in-the-loop* (B), aiming to ensure the correct and ethical operation of the process, oversees at least the major decisions and can intervene at any time.

In this work, with OpenDT, we make three main contributions:

C1. Design and prototype: We design and prototype OpenDT as a datacenter digital twin. The current version of OpenDT support the cycle of continuous digital-twinning through: (1) telemetry ingestion, (2) trace- and configuration-driven, discrete-event, SLO-aware simulation for multi-metric datacenter analysis, (3) support for human-in-the-loop digital twinning. OpenDT is in its early design and prototype stages, with future work in all its main elements, in particular, closing the twinning loop with automated steering (Figure 1, D).

C2. Exploration: We validate OpenDT through real-world experimentation using its prototype. We reproduce and then extend a peer-reviewed experiment [30]. We explore how much OpenDT's self-calibration improves accuracy. All experiments use real-world workload and energy traces from SURF, the Dutch national supercomputing center [37].

C3. Open Science: We follow the principles of FAIR, FOSS, and reproducible science, and release the engineered OpenDT prototype, together with a reproducibility capsule at: https://github.com/atlarge-research/opendt/tree/hcp .

2 Design of OpenDT: an operational ecosystem for datacenter digital twinning

In this section, we first identify design requirements based on datacenter operational needs, then present the architecture of OpenDT.

2.1 Requirement and use case analysis

We identify three main stakeholders, within datacenter operation, computer systems research, and education, which we describe in use case 1 (UC1), UC2, and UC3, respectively:

(UC1) Monitoring and operation datacenters: OpenDT should aid practitioners in (1) monitoring the performance and sustainability of datacenters and (2) conducting infrastructure steering decisions, validated through simulation. OpenDT should ensure high prediction accuracy **(NFR1)**, and rapid simulation and adjustment feedback **(NFR2)**; OpenDT should enable versatile monitoring and operation, and contain multi-layer metrics for performance, sustainability, and availability **(NFR3)**.

(UC2) Researching ICT infrastructure: OpenDT should allow scientific researchers to conduct accurate **(NFR1)**, rapid **(NFR2)**, and cost-efficient, multi-layered what-if analysis of ICT infrastructure under workload **(NFR3)**. We envision future research in simulation real-time recalibration **(FR3)** and how various policies impact twinning accuracy **(NFR1)**, thus influencing practitioners in decision-making processes, and, ultimately, performance, sustainability, and availability **(NFR3)**.

(UC3) Education: OpenDT should also serve as educational material, helping future generations of capable and responsible computer scientists and engineers. OpenDT, as an open-source scientific instrument, should enable students to conduct large-scale experiments on regular-user machines, and be open-sourced, allowing students to dissect OpenDT's operational paradigm, and various metrics are twinned and predicted **(NFR3)**.

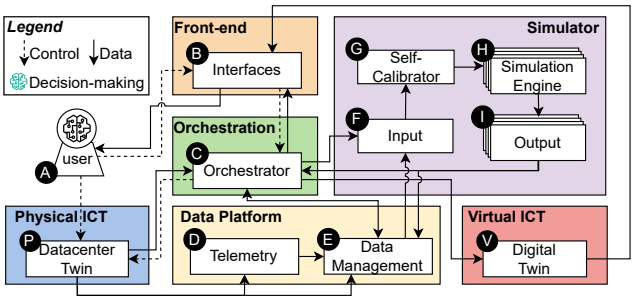

Figure 2: High-level overview of OpenDT.

Based on the established use cases, we identify the following *design requirements*:

(FR1) Digital-twinning ICT: OpenDT should ensure active and continuous replication of real-world ICT infrastructure, through a digital twin, continuously updated through telemetry data.

(FR2) State-of-the-art, discrete-event simulation: OpenDT should adopt a peer-reviewed, discrete-event simulator, able to predict performance, sustainability, and availability of datacenters under workload, at a user-established granularity.

(FR3) Simulator real-time re-calibration: OpenDT should re-calibrate predictions in real-time, based on quantified differences between predictions and real-world measurements.

(NFR1) Accurate, ground-truth adjusted predictions: OpenDT should stay within 10% error rate (community-accepted [23, 24, 29, 30]), at ≥90% of the operational time, with dynamic simulation re-calibration **(FR3)**. Inaccurate predictions can influence C-level officers to make wrong decisions [29], and lead to costly downtimes [28], impactful outages [27], and datacenter shutdowns [2].

(NFR2) Performant, lightweight digital twinning: OpenDT must be able to twin 7 days of real-world operation in under 1 hour, measured on a common machine (i.e., not a supercomputer). This supports live decisions for the engineering team.

(NFR3) Multi-layer metrics: OpenDT must quantify metrics across performance and sustainability. For each, it must provide at least two metrics. This supports diverse scenarios.

2.2 High-level design of OpenDT

We design for OpenDT the architecture depicted in Figure 2. Overall, OpenDT provides a *digital twin* state (label V) that constantly mimics the state of the *physical twin* (P) and is updated with what-if analysis results. The main process supported by this architecture is for closed-loop digital-twinning **(FR1)**, where each iteration (1) begins with telemetry data updates reflecting the physical datacenter's state (P), (2) then, telemetry data is fed into a discrete-event simulator (H) for multi-metric datacenter analysis, and into the independent self-calibration process (G), and (3) the closed-loop ends with datacenter adjustments, suggested by OpenDT (G), and validated and enforced by a human-operator (A) for major changes. (Automating the steering process, label D in Figure 1, requires careful interfacing with various kinds of physical ICT resource managers and policies, and is beyond the scope of this work.)

A key design choice for OpenDT is the simulator at its core, which is responsible for delivering high-quality predictions that represent as precisely and accurately as needed the ICT infrastructure's behaviour. OpenDT uses the state-of-the-art OpenDC [23], a peer-reviewed, open-source, discrete-event simulator **(FR2)**, with over 7 years of deployment and operation [23, 24, 29]. We augment OpenDC's predictive model with a real-time calibration process, based on quantified error rate between simulation and reality **(FR3)**.

To manage the challenges of the main process, where complexity is compounded by data-intensive telemetry and by compute-intensive simulation, OpenDT adopts an *orchestrator-centric architecture*. The Orchestrator (component Ⓒ) operates as a central authority for task scheduling and resource management, and system health monitoring, see Section 2.3.

The OpenDT twinning process is based on *windows of operation*, which are short periods of time during which the system receives telemetry data (asynchronously), runs simulations, processes results, (asynchronously) updates the UI facing the human-in-the-loop, and then continues to the next time window. Each window of operation begins with the *datacenter twin* (Figure 2, label Ⓟ), which generates telemetry data across multiple operational layers, either directly observable metrics (e.g., instantaneous power draw, available devices), or derived information (e.g., task progress, operational phenomena), ingested by Telemetry (component Ⓓ). OpenDT pre-processes the ingested telemetry, converting it to simulator-ready formats and clipping data outside the window of operation, and stores it in the *data management* component (Ⓔ). Component Ⓔ links various data-intensive components, and addresses the scale and frequency of telemetry data through a specialized solution—here, a Parquet-based shared file-storage was enough.

Continuing the sequence in the window of operation, OpenDT's orchestrator retrieves the latest collected telemetry data and feeds it into the simulator, where the *input* (Figure 2, component Ⓕ) consists of two main data entries: (1) historical telemetry and past predictions, for simulation calibration (Ⓖ), and (2) the latest telemetry for updating the datacenter state and for predicting future behaviour. One or more *simulation engines* (Ⓗ) begin making predictions, in parallel, and each produces *simulation output* (Ⓘ). Section 2.4 addresses the interplay between simulation and calibration.

The *orchestrator* (Ⓒ) publishes predictions to the *front-end interface* (Ⓑ), where the overseeing user (Ⓐ) can intervene in decision-making processes. Ultimately, also planned work for a future OpenDT version, the *orchestrator* (Ⓒ) would redirect the changes to the *datacenter twin* (Ⓟ), which modifies the physical ICT based on OpenDT's recommendations.

OpenDT supports the human-in-the-loop with several important capabilities, including scenario configuration, automated calibration processes, and multi-model simulation techniques. We detail these capabilities and the operational workflow in Section 2.7.

2.3 Orchestrator (Component C in Figure 2)

The Orchestrator acts as a central authority and is responsible for managing core operations (e.g., simulation), system monitoring, and ensuring the correctness and coherence of the operational flow of the OpenDT system for each operational cycle.

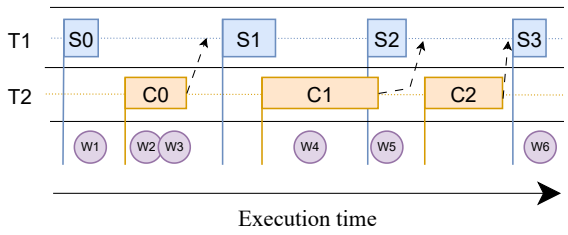

Figure 3: Synchronization between simulator (service thread $T1$) and calibrator ($T2$). The calibrator runs (events labeled C), then provides real-time feedback (dashed arrows) to the next simulation run (S), enabling dynamic recalibration based on the incoming workload tasks (W).

Time-wise, the Orchestrator manages the execution cycle around windows of operation, which are of fixed duration and lead to a *lock-step, synchronized schedule*. This approach mitigates issues such as data misalignment and ambiguity in simulation step assignments. Without this, data, which is produced by the physical twin and does not arrive all at once in the digital twin, situations could become unclear. Similarly, repeatable trace-based simulation requires unambiguous decisions about simulation inputs.

During each window, the Orchestrator (Figure 2, component Ⓒ) retrieves pre-processed telemetry from the Data Platform (via Ⓔ) and starts the simulator with a consistent view of the datacenter state. The Orchestrator also records metadata, such as when a simulation run started and which outputs belong together, which enables correctness and performance analysis of OpenDT itself.

OpenDT supports a configurable *acceleration factor*, expressed as a ratio between simulation and real-world time, with three main modes: (1) Simulation run one-to-one to real time (factor set to 1), (2) Fixed acceleration, e.g., factor set to 10, and (3) Maximum acceleration allowed by computational resources. The first mode is useful for live twinning. The accelerated modes enable faster simulation and exploration of long-term scenarios, but require prior knowledge of the full workload trace and datacenter configuration.

In the current approach, the orchestrator does not manage OpenDT's own resource allocation, nor the way its components are run. This design choice is deliberate: internal scheduling is delegated to the execution environment, allowing the orchestrator to focus exclusively on validating the digital-twinning loop itself. By keeping its functionality simple, the orchestrator ensures smooth integration among the simulation engine, data platform, and user-facing services while keeping operations interference-free.

2.4 Simulator (H) and Calibrator (G) interplay

OpenDT uses a Simulation Engine (Ⓗ) to inform operational decisions. Even after selecting a state-of-the-art simulator for this purpose, OpenDC [31], simulating large-scale and highly heterogeneous ICT infrastructure with high accuracy, precision, and explainability remains a critical yet non-trivial open scientific challenge in computer systems [3, 23, 29].

Like many state-of-the-art simulators in the field, OpenDC uses a deterministic and static simulation strategy. However, static simulation models can drift and introduce errors as the time horizon increases: hardware behavior varies with temperature, aging, and

firmware updates, while workload characteristics evolve over time. These dynamics threaten the validity of assumptions underlying the static model. To mitigate this problem, OpenDT adds a Self-Calibrator (G), which measures the difference between simulation-predicted results and actual telemetry, then continuously adjusts the simulation model to achieve the accuracy target **(NFR1)**.

The Self-Calibrator (SC) employs a grid search strategy over the parameter space of the power model. For each calibration cycle, the SC evaluates a fixed set of candidate parameter values, running short simulations with each configuration and comparing results against the most recent historical telemetry data. The configuration yielding the lowest Mean Absolute Percentage Error (MAPE) is selected and transmitted to the Simulation Engine (SE) for use in subsequent prediction windows.

Figure 3 illustrates the interplay between the SE and SC, which run as parallel processes. The calibrator completes its grid search over one time window (e.g., C0 in the figure) and sends the best-found configuration to calibrate the simulator for the next window (S1). This pipelined approach allows calibration to proceed without blocking simulation. We implement the SC as a separate service to enable independent scaling and inspection; we discuss the resulting synchronization challenges in Section 3.4.

In Section 4.4, we demonstrate experimentally that this calibration approach improves simulation accuracy both over uncalibrated OpenDT and over state-of-the-art tools [29, 30].

2.5 Interfaces (Component B): API and UI

OpenDT enables both monitoring and exploratory analysis on the target system. To this end, we design a simple API interface and an interactive visual component as the front-end (B) to display the state of the digital twin (V).

The API follows three key design principles: (1) *separation of concerns*, where simulation and data retrieval are compartmentalized and independent of each other; (2) *stability*, granted by only using a simple set of endpoints that are well-defined and documented; and (3) *observability*, achieved by showing the simulated and measured metrics in comparison. Table 1 summarizes the API's main endpoints for topology management and metric retrieval, including the detailed data formats for each.

API Design Choices. The topology endpoint uses the `PUT` HTTP method with a complete topology object in the request body. This design enables stateless topology updates: any client (whether a web dashboard, command-line tool, or external orchestration system) can submit a full datacenter configuration without the API server needing to maintain session state or track incremental changes. The topology schema supports hierarchical modeling with clusters containing hosts, where each host specifies CPU, memory, and power model parameters.

For metric retrieval, we provide separate `GET` endpoints for power consumption and carbon emissions rather than a single combined endpoint. This separation prevents unnecessary data transmission when a consumer requires only one metric type. For example, a real-time power monitoring dashboard can poll `/api/power` without receiving carbon intensity data it does not display. Both endpoints accept optional query parameters for sampling interval and start time, allowing clients to request data at the granularity appropriate for their use case, from fine-grained analysis (1-second intervals) to aggregated dashboard views (up to 1-hour intervals).

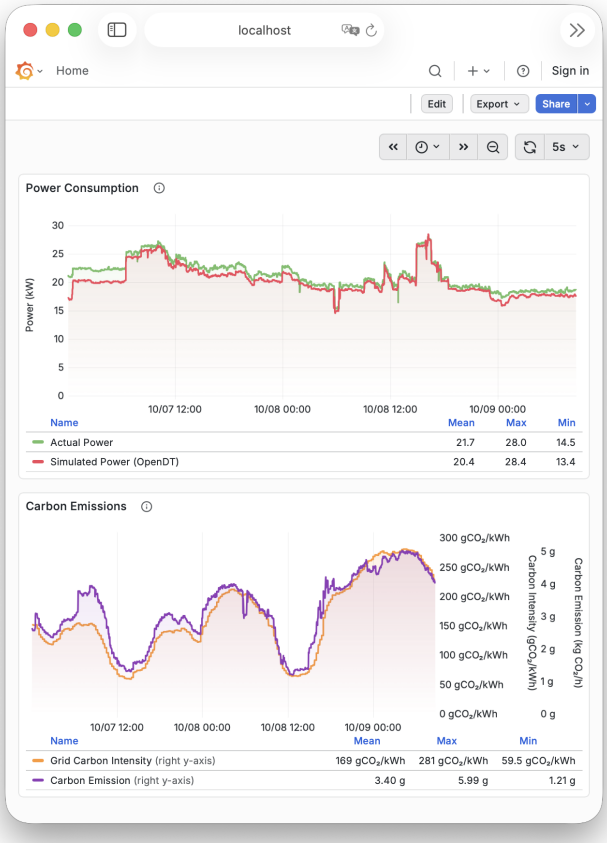

Figure 4: The OpenDT Grafana dashboard.

The API is implemented using FastAPI [33] with built-in documentation rendered via Swagger [35], providing interactive endpoint exploration and request testing for developers integrating with OpenDT.

User Interface. The main user interface is constructed with Grafana, a state-of-the-art visualization tool [14]. It queries the API to display simulated and measured signals side-by-side, providing a comparative view of system performance. Figure 4 shows the OpenDT dashboard, which presents two primary panels: (1) a power consumption panel displaying actual power draw alongside simulated predictions from OpenDT, enabling operators to assess simulation accuracy in real time; and (2) a carbon emissions panel correlating grid carbon intensity with the resulting emission rates. The human-in-the-loop can inspect system state, identify periods of simulation drift, or reconfigure the profile of the simulated ontology through the topology API. As new data arrives in real-time, performance, energy consumption, and effects of calibration can be monitored continuously.

2.6 Data Platform (Components D and E)

The *Data Platform* (Figure 2, components D, E) manages telemetry and digital twinning data throughout the operational cycle.

Table 1: OpenDT API endpoints for datacenter topology management and retrieving simulation metrics. The API uses JSON for all request and response bodies, with timestamps in ISO 8601 format.

Endpoint	Description	Request Body	Response Body	Design Rationale
`PUT /api/topology`	Update the simulated datacenter topology. Validates the structure and publishes it to Kafka for the simulator.	`Topology` object: `clusters[]` containing `hosts[]` with `cpu`, `memory`, `cpuPowerModel`	Confirmation with topology details, or validation error.	PUT with full topology state enables stateless updates from any interface (web UI, CLI, external systems) without requiring the server to track partial changes.
`GET /api/power`	Retrieve aligned power usage time series comparing simulated and actual consumption.	Query params: `interval_seconds` (1–3600) `start_time`	`PowerDataResponse`: `data[]` of {`timestamp`, `simulated_power`, `actual_power`}	Dedicated endpoint prevents transmitting unneeded carbon data; configurable interval supports dashboard polling.
`GET /api/co2_emission`	Retrieve carbon emission time series with grid intensity and power draw.	Query params: `interval_seconds` (1–3600) `start_time`	`CarbonDataResponse`: `data[]` of {`timestamp`, `carbon_intensity`, `power_draw`, `carbon_emission`}	Separate endpoint avoids payload overhead when consumers only need sustainability metrics.

Telemetry Ingestion. The Telemetry component (Ⓓ) continuously receives state-updates from the physical twin representing *observable telemetry*, often captured through multiple lenses; this contrasts to *unobservable telemetry*, which exists in reality but cannot be captured due to privacy or monitoring limitations. In Figure 2, the arrow connecting Ⓓ to Ⓔ represents data (pre)processing, either for further usage (e.g., simulation, calibration), or for storage and logging. The telemetry component converts raw monitoring data into simulator-ready formats and clips data outside the current window of operation.

Storage Format. The simulation results and telemetry data are persisted in Parquet format [36], a columnar storage format designed for analytical workloads. We selected Parquet for four reasons: (1) *scalability*, as columnar storage enables efficient querying over large time-series datasets; (2) *storage efficiency*, through built-in compression that reduces disk footprint; (3) *cross-system compatibility*, allowing data to be consumed by diverse tools including Python (Pandas, PyArrow), Spark, and database systems; and (4) *portability*, enabling straightforward data export for external analysis or archival. The API microservices (Section 2.5) query this Parquet-based storage to provide the human-in-the-loop (Ⓐ) access to simulated power, sustainability, and other relevant metrics through HTTP endpoints, maintaining a clean separation between the simulation engine and data access layers.

2.7 Human-in-the-Loop (Component A)

OpenDT adopts a human-in-the-loop (HITL) design, where OpenDT operates autonomously, yet a human can (optionally) intervene and dictate major decisions. Alternatively, the system could operate fully autonomously, isolated from external input, where only SLOs dictate. However, we argue that only the former is ethically viable, especially for ICT under critical workloads, where the system could prioritize meeting SLOs over human safety or well-being (e.g., prioritizing high-paid workload from industry instead of low-paid, life-dependent workload from hospitals) [19]. Although misprioritization is theoretically preventable through perfect systems and SLOs, distributed systems are far from perfect, and even small operational errors can cause critical disruptions [19].

HITL Capabilities. OpenDT supports three categories of human-in-the-loop interaction. First, *scenario configuration*: users can define telemetry data streams, select which metrics to monitor, and specify multi-metric analysis pipelines. Configuration can occur offline through JSON topology files before the twinning process starts, or at runtime through the REST API (Section 2.5), enabling dynamic adjustments without restarting the system. Second, *automated calibration with human oversight*: while the self-calibrator (Section 2.4) autonomously adjusts simulation parameters, the operator retains visibility into calibration decisions through the dashboard and can override parameters via topology updates if the automated adjustments produce undesirable results. Third, *multi-model simulation*: operators can configure OpenDT to run multiple heterogeneous power models in parallel [29], comparing their predictions to identify model-specific biases and select the most appropriate model for their infrastructure characteristics.

Operational Workflow. The human-in-the-loop interacts with OpenDT through a continuous feedback cycle involving the Grafana dashboard and the topology API. During normal operation, the operator monitors the dashboard (Figure 4), which displays both actual measurements from the physical datacenter and simulated predictions from OpenDT side-by-side. This comparative view enables the operator to assess simulation accuracy in real time: when the simulated power curve closely tracks the actual power consumption, the digital twin is accurately representing the physical infrastructure; when drift occurs, the operator can identify whether intervention is needed.

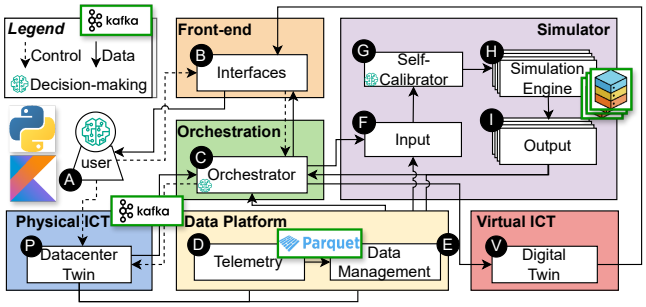

Figure 5: A high-level overview of the software stack in the current OpenDT engineered prototype.

When the operator wishes to explore configuration changes or correct simulation parameters, they submit an updated topology through the API (Section 2.5). The updated topology, which may include modified host counts, adjusted CPU power model parameters, or reconfigured cluster structures, is validated and published to the message queue. The simulator picks up the new topology for subsequent simulation windows, and the resulting predictions are written to the data platform. The dashboard, polling the API at regular intervals, then reflects these updated predictions. This closed-loop interaction allows the operator to iteratively refine the digital twin: submit a topology adjustment, observe its effect on predicted metrics, and adjust further if necessary.

Decision Support. Beyond monitoring, the HITL design supports proactive decision-making. For example, an operator planning a hardware upgrade can first model the change in OpenDT by updating the topology with the proposed new host specifications. The simulator then predicts the power consumption and carbon emissions under the current workload with the hypothetical hardware. By comparing these predictions against the baseline, the operator gains quantitative insight into the expected impact before committing resources to the physical change. Similarly, operators can use OpenDT to evaluate scheduling policy changes, capacity planning scenarios, or the sustainability implications of different power sourcing strategies, all while maintaining oversight and final approval authority over any changes propagated to the physical infrastructure.

3 Implementation

Following the system design, we will now describe the implementation of OpenDT, which provides the concrete system on which the experiments in Section 4 are carried out. Instead of documenting every component individually, we present the main challenges and solutions encountered in the engineering process.

Figure 5 illustrates OpenDT's software stack. Overall, OpenDT is written in Python and is deeply embedded with OpenDC, written in Kotlin. The implementation is built as a Docker Compose microservice system in which the core components (data source, simulator, calibrator, and API) communicate through Apache Kafka [22]. All services mount a shared host directory that serves as a file-based workspace for configuration files and simulation outputs. This shared location makes runtime data easy to inspect, and helps with reasoning about the persisted state of the system.

3.1 Retrofitting OpenDC for Online Digital Twinning

The OpenDC simulator is originally designed as an offline simulator that processes an entire workload in a single execution. In OpenDT, however, workload data arrives continuously, and the simulator must generate updated predictions at fixed intervals. Supporting this online, incremental setting required a redesign of how workloads are aggregated and supplied to OpenDC.

OpenDC models each task as a quantity of compute cycles, which is then subdivided into fragments that describe its expected CPU usage over time. In an online setting, tasks may span multiple prediction windows or originate much earlier while still influencing current CPU utilization and power behaviour. For this reason, OpenDT invokes OpenDC on an accumulated set of all tasks received up to the current simulated time, ensuring that any task that could potentially affect the present window is included.

The drawback of this approach is a growing context horizon, since each OpenDC invocation receives all previously observed tasks. A more efficient implementation would incorporate task completion information from earlier simulation outputs so that future invocations can exclude workloads that have already finished. This remains future work.

3.2 High Throughput Workload Ingestion via Kafka

To support fast experimentation **(NFR2)**, we added a feature to replay the workload traces at a configurable speed factor. This pragmatic choice reduced iteration time but also surfaced limitations in the initial ingestion design. In the first version, tasks and their corresponding fragments were published as separate Kafka messages on two topics. Although this mirrored the structure of the raw trace dataset, it required two producers to stay aligned with the simulated timeline. Since workload traces contain far more fragments than tasks, Kafka struggled to keep up at higher replay speeds. Fragment messages accumulated, arrived late at consumers, and required additional logic to associate them with previously received tasks.

To eliminate this coordination problem, we redesigned ingestion around a single nested task object. The workload generator performs a one-time join between each task and its fragments and sends a single Kafka message containing the complete structure. This removes the fragment topic entirely and reduces the message volume to the number of tasks. It also allows ingestion to proceed through a single producer thread, which avoids the timing issues present in the original design and produces a more predictable ingestion pipeline.

3.3 Progress Markers: Deliverability & Window Completeness

The simulator processes workload traces at a fixed simulation time frequency. Implementing this reliably required a mechanism to determine when all submissions belonging to a time window had

arrived. Without such a guarantee, the simulator might either miss late-arriving tasks or inconsistently adjust window boundaries, which either impacts reproducibility or complicates downstream analysis.

To address this, the workload producer sends periodic progress markers in addition to task submissions. These markers appear at fixed simulated intervals, for example, once per minute. Since Kafka preserves message order within a partition, the arrival of a marker for timestamp t indicates that all submissions with earlier timestamps have been sent. The simulator uses these markers to trigger a simulation step only after the corresponding window is complete. This approach removes the dependence on runtime timing and ensures consistent formation of simulation windows.

3.4 Calibration and Temporal Synchronization

A second major design decision involved the calibration mechanism. We considered integrating the calibrator directly into the simulator, which would have simplified data sharing and ensured a single ordered execution flow. However, this would have tightly coupled the components, prevented independent resource scaling, and made inspection of each component more difficult.

For this reason, we implemented the calibrator as a separate service that consumes the same workload topics and publishes proposed topology updates. This decoupling introduced the challenge of temporal alignment. To address this, we implemented a best-effort synchronization mechanism. Each service tracks its own perceived simulated time; if it runs ahead, it delays execution until the next window boundary, and if it falls behind, it logs warnings because there is no automatic way to accelerate processing. When this happens, the system operator is expected to adjust the speed factor accordingly.

The decoupled and asynchronous behaviour is illustrated in Figure 3. The figure shows how the simulator and calibrator process the workload stream independently, each advancing through its own sequence of workload tasks, with the calibrator sending updated topology information back to the simulator.

Although approximate, this approach keeps the simulator and calibrator sufficiently aligned to produce consistent calibration results and stable system behaviour in practice. With these mechanisms in place, the prototype offers a reproducible execution environment that supports the trace-based experiments presented in the following section.

4 Trace-based experiments, enabled by and conducted with OpenDT

In this section, we validate OpenDT using real-world traces, compare with a peer-reviewed simulator, and explore if, and by how much, real-time simulation recalibration improves accuracy. Overall, our experiments support *main findings (**MF1-3**)*:

(**MF1**) OpenDT allows for continuous replay of real-world datacenter operation, with high accuracy. Compared to the measured MAPE error rate of the peer-reviewed simulator FootPrinter 7.86% [30], for this experiment OpenDT achieves a MAPE of 5.13% (§4.3).
(**MF2**) OpenDT's live self-calibration is enabled by digital twinning, happens regularly and continuously, and improves MAPE; here, from 5.13% to 4.39% (§4.4).
(**MF3**) OpenDT can twin multiple core metrics across operational layers, related to sustainability (e.g., power draw, efficiency, §4.3) and performance (e.g., TFLOPs, §4.3, and CPU utilization, §4.4).
(**MF4**) Supporting C-level decisions and datacenter operators, OpenDT can show underutilised compute infrastructure, yet consuming large amounts of electricity while being idle. One such example is illustrated in Section 4.3 and Figure 8, where, by reproducing a real-world scenario from SURF-22 in silico, with OpenDT capabilities, we show that only 25% of the available and powered-on compute power is utilised.

4.1 Prototype implementation and performance

We implement OpenDT as a Docker Compose microservice system in which the core components (data source, simulator, calibrator, and API) communicate through Apache Kafka [22].

All services mount a shared host directory that serves as a file-based workspace for configuration files and simulation outputs. This shared location makes runtime data easy to inspect and helps with reasoning about the persisted state of the system.

For implementation details, see Section 4.1.

Performance: Enabled by the lightweight nature of the OpenDT prototype, we run both experiments on a common off-the-shelf MacBook Pro, with an M1 Max 10-core CPU and 32 GB of RAM. This showcases OpenDT's capabilities of twinning 7 days of datacenter operation within 46 minutes, thus successfully addressing **(NFR2)**.

4.2 Experiment setup

Infrastructure: In these experiments, we twin the SURF-SARA production cluster at SURF, the Dutch infrastructure for scientific computing. SURF-SARA contains 277 hosts, each with 128 GB of RAM and 16 processing cores running at maximum 2.1 GHz.

Workload trace (public): We use SURF-22, a scientific workload trace from SURF, also used in peer-reviewed articled [30, 37].It traces scientific jobs with an average duration of 39.52 CPU-hours [29].

Quantifying accuracy / error rate: We employ Mean Percentage Average Error (MAPE), a widely used and relative-error metric [26, 29, 30, 32]. MAPE is calculated as MAPE $[\%] = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{R_i - S_i}{R_i}\right| \times 100$, where n is the number of samples, R is the real-world data [37], S is the simulation data, and i is the sample index.

Power model: We model CPU power draw adopting the OpenDC [12, 23] analytical formula: $P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}})\,(2u - u^r)$. Here, u is CPU utilization, P_{idle} and P_{max} represent idle and max power, and r is the *calibration parameter* (see Section 2.4).

4.3 Experiment E1: Peer-reviewed experiment reproduced and extended with OpenDT

Niewenhuis et al. propose FootPrinter [30], a tool for predicting the CO_2 footprint of datacentres. Figure 6 illustrates the workflow we followed in this work, leveraging OpenDT capabilities to reproduce the key experiment [30]. Illustrative for the difference between approaches, whereas FootPrinter (Ⓓ) runs once a hand-tuned energy model [29, 30], OpenDT (Ⓒ) continuously predicts energy consumption at the industry-standard sampling granularity (i.e., 5-minute rate), using a generic predictive model that avoids overfitting for a specific trace. To reproduce the experiment, we use

Table 2: Experiment configurations. SUO = system under observation, WT = workload trace, PM = performance metrics, SM = sustainability metrics, OP = operational phenomena (failures), RSR = real-time simulation recalibration. Time = workload execution time, CPU = average CPU utilization [%],

		Input		Output			
Section	Focus	SUO	WT	PM	SM	OP	RSR
§4.3	Reproduce peer-reviewed experiment with twinning capabilities	S1	WT1	time, CPU	Wh, gCO2	✘	✘
§4.4	Evaluate simulation calibration against current state-of-the-art	S1	WT1	✘	Wh	✘	✔

Table 3: Workload traces used in experiments. Name is source and collection year (e.g., SURF-22 = source SURF, year 2022). SR = sampling rate, CH = CPU hours (millions), S = scientific.

ID	Name	Type	Duration	Jobs	CH	SR
WT1	SURF-22	S	7 days	7,850	0.31	30 s

Table 4: Systems Under Observation (SUO). Scale = number of hosts.

ID	Source	Scale	Resources per host
S1	SURF	277	128 GB RAM, 16 cores, 2.1 Ghz

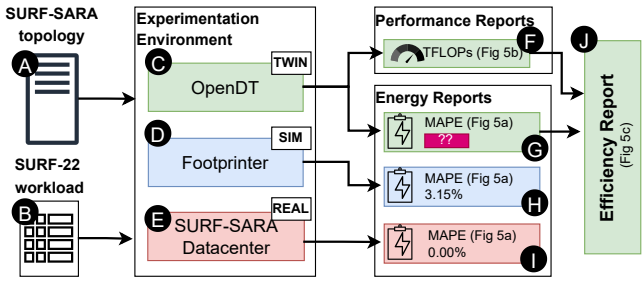

Figure 6: Section 4.3 experiment, adapted and re-run from FootPrinter [30] and extended with OpenDT.

datacenter topology (Ⓐ) and workload trace (Ⓑ) recorded from SURF [29, 30] (Ⓔ), which we regard as "*ground-truth*" (Ⓘ). We measure the MAPE of both OpenDT (Ⓖ), compare it with FootPrinter's (Ⓗ), and also record datacenter performance data (Ⓘ) that enables us to extend the experiment and also report energy-efficiency results (Ⓙ). The results of this experiment firmly support main findings MF1 and MF3.

Accuracy validation by reproducing the peer-reviewed experiment: We determine the accuracy of predictions by determining the MAPE error rate, using the formula described in Section 4.2; the lower the MAPE, the more accurate the prediction. Figure 7A depicts the results of the reproducibility experiment. Between predictions and ground truth (measured reality, we compute FootPrinter's MAPE as 7.86% and OpenDT's MAPE as 5.13% (2.73% better). Here, even

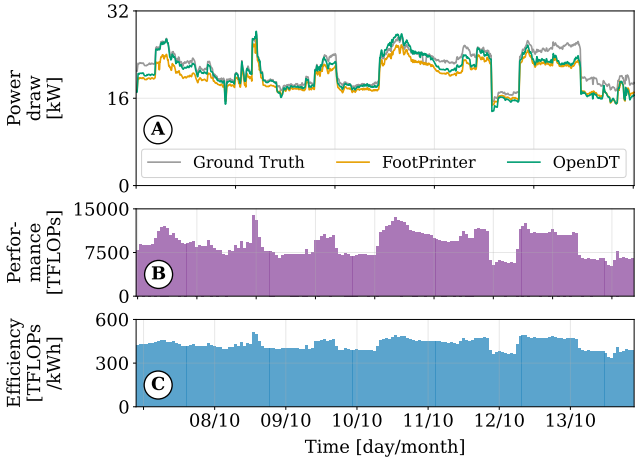

Figure 7: Operational metrics for the compute cluster over time: (A) Power draw simulation results vs. measured reality (reproduced from [29, 30]). (B) Performance measured in TFLOPs. (C) Efficiency measured in TFLOPs/kWh.

without simulation recalibration, OpenDT meets the accuracy requirement **(NFR1)**.

Extending the peer-reviewed experiment: Beyond quantifying datacenter sustainability, OpenDT can also predict performance and efficiency quickly. Figure 7 depicts the results obtained when we extend the FootPrinter experiment with performance results and demonstrates that OpenDT meets **(NFR3)**. In Figure 7B, we illustrate live, continuous predictions of OpenDT on datacenter performance, which are further processed to produce the efficiency evaluation depicted in Figure 7C. Overall, discretising OpenDT predictions per hour, we identify the highest efficiency when the datacenter performance, quantified in floating-point operations, is the highest. Further investigating OpenDT's predictions, we identify underutilization of the available infrastructure: the monitoring period, under 30% of the available processing power is used, while the remaining are idle. Such insights, enabled by OpenDT, could help operators better monitor and plan available infrastructure.

Figure 8 showcases OpenDT's capabilities of predicting two performance metrics **(NFR3)**, in real-time. We digital twin, with OpenDT capabilities, S1 datacenter under SURF-22 workload, and trace the average CPU utilization and latency (i.e., the estimated

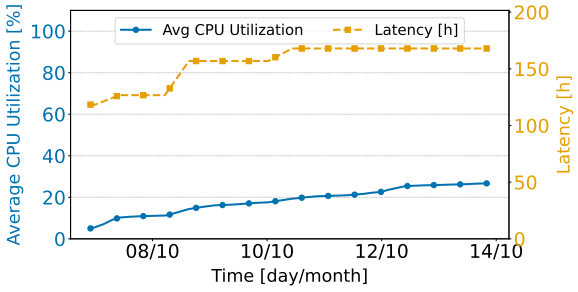

Figure 8: Performance metrics of S1 under SURF-22 workload, over time. Through digital twinning, we capture how variations in the workload can impact average CPU utilization and total runtime (i.e., how much time it takes for the datacenter to complete the workload, assuming no changes).

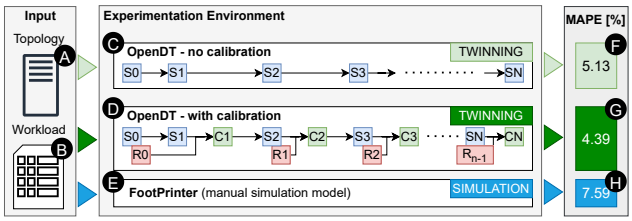

Figure 9: Overview of experiment 2. Evaluating how well OpenDT predicts, with and without real-time simulation re-calibration, in comparison with the peer-reviewed FootPrinter [30]. S = simulation, C = calibration, R = real-world measurements (required for dynamically, on-the-fly, recalibrating predictions).

remaining time, until the workload is complete, assuming no change in the system state).

Simulating latency: We observe a slight increase in latency over time, from ≈60 hours to ≈90 hours until the workload would be finalized. OpenDT's continuous twinning capabilities showcases two main timestamps when the latency increases, potentially due to additional tasks the datacenter needs to execute. Such analysis and OpenDT-aided historical replay can help practitioners **(UC1)** take informed decisions on datacenter management (e.g., up/down-scaling), or could pinpoint to scientists **(UC2)** hot operational intervals, useful for exploration.

CPU utilization: Figure 8 emphasises a critical, yet non-trivial challenge in ICT keeping resources, in this case CPUs, idle [6, 20, 23]; an idle CPU can consume up to 40% of the energy it would consume at maximum utilization [6, 23]. Through OpenDT, we identify, in Figure 8, an under-utilization of the available infrastructure, where only about 25% of the available processing power is used, while the remaining three quarters are consuming power for being idle. Such insights, enabled by OpenDT, could aid operators in better monitoring and planning available infrastructure **(UC1)**.

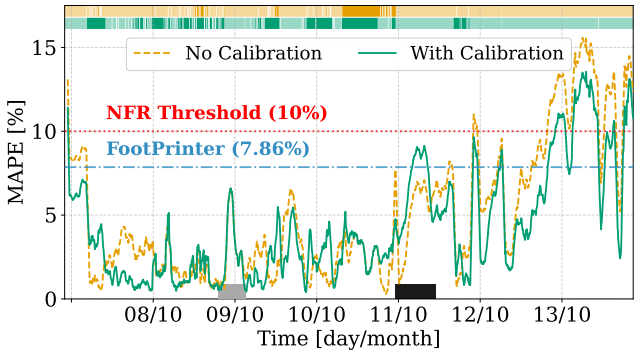

Figure 10: Evolution of error rate in power-draw estimation of OpenDT with calibration (MAPE 4.39%), without calibration (MAPE 5.13%), compared with NFR1 and FootPrinter. The horizontal bars at the top distinguish OpenDT overestimations (solid color) from under (dimmed), without calibration (orange, top bar) and with calibration (green). The black and grey marks at the bottom mark special intervals, see text.

4.4 E2: Evaluating live, self-recalibration against the current simulation state-of-the-art

Addressing the challenge of improving prediction accuracy for digital twins (see Section 2.4), we evaluate the Self-Calibrator. Experimentally, we analyze how real-time recalibration affects OpenDT's accuracy relative to traditional simulation, which is not live-calibrated.

Figure 10 depicts the MAPE over time for OpenDT with and without calibration, against the set threshold, i.e., below 10%, 90% of the time **(NFR1)**. Overall, the live-recalibration approach reduces MAPE by 0.74, from 5.13% to 4.39%. We identify for the uncalibrated OpenDT the MAPE is < 10% occurs only 86% of the time. However, the live re-calibrated OpenDT achieves MAPE < 10% just over 92% of the time, thus successfully meeting **(NFR1)**.

Analysis of calibration vs. no calibration: Although the calibration technique proposed in this work is relatively simple, the calibrated error rate of OpenDT is better than that of the traditional approach. It would be tempting to conclude that using calibration is always beneficial. However, we set out to investigate whether this is the case, with results depicted by Figure 10. We observe that there exist significant simulation-periods when the no-calibration technique performs better than the OpenDT calibration (e.g., a window of about 12 hours centered in 11/10, highlighted in black in Figure 10) or equally good (e.g., around 9/10, highlighted in grey). This suggests calibration is important but requires further attention.

Prediction behaviour: A large body of work, including SPEC RG Cloud's work on auto-scaling metrics [16], addresses the impact of under- and over-estimation in infrastructure provisioning. Underestimating and under-provisioning ICT infrastructure could lead to performance concerns, system faults, and, ultimately, system failures. In contrast, overprovisioning could lead to sustainability concerns, both environmental and financial, where energy (and thus, money) is wasted by idle infrastructure. In Figure 10, marked

by the horizontal bars at the top of the figure, we identify an underestimation trend in OpenDT's predictive model. This occurs both without calibration (in 85% of discrete-event predictions) and with calibration (66%). Overall, *we find calibration alleviates underestimation bias, reducing the frequency of underestimated predictions.*

5 Discussion

In this section, we evaluate OpenDT towards the established functional and non-functional requirements (Section 5.1. Then, in Section 5.2 we present limitations of our experiments, simulation, and real-world tracing (telemetry).

5.1 Requirement analysis

In Section 2, we establish a set of *requirements*, which guided the design and engineering of OpenDT. We now evaluate if, and how well, OpenDT matches the established functional (FR) and non-functional requirements (NFR).

(FR1) Digital-twinning ICT: OpenDT ensures active and continuous replication of real-world ICT infrastructure through a *two-stage* digital twin, which *(1) ingests datacenter telemetry* at a user-established granularity, then *(2) simulates* how the infrastructure would operate in the future (e.g., sustainability, performance), assuming no state changes. We plan future work towards a closed-loop digital twinning process, which would add a third stage where, (3) OpenDT would adjust a real-world datacenter, automatically and aligned with established SLOs, yet supervised by a human-in-the-loop, based on simulations from step (2).

(FR2) State-of-the-art, discrete-event simulation: OpenDT uses OpenDC, a peer-reviewed, discrete-event simulator, with over 7 years of deployment and operation, employed in scientific publications [23, 24, 29], and in national-scale projects (e.g., the Dutch 6G Future Network Services [3]). OpenDC is able to predict performance, sustainability, and availability of datacenters under workload, at a user-established granularity.

(FR3) Simulator real-time re-calibration: OpenDT support real-time simulation re-calibration, as described in Section 2.4, where we detail the interplay between the simulator and the live-recalibration. We quantify in Section 4.4 the accuracy of the live-calibrated OpenDT simulation, as compared with the traditional, state-of-the-art, non-calibrated simulation from FootPrinter [30]; overall, calibration helps reduce MAPE from 7.86% to 5.13% (thus, 2.73% better).

Through experiments conducted in Section 4, we explore real-world scenarios and measure OpenDT against the non-functional requirements established in Section 2.

(NFR1) Accurate, ground-truth adjusted predictions: As shown in Section 4.3, and supported by (**MF1**), OpenDT allows for continuous replay of real-world datacenter operation, with high accuracy. Compared to the measured MAPE error rate of the peer-reviewed simulator FootPrinter 7.86% [30], for this experiment OpenDT achieves a MAPE of 5.13% (§4.3). We envision future work in evaluating how various live-calibration techniques could improve simulation accuracy, aiming for improving the already low error rate even further.

(NFR2) Performant, lightweight digital twinning: We validate OpenDT against **(NFR2)** through the nature of the experimental setup, detailed in Section 4.2. Overall, we conduct all experiments on a common off-the-shelf MacBook Pro, with an M1 Max 10-core CPU and 32 GB of RAM. Further showcasing performance capabilities, we measured the runtime of experiment E1 (Section 4.3), where we twin 7 days of datacenter operation, to 46 minutes.

(NFR3) Multi-layer metrics: As presented in Section 4, and supported by (**MF4**), OpenDT can twin multiple core metrics across operational layers, related to sustainability (e.g., power draw, efficiency, §4.3) and performance (e.g., TFLOPs, §4.3, and CPU utilization, §4.4).

5.2 Limitations

We now analyze limitations of the experiments conducted in Section 4, limitations of OpenDT, and limitations of digital twinning, overall, as a technique for monitoring and operating datacenters.

Evaluating potential threats to *experimental validity*, we identify *generality* and *isolation*. Addressing *(i) the generality aspect*, we acknowledge that the experiments conducted in this work analyze a single workload trace, specifically SURF-22, traced in the Dutch Supercomputer SURF, and reproduce a single peer-reviewed experiment. Thus, the system may or may not generalise to other datacenter topologies or workload types (e.g., business-specific), thus hindering external validity. Addressing the *isolation aspect*, the experiments conducted in this work used simulation or twinning methods which assumed perfect isolation from operational phenomena, such as hardware failures; however, we argue that every real-world system, including the SURF-SARA cluster, is prone to operational phenomena, which we do not capture in this work. We plan to conduct further experimental work of OpenDT, and futher expand our set of experiments with more real-world traces, of various kinds (e.g., business-critical vs. scientific-specific), and with infrastructure facing operational phenomena – all features already supported by OpenDC [23], or by related tools using OpenDC [6, 24, 29].

Evaluating potential threats to method validity, we identify several limitations where digital twinning could potentially be inferior to traditional simulation. At the core of any digital twin, including OpenDT, is the simulator; thus, the validity of the twin's predictions are directly dependent on the validity of the simulator's predictions. For OpenDT, the validity of OpenDT's predictions could, thus, be hindered by a potential threat to the correctness of the simulator's predictions. To address this limitation, we select for OpenDT the peer-reviewed and community-vetted OpenDC, currently trusted by national-wide infrastructure projects [3] and by European-scale projects [4]. Still, we design and engineer OpenDT as modular, making it simple to integrate different discrete-event simulators.

6 Related work: digital twinning and simulation

We now contrast OpenDT with prior work in digital twinning across fields of science and in datacenter simulation.

Digital twinning: Digital twins are already widely used in large-scale sciences such as aviation and space exploration [8], and enable coarse-grained, dynamic adjustments for the physical twin from a distance, without requiring physical access (e.g., Apollo 13 controlling from the Earth [5]). Similarly, small-scale sciences (e.g., biomedical applications) use digital twins to enable what-if analysis and patient monitoring and treatment, where physical access is

not viable (e.g., monitoring ventricular activity) using a closed-loop with a doctor-in-the-loop [34]. In contrast, for the medium-scale science of Computer Systems, where both detailed models exist for some objects and others are treated statistically, digital twins are only starting to emerge, mainly due to the inherent intellectual and computational complexity of combining high-level abstractions with detailed system monitoring and massive amounts of telemetry; OpenDT is to-date the first such digital twin to publicly detail its design and become open source.

Simulation: In this work, we leverage at the core of the digital twin the peer-reviewed OpenDC, a discrete-event simulator able to predict datacenter performance, sustainability, and availability, in time and cost-efficent way [23, 24]. OpenDC is able to simulate with high explainability, through multiple aligned simulation models [29], and with high robustness, through Meta-Models, which predict by combining predictions of other models, thus alleviating individual model biases [29]. Other seminal datacenter simulators, such as DCSim [15], CloudSim [9, 13, 17], or SimGrid [10]. In contrast, OpenDT enables a new mode of operation, twinning live and closing the simulation loop, for which it proposes a novel design, implementation, and evaluation.

7 Open and Reproducible science

We release the engineered OpenDT prototype as FOSS, together with a FAIR dataset of traces, via GitHub[1], accessible by all groups of stakeholders, thus aiding **(UC1)**, **(UC2)**, and **(UC3)**. Adhering to state-of-the-art principles on reproducible science, the OpenDT artifact contains a reproducibility capsule.

Furthermore, addressing the main stakeholders, we ensure robust and clear documentation for OpenDT, through GitHub, live API documentation, and through a static-website[2], containing a demo, a tutorial **(UC3)** presented through a series of pre-recorded videos.

8 Conclusion and future work

Understanding the performance and sustainability of operating and upcoming datacenters is essential to our society and economy. Addressing the lack of an open-source digital twin for datacenters, in this work we have designed, implemented, and experimented with OpenDT. Overall, our experimental results suggest that OpenDT supports continuous replay of real-world datacenter operations, with high accuracy and, through self-calibration, can improve this accuracy over state-of-the-art approaches.

We have released OpenDT as open-source and plan to conduct extensive future experimentation and coupled with real-world infrastructure as closed-loop, as part of a major infrastructure project with over 75 partner institutions [3]. We plan to expand OpenDT to domain specific operation and couple closed-loop with large-scale ICT infrastructure, running LLM inference workloads. Lastly, we plan to build educational material around digital twinning, aided by OpenDT, and release as open-education, and as optional material in computer systems courses on Computer Organization, Distributed Systems, or in a future edition of the course on Modern Distributed Systems MOOC from edX [1], which already uses a form of OpenDC and could therefore use an exercise based on OpenDT.

[1] https://github.com/atlarge-research/opendt/tree/hcp
[2] https://atlarge-research.github.io/opendt/

References

[1] Delft University of Technology (DelftX) 2024. *Modern Distributed Systems.* Delft University of Technology (DelftX). Online course page.

[2] 2023. Overheating datacenter stopped 2.5 million bank transactions. Laura Dobberstein, The Register, https://www.theregister.com/2023/11/07/overheating_datacenter_singapore/.

[3] 2025. Future Network Services: 6G for and by the Netherlands. https://futurenetworkservices.nl/en/.

[4] 2025. Graph-Massivizer: Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe. https://graph-massivizer.eu/.

[5] B Danette Allen. 2021. Digital twins and living models at NASA. In *Digital Twin Summit.*

[6] Georgios Andreadis, Fabian Mastenbroek, Vincent van Beek, and Alexandru Iosup. 2022. Capelin: Data-Driven Compute Capacity Procurement for Cloud Datacenters Using Portfolios of Scenarios. *IEEE Trans. Parallel Distributed Syst.* 33, 1 (2022), 26–39. doi:10.1109/TPDS.2021.3084816

[7] Georgios Andreadis, Laurens Versluis, Fabian Mastenbroek, and Alexandru Iosup. 2018. 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.* IEEE / ACM, 37:1–37:15. http://dl.acm.org/citation.cfm?id=3291706

[8] Hakan Aydemir, Umut Durak, Sven Hartmann, and Ugur Zengin. 2020. The Digital Twin Paradigm for Aircraft – Review and Outlook. In *AIAA SciTech 2020 Forum.* AIAA. doi:10.2514/6.2020-0553

[9] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* 41, 1 (2011), 23–50. doi:10.1002/SPE.995

[10] Henri Casanova. 2001. Simgrid: A Toolkit for the Simulation of Application Scheduling. In *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18, 2001, Brisbane, Australia.* IEEE Computer Society, 430–441. doi:10.1109/CCGRID.2001.923223

[11] Nuria de Lama Sanchez, Peter Haase, Dumitru Roman, and Radu Prodan. 2023. Boosting the Impact of Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe Graph-Massivizer: A Horizon Europe Project. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, ICPE 2023, Coimbra, Portugal, April 15-19, 2023,* Marco Vieira, Valeria Cardellini, Antinisca Di Marco, and Petr Tuma (Eds.). ACM, 233–238. doi:10.1145/3578245.3585334

[12] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz André Barroso. 2007. Power provisioning for a warehouse-sized computer. In *34th International Symposium on Computer Architecture (ISCA 2007), June 9-13, 2007, San Diego, California, USA*, Dean M. Tullsen and Brad Calder (Eds.). ACM, 13–23. doi:10.1145/1250662.1250665

[13] Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Pedro R. M. Inácio, and Mário M. Freire. 2017. CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, May 8-12, 2017.* IEEE, 400–406. doi:10.23919/INM.2017.7987304

[14] Grafana Labs. 2024. Grafana: The Open Observability Platform. https://grafana.com/ Accessed: 2026.

[15] Sandeep K. S. Gupta, Rose Robin Gilbert, Ayan Banerjee, Zahra Abbasi, Tridib Mukherjee, and Georgios Varsamopoulos. 2011. GDCSim: A tool for analyzing Green Data Center design and resource management techniques. In *2011 International Green Computing Conference and Workshops, IGCC 2012, Orlando, FL, USA, July 25-28, 2011.* IEEE Computer Society, 1–8. doi:10.1109/IGCC.2011.6008612

[16] Nikolas Herbst, André Bauer, Samuel Kounev, Giorgos Oikonomou, Erwin Van Eyk, George Kousiouris, Athanasia Evangelinou, Rouven Krebs, Tim Brecht, Cristina L. Abad, and Alexandru Iosup. 2018. Quantifying Cloud Performance and Dependability: Taxonomy, Metric Design, and Emerging Challenges. *ACM Trans. Model. Perform. Evaluation Comput. Syst.* 3, 4 (2018), 19:1–19:36. doi:10.1145/3236332

[17] Tharindu B. Hewage, Shashikant Ilager, Maria Alejandra Rodriguez, and Rajkumar Buyya. 2024. CloudSim express: A novel framework for rapid low code simulation of cloud computing environments. *Softw. Pract. Exp.* 54, 3 (2024), 483–500. doi:10.1002/SPE.3290

[18] IDC. 2024. AI Datacenter Capacity, Energy Consumption, and Carbon Emission Projections. https://www.idc.com/getdoc.jsp?containerId=US52131624.

[19] Alexandru Iosup. 2024. A VU on Digital Twins to Improve the Performance and Technological Sustainability of Datacenters in the Continuum. In *Proceedings of MODSIM 2024.* Seattle, USA. August 14–16, 2024.

[20] Alexandru Iosup, Fernando Kuipers, Ana Lucia Varbanescu, Paola Grosso, Animesh Trivedi, Jan S. Rellermeyer, Lin Wang, Alexandru Uta, and Francesco Regazzoni. 2022. Future Computer Systems and Networking Research in the Netherlands: A Manifesto. *CoRR* abs/2206.03259 (2022). arXiv:2206.03259 doi:10.48550/ARXIV.2206.03259

[21] Alexandru Iosup, Radu Prodan, Ana Lucia Varbanescu, Sacheendra Talluri, Gilles Magalhaes, Kailhan Hokstam, Hugo Zwaan, Vincent van Beek, Reza Farahani, and Dragi Kimovski. 2023. Graph Greenifier: Towards Sustainable and Energy-Aware Massive Graph Processing in the Computing Continuum. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, ICPE 2023, Coimbra, Portugal, April 15-19, 2023*, Marco Vieira, Valeria Cardellini, Antinisca Di Marco, and Petr Tuma (Eds.). ACM, 209–214. doi:10.1145/3578245.3585329

[22] Jay Kreps. 2011. Kafka : a Distributed Messaging System for Log Processing. https://api.semanticscholar.org/CorpusID:18534081

[23] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin Van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. 2021. OpenDC 2.0: Convenient Modeling and Simulation of Emerging Technologies in Cloud Datacenters. In *21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2021, Melbourne, Australia, May 10-13, 2021*, Laurent Lefèvre, Stacy Patterson, Young Choon Lee, Haiying Shen, Shashikant Ilager, Mohammad Goudarzi, Adel Nadjaran Toosi, and Rajkumar Buyya (Eds.). IEEE, 455–464. doi:10.1109/CCGRID51090.2021.00055

[24] Fabian Mastenbroek, Tiziano De Matteis, Vincent van Beek, and Alexandru Iosup. 2025. RADiCe: A Risk Analysis Framework for Data Centers. *Future Gener. Comput. Syst.* 166 (2025), 107702. doi:10.1016/J.FUTURE.2024.107702

[25] Martin Molan, Junaid Ahmed Khan, Andrea Bartolini, Roberta Turra, Giorgio Pedrazzi, Michael Cochez, Alexandru Iosup, Dumitru Roman, Joze M. Rozanec, Ana Lucia Varbanescu, and Radu Prodan. 2023. The Graph-Massivizer Approach Toward a European Sustainable Data Center Digital Twin. In *47th IEEE Annual Computers, Software, and Applications Conference, COMPSAC 2023, Torino, Italy, June 26-30, 2023*, Hossain Shahriar, Yuuichi Teranishi, Alfredo Cuzzocrea, Moushumi Sharmin, Dave Towey, A. K. M. Jahangir Alam Majumder, Hiroki Kashiwazaki, Ji-Jiang Yang, Michiharu Takemoto, Nazmus Sakib, Ryohei Banno, and Sheikh Iqbal Ahamed (Eds.). IEEE, 1459–1464. doi:10.1109/COMPSAC57700.2023.00224

[26] Juan José Montaño Moreno et al. 2013. Using the R-MAPE index as a resistant measure of forecast accuracy. *Psicothema* (2013).

[27] Sebastian Moss. 2023. Surgeries and procedures paused at Wichita hospitals due to data center outage. Data Center Dynamics, https://www.datacenterdynamics.com/en/news/surgeries-and-procedures-paused-at-wichita-hospitals-due-to-data-center-outage/.

[28] NeilMcAllister. 2013. Google goes dark for 2 minutes, kills 40% of world's net traffic. https://www.theregister.com/2013/08/17/google_outage/.

[29] Radu Nicolae, Dante Niewenhuis, Sacheendra Talluri, and Alexandru Iosup. [n. d.]. M3SA: Exploring Datacenter Performance and Climate-Impact with Multi-and Meta-Model Simulation and Analysis. *Available at SSRN 5377101* ([n. d.]).

[30] Dante Niewenhuis, Sacheendra Talluri, Alexandru Iosup, and Tiziano De Matteis. 2024. FootPrinter: Quantifying Data Center Carbon Footprint. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering, ICPE 2024, London, United Kingdom, May 7-11, 2024*, Simonetta Balsamo, William J. Knottenbelt, Cristina L. Abad, and Weiyi Shang (Eds.). ACM, 189–195. doi:10.1145/3629527.3651419

[31] OpenDC Team. [n. d.]. Input: Workload — OpenDC Documentation. https://atlarge-research.github.io/opendc/docs/documentation/Input/Workload. Accessed 2025.

[32] Oracle. 2024. MAPE (Mean Absolute Percentage Error) Documentation. Oracle Cloud Infrastructure Documentation, https://docs.oracle.com/en/cloud/saas/planning-budgeting-cloud/pfusu/insights_metrics_MAPE.html. Accessed: 30 April 2025.

[33] Sebastián Ramírez. 2018. *FastAPI.* https://github.com/fastapi/fastapi

[34] Kaan Sel, Deen Osman, Fatemeh Zare, Sina Masoumi Shahrbabak, Laura Brattain, Jin-Oh Hahn, Omer T Inan, Ramakrishna Mukkamala, Jeffrey Palmer, David Paydarfar, et al. 2024. Building digital twins for cardiovascular health: From principles to clinical impact. *Journal of the American Heart Association* 13, 19 (2024), e031981.

[35] SmartBear Software. 2024. *Swagger UI.* https://swagger.io/tools/swagger-ui/ Accessed: 2026.

[36] The Apache Software Foundation. 2026. Apache Parquet. https://parquet.apache.org/.

[37] Laurens Versluis, Mehmet Çetin, Caspar Greeven, Kristian Laursen, Damian Podareanu, Valeriu Codreanu, Alexandru Uta, and Alexandru Iosup. 2023. Less is not more: We need rich datasets to explore. *Future Gener. Comput. Syst.* 142 (2023), 117–130. doi:10.1016/J.FUTURE.2022.12.022