A Survey of Simulators and Traces for the Compute Continuum

Guanghe Xie
Vrije Universiteit Amsterdam

Matthijs Jansen
Vrije Universiteit Amsterdam

Daniele Bonetta
Vrije Universiteit Amsterdam

Abstract

The compute continuum enables dynamic resource allocation across cloud, edge, and endpoint devices, but also introduces significant challenges for system design and performance evaluation. Simulators and workload traces are critical for realistic modeling and benchmarking, yet current resources are fragmented and lack standardization due to the inherent complexity of the continuum, including device heterogeneity, workload diversity, and use-case specificity. This survey provides a systematic review and taxonomy of existing simulators and trace frameworks for the compute continuum, analyzes their modeling capabilities and trace support, and categorizes available trace datasets. We also discuss key issues in integrating traces with simulators and outline future research directions, including standardization and digital twin approaches. Our survey aims to guide researchers and practitioners in selecting appropriate tools and advancing the state of compute continuum simulation.

1 Introduction

In recent years, the compute continuum has emerged as a comprehensive paradigm that seamlessly integrates cloud computing, edge computing, and endpoint devices [1, 65]. In this continuum, computation, data, and services can flexibly migrate across the cloud, edge, and endpoint layers to better satisfy the diverse and dynamic requirements of modern applications. By leveraging the massive scalability of cloud data centers together with the low latency responsiveness of edge and endpoint devices, the compute continuum enables a new generation of applications ranging from real-time analytics and autonomous vehicles to smart cities and large-scale IoT deployments. Figure 1 provides a simple view of the compute continuum.

However, this highly distributed and heterogeneous continuum infrastructure introduces significant challenges for system design, resource management, and performance optimization [65]. As a result, researchers and practitioners urgently require effective tools to experiment, validate, and optimize new architectures and management strategies before deployment in real-world systems. Among these tools, simulators and traces play foundational roles. Simulators and trace generation frameworks serve distinct purposes in compute continuum research. Simulators create controlled virtual environments to test algorithms and systems under configurable conditions, whereas trace generation focuses on capturing or synthesizing real or representative workload behavior to feed into such simulators.

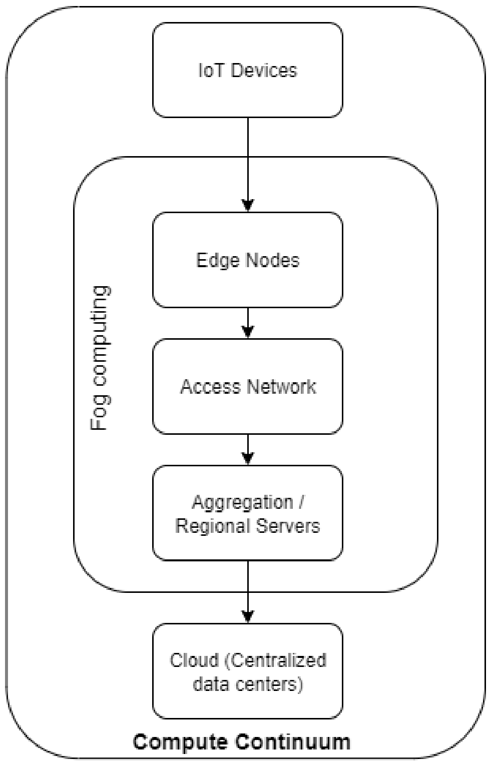

Figure 1: Compute Continuum Layers

Simulators are indispensable tools in the study of cloud-edge computing systems because deploying and experimenting with real large-scale infrastructures is often costly, time-consuming, and in many cases even infeasible [12] [9]. Simulators provide a safe and controlled environment to model the complex behavior of the compute continuum, evaluate resource management algorithms, and test new application scenarios under a variety of conditions. To ensure simulation results are realistic and actionable, it is crucial to drive simulators with accurate and representative traces datasets that record the actual workloads, user behaviors, resource consumption patterns, and event sequences observed in operational systems [24]. High-quality traces make it possible to benchmark system performance, stress-test scalability, and analyze behavior under diverse operating conditions. Conversely, the lack of appropriate traces can lead to misleading results or overly optimistic conclusions.

Despite their importance, there is currently a lack of a clear survey on available simulators and trace generation frameworks specifically tailored for the compute continuum. Existing tools and datasets are often fragmented, focus only on specific layers (cloud or edge), or lack interoperability. This fragmentation is largely due to the sheer scale and complexity of the compute continuum, which encourages specialization tools are often developed for specific use cases, layers, or domains, rather than general purpose continuum wide developing. And the fragmentation complicates the evaluation and comparison of research results, and limits the community's ability to advance the state of the art.

To address these challenges, this survey systematically investigates existing simulators and trace generation frameworks for the compute continuum. It establishes a comprehensive taxonomy, supports researchers and practitioners in selecting appropriate tools, identifies current limitations and integration issues, and outlines future research directions toward more realistic and practical simulation environments.

The key contributions of this survey are as follows:

1. We provide a comprehensive survey and taxonomy of simulators that support compute continuum operations, analyzing their architectural models, simulation paradigms, and target layers. (see Section 4)

2. We review and classify existing trace datasets relevant to the compute continuum, covering real-world and synthetic sources, layer coverage (cloud, edge, or both), and application domains. (see Section 5)

3. We analyze the compatibility between traces and simulators, identifying current limitations, challenges in integration, and future opportunities for standardization, automation, and digital twin–driven simulation. (see Section 6)

2 Related Work

Existing research on the compute continuum spans multiple directions. In this survey, we concentrate on three aspects that are most relevant to building and using simulators and traces: architectural overviews, simulation tools, and trace generation frameworks. This section reviews related work across these areas. We begin with studies that define and structure the compute continuum, followed by surveys of simulation platforms in cloud and edge computing, and conclude with research on trace generation frameworks and public datasets, highlighting their roles and limitations in continuum-based evaluation.

We use the term simulator to encompass any software framework that models or emulates continuum infrastructures, including what some papers call "simulation platforms." Likewise, we use trace to denote any workload log or dataset captured from real systems (or faithfully generated), which some literature refers to as "workload datasets."

2.1 Background on the Compute Continuum

The concept of the compute continuum, spanning endpoint devices, edge nodes, and cloud data centers, has been widely discussed in recent surveys. Al-Dulaimy et al. [1] proposed a unified vision of the compute continuum, outlining reference architectures that integrate computational and communication infrastructures. They emphasized the importance of seamless orchestration across different layers to achieve performance, reliability, and energy efficiency goals. Similarly, Gkonis et al. [25] offered an extensive overview of IoT-Edge-Cloud systems, highlighting challenges such as resource heterogeneity, security, and distributed learning across layers.

Although these studies are valuable for understanding the architectural and operational aspects of continuum systems, they mainly focus on system-level challenges. They do not address simulation tools or trace generation frameworks, which are essential for designing, evaluating, and optimizing applications across the continuum. As a result, researchers lack a comprehensive understanding of the simulation resources and workload datasets available for continuum research.

2.2 Existing Surveys on Simulators

Simulation platforms have been a major focus in cloud and edge computing research. Lata and Singh [38] surveyed over 25 cloud simulators, such as CloudSim, GreenCloud, and iCanCloud, comparing features like virtualization support, energy modeling, scalability, and scheduling policies. Likewise, Fakhfakh et al. [21] conducted a comparative study on cloud simulation tools, focusing on their performance evaluation features, usability, and extensibility.

In the edge computing domain, Le et al. [77] reviewed nine simulation platforms, pointing out the unique needs of

Table 1: Coverage of Key Topics in Recent Surveys and This Survey

Survey	**Compute Continuum**	**Simulator**	**Trace**
Fakhfakh et al. (2017) [21]	✓	✓	
Le et al. (2021) [77]	✓	✓	
Lata & Singh (2022) [38]	✓	✓	
Pati et al. (2022) [56]	✓	✓	
Toczé et al. (2022) [76]	✓		✓
Gkonis et al. (2023) [25]	✓		
Al-Dulaimy et al. (2024) [1]	✓		
Liu et al. (2025) [41]	✓		✓
Mechalikh et al. (2025) [49]	✓	✓	
This Survey	✓	✓	✓

edge simulations, such as mobility modeling, low-latency support, and flexible network topologies. Their work highlighted how edge environments pose different simulation challenges compared to traditional cloud systems. Mechalikh et al. [49] reviewed the development of edge computing simulators over the past decade, with a particular focus on the progress of major open-source platforms since 2018 in terms of modeling capability, scalability, and flexibility. They evaluated the strengths and weaknesses of mainstream platforms and proposed future challenges and prospects. Moreover, Pati et al. [56] expanded the discussion by reviewing simulation tools across IoT, fog, edge, and cloud layers, although with uneven coverage across each layer.

While these surveys provide important overviews, most of them are limited to a single layer, either cloud or edge, and do not consider the compute continuum as a whole. Also, they primarily compare technical features but often overlook how simulations can be driven by real-world workload traces. As continuum environments become more complex, trace-driven, realistic simulations across multiple layers are increasingly necessary, highlighting a gap in the existing literature.

2.3 Existing Surveys and Analyses on Trace Generation Frameworks

Trace generation frameworks and workload datasets are critical for realistic evaluation, but have received less attention compared to simulators. Toczé et al. [76] gathered and analyzed edge workload traces, revealing the limited availability of public datasets and the challenges in modeling edge workloads. Their work emphasized the difficulty of capturing realistic, diverse traces, especially for dynamic and resource-constrained edge systems.

Liu et al. [41] provided a comprehensive survey of public datasets available for cloud computing, identifying 42 datasets and classifying them based on features like size, resource types, application domains, and energy information. While this survey enriches the available resources for researchers, it mainly focuses on cloud environments and offers limited insights into edge or endpoint traces.

In addition to dataset surveys, empirical studies like Alam et al. [2] have performed detailed analysis of specific datasets, such as the Google Cluster Trace, using clustering techniques to characterize workloads based on resource usage patterns. Such studies provide valuable observations but typically focus on a single dataset and do not address broader issues like workload heterogeneity across the continuum or trace-simulator compatibility.

Overall, although some work has been done on trace collection and analysis, there is still no systematic review that connects traces with simulation platforms in a continuum-aware manner.

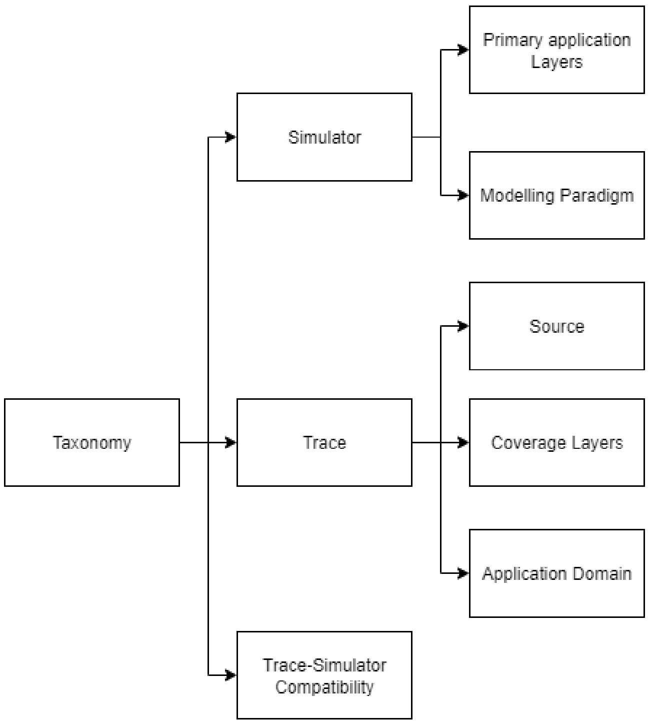

Figure 2: Taxonomy of the survey

2.4 Summary of Gaps

To summarize, prior surveys have either focused on architectural challenges of the compute continuum, compared simulation tools for isolated layers, or cataloged available workload traces within specific domains. However, there is a clear gap in providing a comprehensive, continuum-wide review that jointly examines simulation tools and trace generation frameworks, while analyzing their compatibility and coverage across endpoint, edge, and cloud layers.

Our survey addresses this gap by proposing a unified taxonomy of simulators and traces, analyzing their scope, source layers, usability, and suitability for realistic continuum evaluations. By combining the definition of the compute continuum, the capability analysis of various representative simulators, and the characteristics and compatibility evaluation of public traces, this survey provides an integrated perspective that was previously lacking in this field. This approach is particularly important because architectural choices, simulator capabilities, and real-world trace data are fundamentally interrelated; only by considering them together can researchers ensure that simulation-based evaluation results are reproducible, realistic, and practically valuable in real compute continuum scenarios. In doing so, we aim to offer a solid reference for researchers and practitioners working on the design, benchmarking, and optimization of applications across the compute continuum. Figure 2 presents an overview of the taxonomy developed in this survey.

3 Research Methodology

The design of the research method is a crucial step to explicitly clarify how relevant literature was collected. As shown in Figure 3, this section details the entire process of this literature survey, which mainly includes three primary stages: planning, execution, and extraction.

3.1 Planning

Objective: The main objective of this study is to systematically review and analyze the simulators and trace resources used in the compute continuum, clarify their classification, capability differences, and the key challenges in their integration process.

Research questions: This study addresses the following research questions:

RQ1: What simulators currently exist for compute continuum operations?

RQ2: What traces are available in the compute continuum domain?

RQ3: How well do existing traces support current simulators? What issues exist? How can future research improve simulation and trace generation?

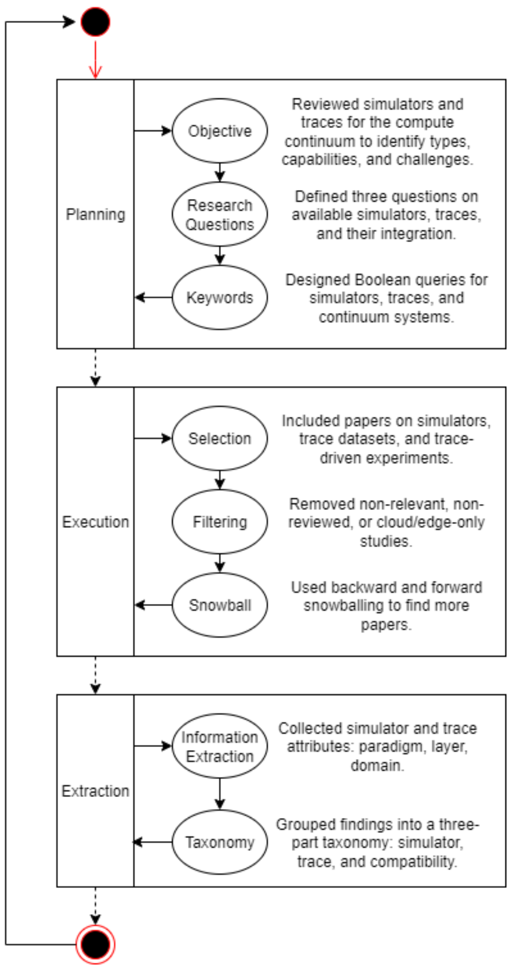
Planning
Objective
Reviewed simulators and traces for the compute continuum to identify types, capabilities, and challenges.
Research Questions
Defined three questions on available simulators, traces, and their integration.
Keywords
Designed Boolean queries for simulators, traces, and continuum systems.
Execution
Selection
Included papers on simulators, trace datasets, and trace-driven experiments.
Filtering
Removed non-relevant, non-reviewed, or cloud/edge-only studies.
Snowball
Used backward and forward snowballing to find more papers.
Extraction
Information Extraction
Collected simulator and trace attributes: paradigm, layer, domain.
Taxonomy
Grouped findings into a three-part taxonomy: simulator, trace, and compatibility.

Figure 3: Research Method

Keyword:
Initial keyword combination:

("cloud-edge" OR "cloud" OR "edge" OR "compute continuum" OR "fog computing") AND ("simulation" OR "simulator" OR "digital twin" OR "emulation") AND ("trace" OR "workload" OR "dataset" OR "benchmark")

Subsequently, Boolean clauses related to specific subtopics (HPC, AI/ML, IoT) were added to ensure coverage of multiple application domains.

3.2 Execution

Selection: This step aims to select papers related to the research field. The preliminary inclusion criteria include studies that propose, evaluate, or extend cloud/edge/fog/continuum simulators; studies that publish, analyze, or synthesize trace datasets at any layer; as well as studies describing trace-driven experiments.

Filtering: After collecting the initial set of papers, further filtering is performed to narrow the range of literature. This is mainly achieved by reading abstracts to exclude studies that focus only on a single layer of cloud or edge, and those unrelated to simulation or tracing, as well as blog articles, master's theses, or papers with unclear research methodology.

Snowball: To enhance the comprehensiveness of the survey and prevent missing relevant literature, the snowballing method was also employed. Specifically, several highly relevant core papers were chosen as the starting point, and both backward snowballing (checking their references) and forward snowballing (identifying later works that cite them) were conducted to systematically expand the literature pool.

3.3 Extraction

Information Extraction: At this stage, the filtered papers are read in depth to extract information related to the research questions. Labels include simulator attributes (such as target layer, modeling paradigm), trace attributes (such as real/synthetic, layer coverage), and application domains.

Taxonomy: The final step is to organize the extracted key information into a structured taxonomy. This taxonomy is mainly structured along three dimensions: simulator, trace, and trace-simulator capability.

4 Simulator Taxonomy

Observations:

O-1: The mainstream simulator ecosystems are represented by the CloudSim and SimGrid families.

O-2: Simulators that support integrated cloud–edge–endpoint modeling remain scarce; cross-layer scalable modeling is a key direction for future development.

O-3: Mobility, fine-grained network modeling, energy consumption, and reliability are becoming standard capabilities in next-generation simulators.

This section provides a structured taxonomy of simulators in the compute continuum domain. After systematically analyzing existing simulators, we categorize existing simulators based on their modeling scope as well as their underlying simulation paradigms. For each category, we summarize representative platforms and highlight their design features, modeling capabilities, and application scenarios.

4.1 Primary application Layers

In this subsection, we examine simulators according to their primary application layers along the continuum. We distinguish between simulators that focus mainly on the cloud layer, those tailored for edge or fog environments, and platforms capable of modeling the integrated cloud–edge ecosystem. The following sections detail representative tools in each category.

4.1.1 Cloud-centric

Cloud computing simulators at the cloud layer provide flexible and cost-effective experimental environments for resource scheduling, system design, and performance evaluation. CloudSim is the most representative open-source platform in this field, capable of finely simulating multiple data centers, virtual machines, tasks (Cloudlets), and various scheduling and energy management strategies [12]. It supports multilevel cloud service modeling such as IaaS and PaaS, and allows users to customize extensions. On this basis, CloudAnalyst simplifies the modeling and visualization of features such as geographic distribution, load balancing, and service cost with a graphical interface, making it suitable for performance analysis of large-scale cloud applications [82]. CloudReports further integrates energy consumption modeling, automatic report generation, and a plug-in extension mechanism, making it convenient to study data center energy consumption and QoS trade-offs under different scheduling strategies [75]. DynamicCloudSim has been specially extended to address resource heterogeneity, dynamic performance fluctuations, and node failures in real cloud environments, and can support complex instability simulation of public clouds such as Amazon EC2, as well as algorithm testing in scenarios like scientific workflow scheduling [10].

In addition to mainstream tools represented by CloudSim, the cloud computing field has also seen the emergence of various simulation platforms focusing on different dimensions. GreenCloud, as an extension of NS2, pays more attention to energy consumption modeling and fine-grained analysis of network communications [33]. It can track the energy consumption distribution of servers, switches, and links in data centers, and is especially suitable for research on energy efficiency optimization and green computing schemes. However, due to its packet-level simulation approach, GreenCloud is relatively limited in simulation efficiency for ultra-large-scale cluster scenarios. In contrast, iCanCloud features flexible virtual machine modeling and high scalability, supports customizable hardware configurations and diverse scheduling strategies, and can efficiently reproduce the application scenarios of mainstream cloud services such as Amazon EC2 [55]. The C++ implementation of iCanCloud brings better runtime performance and resource management capabilities, making it suitable for cloud environment modeling that requires a large number of experiments and parameterized evaluation. The latest GAME-SCORE combines energy consumption and performance scheduling by introducing a Stackelberg game model, dynamically coupling scheduling strategies with energy efficiency management, and achieving adaptive trade-offs in resource allocation and energy optimization [22].

Overall, these cloud-only simulation platforms each focus on energy modeling, scalable experimentation, or dynamic scheduling mechanisms, greatly enriching the performance evaluation and algorithm verification methods of cloud computing systems.

4.1.2 Edge/Fog-centric

The academic community has developed a variety of simulation tools targeting edge and fog computing environments, which have greatly promoted research and innovation in related fields. Among them, iFogSim is currently the most widely used fog computing simulation platform. Based on CloudSim, this tool extends support for hierarchical modeling of large-scale heterogeneous fog nodes and IoT devices, and enables flexible configuration of physical, logical, and management components. It can evaluate multiple metrics such as end-to-end latency, network load, energy consumption, and QoS satisfaction, and is widely used in simulation studies of resource allocation and energy efficiency optimization [27]. Building on this, MyiFogSim extends iFogSim to address resource allocation needs in mobile scenarios, with a particular focus on supporting virtual machine migration (VM migration) for mobile users [42]. By introducing user mobility, access point (AP) modeling, and various migration strategies, MyiFogSim can simulate dynamic migration of a user's virtual machine to the nearest cloudlet as the user moves across different access points, thereby effectively reducing latency and improving service quality.

Meanwhile, FogNetSim++ addresses the shortcomings of existing simulators in network modeling and mobility support by proposing a more flexible simulation framework [61]. This platform not only supports heterogeneous devices and various mobility models, but also allows customized node management and scheduling algorithms. It can realistically simulate distributed deployment, device handover, and multiple protocol interactions (such as MQTT/CoAP), significantly improving modeling capabilities in complex network environments. In addition, YAFS, a new simulator implemented in Python, adopts complex network theory for topology modeling, supports dynamic deployment, node failure, and user mobility strategies, facilitates integration with third-party tools, and provides fine-grained event recording and JSON scenario definition, making it suitable for IoT/Fog application evaluation in large-scale, dynamic, and heterogeneous environments [39].

Overall, mainstream edge/fog simulation tools are continuously evolving towards heterogeneous modeling, protocol support, dynamic scenario extension, and security and trustworthiness, greatly enriching the means for performance evaluation and mechanism innovation of edge computing systems.

4.1.3 Compute Continuum

Compared with cloud-centric or edge-centric simulators, simulation tools capable of unified modeling and analysis of the compute continuum remain relatively scarce. ENIGMA is a recently proposed scalable simulation platform that supports multilayer modeling of large-scale IoT, edge, fog, and cloud environments [18]. It introduces features such as device mobility, energy consumption modeling, and multilevel task distribution, making it suitable for resource optimization and performance analysis in complex scenarios like smart cities. Built as an extension to SimGrid, ENIGMA maintains high scalability while allowing detailed evaluation of different algorithms in continuum environments.

Another representative tool is IoTSim-Osmosis, which is specifically designed for modeling and analyzing complex IoT applications in integrated edge–cloud (osmotic computing) environments [5]. This tool supports multilayer heterogeneous resource modeling, dynamic task migration, and SDN/SD-WAN network coordination. It represents IoT applications using microservice graphs (MEL Graph), making it especially suitable for analyzing end-to-end QoS, energy consumption, and complex scheduling strategies in practical applications.

EdgeCloudSim extends CloudSim by enabling collaborative modeling between the edge and the cloud [72]. It enhances support for device mobility, wireless/wide-area network modeling, and edge orchestration, allowing flexible analysis of the impact of different architectures and service offloading strategies on system performance such as latency, bandwidth, and energy efficiency. Its modular design also facilitates the extension of various scenarios and customization of experimental parameters.

In addition, RECAP Simulator targets next-generation cloud/edge/fog capacity optimization, emphasizing integration with real resource monitoring, resource allocation, and optimizers [11]. It supports multilayer deployment, large-scale devices and tasks, and failure event simulation, making it suitable for QoS, energy consumption, and cost strategy verification and system evaluation in large-scale environments.

Overall, although compute Continuum Simulators continue to advance in aspects such as heterogeneous multilayer modeling, mobility, and network support, tools that can truly realize an end-to-end complete continuum while also offering scalability and ease of use remain limited. Related research and tools are still being continuously improved and developed.

4.2 Modeling Paradigm

Observations:

O-4: DES (Discrete Event Simulation) can scale to tens of thousands of nodes on a single machine by advancing simulation time through events, but it can-

Table 2: Representative simulators: layer, paradigm, and notes

Simulator	**Layer**	**Paradigm**	**Notes**
CloudSim	Cloud	DES	Multi-datacenter, VM, Cloudlet; scheduling and energy models; IaaS/PaaS modeling; extensible.
CloudAnalyst	Cloud	DES	GUI for geographic distribution, load balancing, and cost analysis; built on CloudSim.
CloudReports	Cloud	DES	Energy modeling and automatic report generation; plug-in extensions.
DynamicCloudSim	Cloud	DES	Models heterogeneity, performance fluctuations, and node failures.
GreenCloud	Cloud	DES	NS2-based; energy-focused with packet-level network analysis; less efficient for ultra-large scale.
iCanCloud	Cloud	DES	C++ implementation; configurable hardware; scalable for large experimental campaigns.
iFogSim	Edge/Fog	DES	Hierarchical heterogeneous fog/IoT modeling; evaluates latency, network load, energy, QoS.
MyiFogSim	Edge/Fog	DES	Extends iFogSim for mobility; VM migration to nearest cloudlet to reduce latency.
FogNetSim++	Edge/Fog	DES	Heterogeneous devices and mobility models; customizable management; MQTT/-CoAP.
YAFS	Edge/Fog	DES	Python; complex-network topologies; dynamic deployment/failures/mobility; JSON scenarios.
ENIGMA	Continuum	DES	Scalable multilayer IoT/Edge/Fog/Cloud modeling; mobility and energy; built on SimGrid.
IoTSim-Osmosis	Edge+Cloud	DES	Integrated edge–cloud (osmotic) modeling; dynamic migration; SDN/SD-WAN; microservice (MEL) graphs.
EdgeCloudSim	Edge+Cloud	DES	Extends CloudSim; mobility; wireless/WAN modeling; edge orchestration; offloading analysis.
RECAP Simulator	Continuum	DES	Capacity optimization; integrates monitoring and optimizers; multilayer deployment; failure events.
SimGrid (family)	Cloud/HPC	DES	High-performance C++ event engine; large-scale scenarios on a single machine.
Mininet	SDN/Network	Emulation	Real protocol stacks/apps via Linux namespaces; single-machine rapid SDN prototyping.
MaxiNet	SDN/Network	Emulation	Distributed extension of Mininet across multiple hosts; thousand-node experiments.
EmuFog	Fog/Network	Emulation	Builds on Mininet/MaxiNet; fog nodes, topology generation, Docker instances; multi-machine emulation.
WoTemu	Edge/IoT	Emulation	Docker Swarm with W3C WoT; traffic shaping; metrics collection; horizontal scaling; runs real code.
Simu5G	Edge/5G	Hybrid	OMNeT++/INET; 5G NR stack; DES coupled with real sockets; supports real-time HIL/MEC setups.

not directly run real applications.

O-5: Emulation retains real protocol stacks and binaries, making it suitable for high-fidelity and latency-sensitive validation.

O-6: The hybrid approach combines real protocols on critical nodes with abstract models for the rest, striking a balance between scalability and realism.

The choice of simulation paradigm greatly influences the fidelity, scalability, and usability of a simulator. In this subsection, we discuss three main modeling paradigms: discrete event simulation (DES), emulation, and hybrid approaches, each offering distinct advantages and trade-offs.

4.2.1 Discrete Event Simulation (DES)

Discrete Event Simulation (DES) advances simulation time by processing an event queue sorted in chronological order. Each event modifies the global state, such as virtual machine startup or task completion, and may schedule new events [46]. Since the overhead of time advancement is proportional to the number of events rather than to real wall-clock time, DES can scale to tens of thousands of nodes on a single laptop. Both the CloudSim family and SimGrid family of mainstream simulators are built around a native DES engine.

In terms of tooling ecosystem, the CloudSim family (CloudSim, CloudAnalyst, EdgeCloudSim, etc.) provides Java-based IaaS/PaaS modeling interfaces, supporting virtual machine and container lifecycles, energy consumption and cost models, and, in EdgeCloudSim, introduces wireless links and terminal mobility [12] [82] [72]. As a result, it is widely used for resource scheduling and energy efficiency research in cloud-centric or edge-centric scenarios. In parallel, the SimGrid family (SimGrid 4, ENIGMA, RECAP Simulator, etc.) employs a high-performance C++ core and optimized event queue management, enabling replay of scenarios with over a hundred thousand nodes on a single machine [15] [18] [11]. Its derivative tools extend DES capabilities to large-scale compute continuum environments by incorporating mobile devices, fog nodes, and network hierarchies.

However, DES cannot directly run real applications. Moreover, research shows that as model details increase, the accuracy gains of DES diminish, and event queue management can become a bottleneck [64]. Therefore, when research focus shifts to real systems, scholars may turn to emulation.

4.2.2 Emulation

Compared with discrete event simulation (DES), emulation retains real protocol stacks and application binaries, allowing experimental traffic to progress according to the real wall-clock time [26]. It uses virtualization or container technologies to simulate network links, latency, and bandwidth, thereby providing researchers with a controllable and repeatable experimental environment.

Mininet and MaxiNet are currently the most widely used SDN network emulation platforms. Mininet is a lightweight network emulator that uses Linux processes and network namespaces to run real protocol stacks and applications on a single machine, and is commonly used for rapid prototyping of SDN [37]. Building on this, MaxiNet extends Mininet to a distributed architecture, allowing virtual networks to be partitioned and run across multiple physical machines, thereby supporting simulations at the scale of thousands of nodes and enabling the emulation of real data center traffic [81].

EmuFog, based on MaxiNet/Mininet, integrates fog computing nodes, topology generation, and Docker application instances into a scalable framework, enabling real-time emulation of large-scale fog computing infrastructures across multiple physical machines and supporting the evaluation of edge node placement algorithms [47]. WoTemu combines container orchestration (Docker Swarm) with the W3C Web of Things standard, supporting automatic traffic shaping, application-level metric collection, and horizontal scaling, all while running real code [43]. These tools allow direct execution of production-grade Docker images, avoiding the need to re-abstract business logic.

Emulation is suitable for high-fidelity validation of protocol prototypes, latency-sensitive edge/fog computing applications, and test scenarios requiring integration with real devices or third-party systems.

4.2.3 Hybrid

The hybrid paradigm combines the scalability of DES with the high fidelity of emulation, allowing the simulation kernel to operate collaboratively with external real applications or devices through a “soft real-time” mechanism.

Simu5G, based on the OMNeT++/INET framework, implements the 5G NR protocol stack and couples the event-driven scheduler with real TCP/IP sockets. Researchers can perform large-scale DES offline, or connect to MEC platforms or vehicular terminals in real-time mode to conduct end-to-end HIL evaluation [53]. Its lightweight base station mechanism simulates interference while simplifying higher-layer protocols, enabling hybrid scenarios with a dozen base stations to run in real-time on a single machine.

Hybrid approaches offer the advantages of both paradigms: they can run real protocols on critical nodes while using abstract models to simulate the rest, thus maintaining scalability.

Table 2 provides a structured summary of representative simulators, covering their primary layer focus, modeling paradigm, and notable features.

5 Trace Taxonomy

Observations:

O-7: Real-world traces are representative but often suffer from missing fields, incompleteness, or anonymization; synthetic traces offer flexibility and control but require rigorous validation.

O-8: Publicly available datasets are predominantly cloud-focused, with a clear lack of edge-side and cloud–edge joint traces.

O-9: The application domain (HPC/AI/IoT) determines event granularity and metric sets, which directly influence simulation mapping strategies.

In this section, based on existing work, we propose a taxonomy for categorizing existing traces relevant to the compute continuum. The classification focuses on three key aspects: the source of the trace, its coverage layers across cloud and edge infrastructures, and the application domain it targets.

5.1 Source

The source of a trace describes how the dataset was generated or collected. We distinguish between two main types: Real-world traces and Synthetic traces.

5.1.1 Real-world

Real-world traces refer to datasets collected from operational systems functioning in production environments. They capture authentic workload behaviors, resource usage patterns, and system events over time, providing critical insights for modeling and evaluating real-world system performance [63] [24]. Although highly valuable, real-world traces often face challenges such as data incompleteness (e.g., missing logging periods and unavailable features) [30].

A representative example is the Google cluster usage trace, which records production workload data from a compute cell managed by Google's Borg system. It provides fine-grained information on job and task lifecycles, machine events, and resource utilization patterns across thousands of nodes. Due to privacy considerations, identifiers and resource units are obfuscated. Despite its richness, the trace presents challenges such as data omissions and the lack of certain scheduling dependencies [63].

5.1.2 Synthetic

Synthetic traces refer to datasets that are artificially generated rather than collected from operational systems. They are created using workload generation tools, simulation platforms, or statistical modeling techniques to replicate realistic workload behaviors. Synthetic traces address scenarios where real-world traces are scarce, incomplete, or unavailable due to privacy, operational, or scalability constraints.

Several approaches exist for generating synthetic traces. One approach relies on profiling real-world cloud workloads to model key behavioral characteristics, such as request arrival patterns and resource usage distributions. Tools like TRAGEN generate synthetic request streams to reproduce traffic patterns in distributed systems and cloud environments [67].

In addition, Bahga and Madisetti proposed a workload characterization and modeling framework for cloud applications, where benchmark and workload models are extracted from real traces and used to drive synthetic workload generators [8]. Their method enables sensitivity analysis and scalable benchmarking without relying on access to operational cloud data.

While synthetic traces offer flexibility, reproducibility, and scalability, they may not fully capture the nuanced, unpredictable behaviors present in real-world systems. Careful modeling and validation are essential to ensure their representativeness for the intended application.

5.2 Coverage Layers

Coverage layers define which layers of the compute continuum a trace represents. We classify traces into three categories based on their target layers: Cloud-only, Edge-only, and Cloud+Edge.

5.2.1 Cloud-only

Cloud-only traces focus exclusively on workloads executed within centralized cloud infrastructures. They typically record data related to virtual machine (VM) provisioning, container orchestration, resource allocation, and system-level events within large-scale data centers [63]. Cloud traces play a crucial role in enabling resource optimization and elastic scaling in cloud systems. For instance, in [69], the authors propose CloudScale, a system that uses resource usage prediction to automate elastic scaling decisions. The vast majority of traces currently released are Cloud-only traces.

Prominent examples of cloud-only traces include the Alibaba Cluster Trace [3], which provides detailed information on containerized workloads, user job submissions, and resource consumption metrics collected from Alibaba's internal production clusters. Similarly, the Microsoft Azure VM Workload Trace [51] captures virtual machine lifecycles, resource utilization patterns, and deployment behaviors across Microsoft's cloud infrastructure. Both traces have been extensively utilized in research on cloud workload characterization, scheduling simulations, and resource optimization strategies.

Despite their richness, cloud-only traces often lack information about decentralized or distributed computing across the continuum, making them insufficient for modeling emerging hybrid edge-cloud scenarios without significant adaptation.

5.2.2 Edge-only

Edge-only traces record system-level activities occurring at the network edge, closer to end devices. These traces typically capture interactions involving IoT sensors, mobile devices, or fog nodes, but do not include raw user content. However, truly publicly available edge-only trace datasets are extremely rare. This is mainly due to the fact that edge devices often handle sensitive information, raising security and privacy concerns, as well as the lack of standardized environments and benchmarking practices [32] [34]. Moreover, since cloud and data center traces are more readily available and can be partially applied to edge scenarios, many existing studies still prefer to rely on these available datasets.

In response to this gap, recent research efforts have emerged. Toczé et al. propose a systematic methodology for edge workload classification and trace gathering, releasing open-source traces from real-world applications to foster the development of representative benchmarks for edge computing research [76]. Similarly, Samani et al. released an edge infrastructure trace dataset collected from heterogeneous devices running microservices for video processing tasks, which includes execution times, bandwidth, and latency measurements across different edge nodes [68].

5.2.3 Cloud+Edge

In our taxonomy, the Cloud+Edge layer includes traces that span edge devices and cloud infrastructure, reflecting the complexity and comprehensiveness of compute continuum environments. Unlike scenarios limited to cloud or edge alone, Cloud+Edge traces integrate data flows and computations across multiple heterogeneous environments, bringing unique challenges to resource management and performance optimization. Although the importance of such traces is increasing, comprehensive datasets and detailed studies that fully cover the spectrum from edge to cloud remain scarce.

Kolosov et al. [35] proposed a workload composition method. They realized that existing datasets usually capture only partial characteristics of edge computing, such as user mobility, storage access patterns, or computational demands, and therefore advocated combining attributes from multiple existing datasets. This innovative method enables researchers to construct synthetic yet realistic workloads that accurately reflect the complex interactions in edge-to-cloud scenarios, providing a new perspective for related research in the absence of comprehensive real-world datasets.

To better capture end-to-end trace data, Rosendo et al. [66] developed ProvLight, a provenance capture tool integrated into the E2Clab framework. Their method focused on workflow provenance collection across IoT/edge devices and cloud/HPC infrastructures. By deploying federated learning workloads on large-scale testbeds such as FIT IoT-LAB and Grid'5000, they successfully captured detailed runtime traces. These datasets include key execution provenance, such as inputs, outputs, and intermediate states, providing valuable insights for analyzing system performance in compute continuum scenarios.

The work of Tanaka et al. [74] also contributed to trace research from edge to cloud. They extended the Pegasus Workflow Management System to support automated workflows across edge and cloud infrastructures. In their experiments on the Chameleon Cloud testbed, they used representative workflows such as CASA-Wind and Orcasound, which generate data on edge devices and perform subsequent intensive computations in the cloud. The integrated monitoring tools of Pegasus captured detailed provenance and performance data, enabling researchers to systematically analyze the impact of different task placement strategies on overall system efficiency.

Overall, although these research results are not complete trace databases for Cloud+Edge environments, they provide some of the key methods and tools needed to develop realistic, trace-driven models in Cloud+Edge environments. They lay the foundation for future innovation and optimization in the compute continuum field.

5.3 Application Domain

Application domain refers to the specific application scenarios and fields that the traces correspond to. Beyond the actual workload content, the application domain also shapes how a trace is structured: it determines the granularity of events (e.g., job-level records in HPC versus packet-level logs in IoT), the resource metrics captured (CPU memory for HPC, GPU utilization for AI/ML, network flows and sensor data for Edge/IoT), and even the labeling or provenance information provided. Consequently, different domains require distinct preprocessing steps and simulation mappings. In the following, we will specifically discuss three representative application domains: High-Performance Computing (HPC), Artificial Intelligence and Machine Learning (AI/ML), and Edge/IoT, reviewing the existing research and related achievements in each domain.

5.3.1 HPC Traces

The collection and analysis of traces in the HPC field have always been crucial for understanding and optimizing supercomputing systems, yet challenges remain in terms of data diversity and accessibility. Amvrosiadis et al. [6] pointed out that research overly relying on a single dataset (such as the Google cluster trace) leads to results that are difficult to generalize. They introduced the LANL Mustang and Trinity HPC traces and found that these traces differ significantly from the Google trace as for job size, duration, and failure patterns, emphasizing that the evaluation of new methods requires diverse and representative traces.

In recent years, researchers have been expanding the scope of trace analysis to address the diversity of modern HPC workloads. Chu et al. [16] compared traditional jobs and machine learning (ML) jobs in a large HPC data center and found that although ML jobs are relatively few in number, they account for a disproportionately high share of energy consumption, and a large amount of energy is actually consumed by jobs that do not successfully complete. This provides significant opportunities for future improvements in resource scheduling and energy efficiency. Publishing traces containing information on jobs, nodes, and energy consumption, as they did, helps provide a deeper understanding of system efficiency and reliability issues.

In the aspect of trace collection, McKerracher et al. [48] proposed the FRESCO framework, which provides a unified trace dataset for HPC environments across multiple institutions and clusters. It combines job-level attributes with fine-grained performance metrics through standardized data collection and integration processes and makes the dataset publicly available to support cross-cluster and cross-institution workload analysis.

On the other hand, the automatic generation of synthetic traces has become a research hotspot. Paul et al. [57] proposed a machine learning-based HPC I/O trace generation method that uses Darshan fine-grained logs. Through feature engineering and a two-stage deep generative model, the method can generate synthetic HPC traces that closely resemble real ones, alleviating the difficulty of obtaining real traces and thus enabling performance simulation and evaluation in scenarios where large-scale real traces are unavailable.

In conclusion, the HPC trace ecosystem is evolving toward more diverse, with increasing emphasis on standardization and openness. These developments can help researchers gain a more realistic understanding of HPC traces.

5.3.2 AI/ML Traces

Over the past few years, with the rise of large-scale machine learning and deep learning applications, the industry has continuously released real trace data from large-scale GPU clusters to support in-depth research on AI/ML systems in both academia and industry. Alibaba has successively published several representative GPU cluster traces, covering different types of ML tasks and real production scenarios [3]. Weng et al. [79] released and analyzed a two-month trace from Alibaba's PAI cluster, which contains over 6,000 GPUs and encompasses training, inference, computer vision, NLP, recommendation, reinforcement learning, and other tasks, while revealing scheduling and resource utilization issues under heterogeneous hardware and diverse workloads. Weng et al. [80], based on actual inference traces, analyzed the high dynamism and high concurrency characteristics of large-scale inference tasks, and emphasized the need for resource isolation and scheduling optimization. Yang et al. [83], on the basis of DLRM inference traces, proposed a GPU resource decoupling and heterogeneous scheduling system, effectively improving the resource utilization and elasticity of inference clusters, with the related traces also made publicly available.

Additionally, Jeon et al. [29] systematically analyzed deep learning training traces from multi-tenant GPU clusters in Microsoft's production environment, revealing practical challenges such as gang scheduling, locality constraints, and resource fragmentation, and providing real data support for multi-tenant scheduling optimization.

In terms of advancing standardization and toolchain development, Sridharan et al. [73] proposed the Chakra framework, which defines a unified execution trace (ET) format, supports automatic collection and conversion of traces from mainstream deep learning frameworks, and integrates generative AI techniques to synthesize privacy-preserving representative traces. This system not only promotes the open sharing of traces but also provides flexible support for distributed training simulation and benchmarking.

Meanwhile, Sibai et al. [70] conducted detailed micro level trace profiling on AI benchmarks such as AIBench, using VTune Profiler to capture multidimensional performance events such as CPU and memory, and employed machine learning models to achieve automatic workload classification and performance prediction based on traces. This has brought new insights for AI processor design and heterogeneous system optimization.

5.3.3 Edge/IoT Traces

With the development and popularization of edge computing and the Internet of Things (Edge/IoT) technologies, researchers have continuously constructed and released real trace datasets to support tasks such as network security, traffic modeling, and intelligent analytics. Ferrag et al. [23] proposed the Edge-IIoTset dataset, which is based on a complex seven-layer experimental platform and collects network traffic from more than ten types of IoT/IIoT devices under various attack scenarios. The dataset supports federated learning and multiple analysis paradigms, emphasizing feature richness and broad attack coverage. Neto et al. [54] released the CICIoT2023 dataset, which was collected in an environment with 105 real IoT devices and includes 33 types of attacks as well as normal traffic. All attacks are launched by real malicious IoT devices, highlighting the heterogeneity of the devices and the diversity of attack methods.

Koroniotis et al. [36] designed the Bot-IoT dataset, which combines virtual and emulation environments, covers both normal and various attack traffic, and provides detailed labels of attack types and protocol features. It has been widely used in IoT security detection research. Moustafa et al. [52] proposed the TON_IoT dataset, which is based on a real edge-fog-cloud three-layer architecture and collects heterogeneous traces such as network traffic, IoT/IIoT sensor data, and operating system audit logs. It covers nine types of modern network attacks, all data are provided with high-quality labels, and it is specifically designed for the evaluation of AI-driven security analytics systems, becoming an important data foundation for smart city and Industry 4.0 security research.

Furthermore, in practical application fields, Peyman et al. [58] collected real-time IoT traces from transportation systems at the edge layer, combined them with optimization algorithms for urban traffic scheduling, and demonstrated the potential of trace data in smart transportation and large-scale urban IoT systems.

6 Trace-Simulator Compatibility

This section explores the compatibility between workload traces and mainstream simulators in the cloud and edge computing domains. We analyze the extent to which different simulators support the replay and integration of traces, and discuss how these traces are adapted or utilized in simulation workflows.

Table 3: Trace-support levels of representative cloud/edge simulators

Simulator	Trace support	Notes
CloudSim (orig.)	Partial	Ships with small PlanetLab samples; real traces must be converted to `Cloudlet` format.
CloudSim Plus	Yes	Built-in parsers (e.g. Google cluster CSV/JSON) map jobs and hosts directly.
EdgeCloudSim	No	Only synthetic load generators; real traces require custom workload generators.
iFogSim / iFogSim2	Partial	No official parser; public logs (Google, Alibaba, Azure) can be replayed via user scripts.
YAFS	Partial	Highly programmable; users can ingest any log and replay events, but scripts are required.
SimGrid	Yes	Supports XML/CSV execution traces (MPI, SimDag) for event-driven simulation.
FaaS-sim	Yes	Designed for serverless; replays production invocation logs out-of-the-box.
OpenDC	Yes	GUI/CLI import of diverse workload traces; flexible format handling.
PerficientCloudSim	Yes	CloudSim extension with direct Google/Azure trace ingestion and VM-migration support.
K8sSim	Yes	Kubernetes-focused; includes parser for Alibaba K8s job traces.
DISSECT-CF	Yes	Fine-grained simulator; accepts device-level logs or synthetic distributions.

6.1 Trace Support in Common Cloud/Edge Simulators

Observations:

O-10: Simulators differ significantly in their built-in support for traces, ranging from "sample-only/manual script import" to "native parsers included."

O-11: Emerging tools targeting Kubernetes and Serverless platforms (e.g., K8sSim, FaaS-sim) tend to adopt native trace-driven simulation approaches.

Currently, there are many simulators widely used in the fields of cloud computing and edge computing, but their support for workload traces (trace) varies. The following is a brief overview of major simulators and their support for real/synthetic traces:

CloudSim series: A classic discrete-event cloud simulation tool. CloudSim can read traces from given text files and create VMs accordingly; however, it does not support reading large trace files for long durations, nor dynamically creating VMs at runtime [84]. For example, CloudSim's built-in examples use CPU utilization traces from the PlanetLab platform to drive VM workloads, but these traces are merely sample data bundled with the simulator [7]. Later versions and extensions of CloudSim have enhanced trace support: CloudSim Plus, for instance, adds a reader for Google cluster traces, allowing real datacenter traces to be parsed into simulation tasks (Cloudlet) and hosts [71].

EdgeCloudSim: Extended from CloudSim and focused on edge computing scenarios. In terms of trace support, EdgeCloudSim does not directly provide a module for reading real traces; users typically generate workloads via custom code (e.g., tasks arriving according to a Poisson process, requests from mobile nodes generated according to a given distribution). Therefore, for real trace data, users need to convert the traces into EdgeCloudSim's simulation input format (such as generating corresponding tasks and events) [72].

iFogSim: Built on CloudSim for simulating IoT, fog, and edge computing. Trace support is mainly reflected in allowing users to customize task generation: researchers usually program assumed scenarios specifying the sensor data rate or task arrival events, without directly reading external trace files [27]. iFogSim does not officially support direct import of Google/Alibaba traces, but research cases have shown that such traces can be used for simulator validation and scenario configuration. For example, some studies use Google cluster data, Alibaba cluster traces, and Azure traces in iFogSim to create different workload scenarios and test resource scheduling algorithms. In these cases, researchers convert public trace data (task scheduling records, load variation over time, etc.) into task submission and resource usage patterns within iFogSim, making the simulation closer to reality. Overall, iFogSim supports trace-driven experiments via code flexibility, but lacks a dedicated trace parser module [40].

YAFS: Yet Another Fog Simulator, suitable for IoT–fog–edge environments. YAFS is characterized by flexible event definition, support for dynamic topology changes, and customizable strategies. Regarding trace support, YAFS allows importing user and mobile terminal location movement routes, which is very useful for simulating mobility scenarios. As for computing task traces, YAFS leans toward users defining via code when and what tasks are generated. However, due to its high programmability, users can easily read real system task logs and replay them according to time in the simulation process, thereby achieving trace-driven simulation [39].

SimGrid: A high-performance simulator for distributed systems, widely used in HPC, grid, and cloud system research. SimGrid emphasizes precise simulation of network communications and parallel task execution, supporting the loading of execution traces (such as MPI traces, event lists) into the simulation to achieve event-driven simulation flows [14].

FaaS-sim: A new class of simulators designed specifically for Serverless/FaaS scenarios. FaaS-sim is inherently trace-driven: it records function invocation logs (such as request arrival times, resource usage) from real serverless platforms and replays them in simulation to evaluate edge serverless deployment's function scheduling and scaling strategies. FaaSsim can replay real platform invocation traces and supports adjusting function behavior, providing simulation results that match actual serverless workloads [62].

OpenDC: An open-source datacenter simulation platform emerging in recent years, offering a graphical interface and model-driven simulation. OpenDC particularly emphasizes workload trace and metric collection: users can import custom workload traces and set monitoring metrics via the interface, which are then executed by the built-in discrete-event engine. OpenDC comes with some predefined scenarios, such as serverless and HPC cluster scenarios, and supports importing real traces from various sources. Although OpenDC is mainly aimed at datacenter computing, it supports multiple workload types and is relatively flexible regarding trace file formats, making it adaptable to diverse data sources. One of its design goals is to lower the threshold for using real traces in simulation [45] [7].

In addition to the above tools, many other simulators have unique features. For example, **PerficientCloudSim** extends CloudSim to support heterogeneous resources and VM migration, and uses real Google/Azure traces to validate the model [84]; **K8sSim** is designed for Kubernetes scheduling and uses real Alibaba Cloud traces to evaluate the performance of different scheduling algorithms [78]; **DISSECT-CF** focuses on fine-grained resource simulation, supporting device behavior defined by trace files or workload generation according to distributions [31].

As shown in the above comparison, the degree of trace support varies significantly among simulators. Some require researchers to manually adapt traces for import, while others have built-in parsing modules; some focus on specific trace types (such as serverless functions or Kubernetes jobs), while others rely on synthetic models to generate workloads. The following section further discusses types of real traces and their current application in simulators.

6.2 Application of Traces in Simulators

Observations:

O-12: In practice, techniques such as sampling, windowing, and feature extraction are commonly used to convert large-scale traces into reproducible simulation inputs.

Real workload traces usually originate from the operational logs of large-scale clusters or cloud platforms and consist of job submissions and resource usage data from real users. In cloud computing research, the Google cluster trace, Alibaba cluster trace, and Microsoft Azure trace are the most representative public datasets. However, different traces vary in their suitability for use in simulators:

Google Cluster Trace: Released by Google, this dataset provides one month of datacenter scheduling logs, containing about 670,000 jobs and 25 million task records across 12,000 machines. The trace records task lifecycle events (submission, scheduling, start, finish, etc.) and resource requests/usage (normalized values), among other information [63]. The data volume and complexity of the Google trace are significant. When applying it in simulators, issues such as extraction and conversion arise: researchers typically do not validate all 25 million tasks, but rather select a specific time window or a sampled subset to reduce simulation scale. For example, the GloudSim tool is specifically designed for the Google trace and generates sampled task sets for simulation after analyzing trace characteristics. GloudSim considers unique task behaviors in the trace (such as checkpoint/restart overhead and constant task memory usage), reproducing them accurately in the simulation [19]. A major feature of the Google trace is that fields are anonymized and normalized: for confidentiality, almost all numerical values (such as CPU and memory amounts) are linearly normalized to their maximum values. This means simulation requires de-normalization or assumptions about actual scale [59]. The Google trace also contains a wealth of scheduling semantics (such as priority queues, preemption, and failure retries), often exceeding the original models of traditional simulators and necessitating custom simulation logic. To narrow the semantic gap, some simulation extensions (such as GloudSim) implement task interruption and recovery mechanisms so that the simulation can capture preemption and checkpoint behavior present in the trace [19]. In summary, due to its scale and detail, the Google trace is an important benchmark for cloud simulation research, but it must be extracted, abstracted, and adapted before it can be effectively replayed in simulators.

Alibaba Cluster Trace: Alibaba has released cluster job scheduling data, including a series of batch and long-running online service workloads [3]. In simulation applications, because the Alibaba trace is of moderate scale, direct replay is more feasible. Some scheduling studies load job lists from the Alibaba trace into simulation frameworks to emulate resource scheduling and validate algorithm performance under real workloads [50]. Some researchers have developed the K8sSim simulator, using the Alibaba trace as input to reproduce the scheduling process and compare the effectiveness of different Kubernetes scheduling algorithms [78]. This demonstrates the value of using the Alibaba trace in simulators: it preserves real workload patterns while improving testing efficiency.

Azure Cloud Platform Trace: Microsoft Azure has released several VM workload traces. Compared to the Google Cluster Trace or Alibaba Cluster Trace, Azure's trace focuses more on the statistics of VM requests and resource usage, and offers less support for task-level details (such as scheduling events, failures, or preemption). The Azure trace is often used to validate the effectiveness of scheduling models, performance prediction, or resource allocation strategies [17].

In summary, real traces play a benchmark role in cloud/edge simulation, enabling evaluation of how closely simulators match real-world scenarios. However, due to their complex formats and massive size, direct application is impractical, and they often require sampling, conversion, or expansion. The advantage of synthetic traces is flexibility and controllability, but the downside is the possible omission of details and burst patterns present in real workloads. Therefore, some research suggests combining real traces with simulation modeling: first extract statistical models from real traces, then use them to generate longer or larger-scale simulation traces [60].

6.3 Key Challenges in Integrating Traces with Simulators

Observations:

O-13: Format incompatibilities and differences in field semantics result in substantial preprocessing and parsing efforts.

O-14: Semantic gaps—such as priority scheduling, preemption, and container scaling—make simple trace replay insufficient to reflect real-world decision logic.

To systematically identify and categorize the challenges in integrating traces with simulators, we followed an iterative bottom-up coding approach. We reviewed multiple recent academic papers and technical reports that described trace driven simulation efforts in cloud, edge, and IoT systems. Through open coding, we extracted frequently mentioned obstacles and recurring patterns, which we then grouped into four main categories based on their technical nature and impact on simulation: (i) format compatibility, (ii) granularity differences, (iii) semantic gap, and (iv) performance bottlenecks. These categories reflect the most commonly encountered and discussed issues in the literature, and serve as a structured lens for analyzing trace–simulator integration.

Format compatibility issues: Traces from different data sources vary in format, field meanings, and granularity, while each simulator has fixed requirements for input formats. This leads to obvious incompatibility. For example, the Google cluster trace consists of multiple CSV logs, including machine events, task events, resource usage, etc., which need to be combined to describe task execution [63]. CloudSim, by design, does not natively support directly reading the CSV format of the Google trace. Therefore, to simulate real workloads, it is indeed necessary to develop a custom parser that converts the CSV files into CloudSim's internal data structures (such as Hosts, Cloudlets, etc.) [71]. Solving format compatibility usually requires extensive preprocessing: writing scripts to parse trace files, extracting necessary fields, and generating simulation events in chronological order. Differences in field semantics across traces also pose compatibility challenges. For example, the Google trace uses Normalized CPU to indicate usage, requiring one to assume a total CPU value and then convert it back to an absolute value; or, Azure trace may lack fine-grained task logs and only provide statistics at the VM level, so if the simulator requires task-level events, the information is missing. Overall, the lack of unified standards is a general problem—traces from different sources are difficult to simply plug into different simulators, and manual work is needed to bridge format differences.

Granularity Differences: The time advancement mechanism of the simulator and the time granularity recorded in the trace may not be consistent. If the trace records are too fine-grained, the number of simulation events increases dramatically, resulting in excessive overhead; conversely, if the records are too coarse-grained, critical dynamics may be lost. Traces at the Cloud/Edge layer typically record coarse-grained events (such as task scheduling or function calls), whereas IoT device traces are often fine-grained continuous sensor data. Studies have shown that there is a "semantic gap" between low-level sensor data and high-level business events, and abstraction is required to assign business meaning [44].

Semantic Gap: This refers to the mismatch between the real system behaviors reflected in the trace and the models within the simulator. Sometimes, concepts present in the trace do not exist at all in the simulator. For example, the Google trace involves multi-priority task scheduling, preemptive scheduling, and jobs composed of multiple tasks, but early versions of CloudSim did not have corresponding mechanisms [84]. Another example is that the trace may contain container or microservice scaling, while the simulator only supports VM abstraction. This semantic mismatch means that even if trace events are forcibly replayed, the simulation results may be distorted, as the simulator cannot replicate the real system's decision logic at that time.

Performance Bottlenecks and Scalability Issues: Trace-driven simulation brings a sharp increase in scale and complexity, making it prone to performance bottlenecks. In real environments, the number of IoT and edge devices is enormous, continuously generating streams of logs (high throughput, high frequency), and feeding these into a simulator often leads to I/O bottlenecks and excessive memory consumption [4]. Sequential execution of a large number of events may result in prolonged simulation runtimes or even memory exhaustion. Simulators such as early versions of CloudSim were originally designed for relatively small-scale scenarios and thus have limited scalability.

6.4 Future Research Directions and Prospects

Observations:

O-15: Promoting trace standardization and openness can significantly reduce integration costs and enhance reproducibility.

O-16: Digital twins and automated abstraction extraction—including AI-generated workloads and alignment validation—are key directions for improving simulation timeliness and representativeness.

Based on the current state of the field, there are several directions worthy of exploration and improvement regarding the integration of traces and cloud/edge simulators:

Standardization and Openness of Trace Formats: There is currently a lack of unified trace format standards, resulting in frequent format conversion in research. In the future, both industry and academia should work together to establish a universal cloud workload trace schema (similar to open telemetry standards). At the same time, cloud service providers should be encouraged to make more diverse traces public (covering traditional VMs, containers, serverless, edge devices, etc.). Through standardization, simulator developers can directly support standard formats, reducing repetitive effort. Opening more traces can also address gaps in current coverage, such as traces of edge IoT device interactions, 5G network slicing traces, etc., providing research data for these new scenarios.

Digital Twin: Digital twin is considered an important future direction for the simulation of complex systems. Its core is to build a real-time, data-driven virtual replica of the physical system, allowing the model to continuously update itself according to streaming trace data [20]. Through such model adaptation, the twin can automatically adjust its state or parameters as new sensor data arrives, thus maintaining long-term synchronization with the real object. Some preliminary explorations have already demonstrated the value of digital twins in the IoT domain: for example, in the TWIN-ADAPT framework proposed by Gupta et al., a continuous learning mechanism was introduced to dynamically adjust its anomaly detection model in real time for concept drift in IoT data streams [28].

Automated Trace Analysis and Abstraction Extraction: Facing massive volumes of trace data, manual analysis and conversion are inefficient. In the future, AI technology can be used to automate the extraction of simulation models from traces. Some works on TechRxiv have already attempted to use deep learning to predict trace workloads, and in the future, this can be extended to automatically generate simulation scenarios [85]. With the help of AI, it is even possible to generate workloads that do not exist but could potentially appear for prospective research. It is necessary to ensure that these automatically generated models still retain the key characteristics of the original traces and do not distort the underlying semantics.

Higher Scalability : As the scale of production systems continues to grow, the volume of trace data is constantly increasing, and simulators need to possess orders-of-magnitude scalability, such as supporting the simulation of millions of tasks and hundreds of millions of events. Currently, some parallel discrete-event simulation frameworks (such as MPI-based ROSS) have already achieved ultra-large-scale simulation in the HPC field [13]. These technical approaches can serve as references for the scalability design of future cloud and edge computing simulators.

Acquisition and Utilization of Edge and IoT Scenario Traces: Compared with cloud datacenter traces, traces in the edge and IoT domains are currently scarce. In the future, efforts should be made to strengthen the collection of such traces (for example, operational logs from large-scale IoT sensor networks, task records from edge computing nodes, vehicular network traces, etc.) and to expand simulators to support these traces. For instance, geographic information processing modules could be introduced to support city-level simulations, and physical environment simulation (such as climate and power) could be combined with IoT trace usage. Edge simulators should also take privacy issues into account, as many IoT traces involve personal data; how to protect privacy in simulation is also a major topic.

7 Conclusion

This survey reviews the simulators and workload trace resources in the field of compute continuum, proposes a unified classification method, and conducts a comprehensive analysis of existing tools, datasets, and their compatibility. The study finds that although simulation platforms and workload traces are essential for evaluating and optimizing continuum systems, there is still a lack of holistic surveys and analyses. Most simulators are mainly applied to a single layer or specific domain, and the formats and integration methods of traces lack standardization, making reproducible and highly reliable evaluation challenging. We also identify the insufficiency of publicly available edge and cloud-edge collaborative trace data, as well as key technical challenges such as format incompatibility, semantic mismatch, and scalability bottlenecks.

Regarding future development directions, relevant research should promote the standardization and open sharing of trace formats, develop digital twin-based real-time trace-driven simulation methods, and advance the automation of trace analysis and abstraction extraction. At the same time, expanding the collection and utilization of edge and IoT trace data is of great significance for supporting emerging applications and ensuring the practical applicability of simulations. We believe that addressing these issues will help the field move toward more realistic, scalable, and reproducible simulation-based research. This survey aims to provide practical references for researchers and engineering practitioners, assist them in selecting and combining appropriate tools, and offer suggestions for future development.

References

[1] Al-Dulaimy, A., Jansen, M., Johansson, B., Trivedi, A., Iosup, A., Ashjaei, M., Galletta, A., Kimovski, D., Prodan, R., Tserpes, K., Kousiouris, G., Giannakos, C., Brandic, I., Ali, N., Bondi, A. B., and Papadopoulos, A. V. The computing continuum: From iot to the cloud. *Internet of Things 27* (2024), 101272.

[2] Alam, M., Shakil, K. A., and Sethi, S. Analysis and clustering of workload in google cluster trace based on resource usage. In *2016 IEEE Intl conference on computational science and engineering (CSE) and IEEE Intl conference on embedded and ubiquitous computing (EUC) and 15th Intl symposium on distributed computing and applications for business engineering (DCABES)* (2016), IEEE, pp. 740–747.

[3] Alibaba. Alibaba cluster trace program. https://github.com/alibaba/clusterdata.

[4] Almutairi, R., Bergami, G., and Morgan, G. Advancements and challenges in iot simulators: A comprehensive review. *Sensors 24*, 5 (2024), 1511.

[5] Alwasel, K., Jha, D. N., Habeeb, F., Demirbaga, U., Rana, O., Baker, T., Dustdar, S., Villari, M., James, P., Solaiman, E., et al. Iotsim-osmosis: A framework for modeling and simulating iot applications over an edge-cloud continuum. *Journal of Systems Architecture 116* (2021), 101956.

[6] Amvrosiadis, G., Park, J. W., Ganger, G. R., Gibson, G. A., Baseman, E., and DeBardeleben, N. On the diversity of cluster workloads and its impact on research results. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (2018), pp. 533–546.

[7] Andreoli, R., Zhao, J., Cucinotta, T., and Buyya, R. Cloudsim 7g: An integrated toolkit for modeling and simulation of future generation cloud computing environments. *Software: Practice and Experience 55*, 6 (2025), 1041–1058.

[8] Bahga, A., Madisetti, V. K., et al. Synthetic workload generation for cloud computing applications. *Journal of Software Engineering and Applications 4*, 07 (2011), 396.

[9] Beloglazov, A., Abawajy, J., and Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems 28*, 5 (2012), 755–768. Special Section: Energy efficiency in large-scale distributed systems.

[10] Bux, M., and Leser, U. Dynamiccloudsim: Simulating heterogeneity in computational clouds. In *Proceedings of the 2nd acm sigmod workshop on scalable workflow execution engines and technologies* (2013), pp. 1–12.

[11] Byrne, J., Svorobej, S., Gourinovitch, A., Elango, D. M., Liston, P., Byrne, P. J., and Lynn, T. Recap simulator: Simulation of cloud/edge/fog computing scenarios. In *2017 Winter Simulation Conference (WSC)* (2017), IEEE, pp. 4568–4569.

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

[13] Carothers, C. D., Bauer, D., and Pearce, S. Ross: A high-performance, low-memory, modular time warp system. *Journal of parallel and distributed computing 62*, 11 (2002), 1648–1669.

[14] Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid. *Parallel Computing 123* (2025), 103–125.

[15] Casanova, H., Legrand, A., and Quinson, M. Simgrid: A generic framework for large-scale distributed experiments. In *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)* (2008), IEEE, pp. 126–131.

[16] Chu, X., Hofstätter, D., Ilager, S., Talluri, S., Kampert, D., Podareanu, D., Duplyakin, D., Brandic, I., and Iosup, A. Generic and ml workloads in an hpc datacenter: Node energy, job failures, and node-job analysis. In *2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS)* (2024), IEEE, pp. 710–719.

[17] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (2017), pp. 153–167.

[18] Del-Pozo-Puñal, E., García-Carballeira, F., and Camarmas-Alonso, D. A scalable simulator for cloud, fog and edge computing platforms with mobility support. *Future Generation Computer Systems 144* (2023), 117–130.

[19] Di, S., and Cappello, F. Gloudsim: Google trace based cloud simulator with virtual machines. *Software: Practice and Experience 45*, 11 (2015), 1571–1590.

[20] Digital Twin Consortium. Digital twin consortium defines digital twin. https://www.digitaltwinconsortium.org/2020/12/digital-twin-consortium-defines-digital-twin/, Dec. 3 2020.

[21] Fakhfakh, F., Kacem, H. H., and Kacem, A. H. Simulation tools for cloud computing: A survey and comparative study. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)* (2017), IEEE, pp. 221–226.

[22] Fernández-Cerero, D., Jakóbik, A., Fernández-Montes, A., and Kołodziej, J. Game-score: Game-based energy-aware cloud scheduler and simulator for computational clouds. *Simulation Modelling Practice and Theory 93* (2019), 3–20.

[23] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., and Janicke, H. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. *IEEe Access 10* (2022), 40281–40306.

[24] Ganapathi, A., Chen, Y., Fox, A., Katz, R., and Patterson, D. Statistics-driven workload modeling for the cloud. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)* (2010), pp. 87–92.

[25] Gkonis, P., Giannopoulos, A., Trakadas, P., Masip-Bruin, X., and D'Andria, F. A survey on iot-edge-cloud continuum systems: Status, challenges, use cases, and open issues. *Future Internet 15*, 12 (2023), 383.

[26] Gomez, J., Kfoury, E. F., Crichigno, J., and Srivastava, G. A survey on network simulators, emulators, and testbeds used for research and education. *Computer Networks 237* (2023), 110054.

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

[28] Gupta, R., Tian, B., Wang, Y., and Nahrstedt, K. Twin-adapt: continuous learning for digital twin-enabled online anomaly classification in iot-driven smart labs. *Future Internet 16*, 7 (2024), 239.

[29] Jeon, M., Venkataraman, S., Qian, J., Phanishayee, A., Xiao, W., and Yang, F. Multi-tenant gpu clusters for deep learning workloads: Analysis and implications. *Technical report, Microsoft Research* (2018).

[30] Kavulya, S., Tan, J., Gandhi, R., and Narasimhan, P. An analysis of traces from a production mapreduce cluster. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (2010), IEEE, pp. 94–103.

[31] Kecskemeti, G. Dissect-cf: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory 58* (2015), 188–218.

[32] Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., and Ahmed, A. Edge computing: A survey. *Future Generation Computer Systems 97* (2019), 219–235.

[33] Kliazovich, D., Bouvry, P., and Khan, S. U. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing 62*, 3 (2012), 1263–1283.

[34] Kolevski, D., and Michael, K. Edge computing and iot data breaches: Security, privacy, trust, and regulation. *IEEE Technology and Society Magazine 43*, 1 (2024), 22–32.

[35] Kolosov, O., Yadgar, G., Maheshwari, S., and Soljanin, E. Benchmarking in the dark: On the absence of comprehensive edge datasets. In *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)* (2020).

[36] Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems 100* (2019), 779–796.

[37] Lantz, B., Heller, B., and McKeown, N. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (2010), pp. 1–6.

[38] Lata, S., and Singh, D. Cloud simulation tools: A survey. In *AIP Conference Proceedings* (2022), vol. 2555, AIP Publishing.

[39] Lera, I., Guerrero, C., and Juiz, C. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access 7* (2019), 91745–91758.

[40] Lilhore, U. K., Simaiya, S., Sharma, Y. K., Rai, A. K., Padmaja, S., Nabilal, K. V., Kumar, V., Alroobaea, R., and Alsufyani, H. Cloud-edge hybrid deep learning framework for scalable iot resource optimization. *Journal of Cloud Computing 14*, 1 (2025), 5.

[41] Liu, G., Lin, W., Zhang, H., Lin, J., Peng, S., and Li, K. Public datasets for cloud computing: A comprehensive survey. *ACM Computing Surveys 57*, 8 (2025), 1–38.

[42] Lopes, M. M., Higashino, W. A., Capretz, M. A., and Bittencourt, L. F. Myifogsim: A simulator for virtual machine migration in fog computing. In *Companion proceedings of the10th international conference on utility and cloud computing* (2017), pp. 47–52.

[43] Mangas, A. G., Alonso, F. J. S., Martínez, D. F. G., and Díaz, F. D. Wotemu: An emulation framework for edge computing architectures based on the web of things. *Computer Networks 209* (2022), 108868.

[44] Mangler, J., Seiger, R., Benzin, J.-V., Grüger, J., Kirikkayis, Y., Gallik, F., Malburg, L., Ehrendorfer, M., Bertrand, Y., Franceschetti, M., et al. From internet of things data to business processes: Challenges and a framework. *arXiv preprint arXiv:2405.08528* (2024).

[45] Mastenbroek, F., Andreadis, G., Jounaid, S., Lai, W., Burley, J., Bosch, J., Van Eyk, E., Versluis, L., Van Beek, V., and Iosup, A. 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)* (2021), IEEE, pp. 455–464.

[46] Matloff, N. Introduction to discrete-event simulation and the simpy language. *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August 2*, 2009 (2008), 1–33.

[47] Mayer, R., Graser, L., Gupta, H., Saurez, E., and Ramachandran, U. Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures. In *2017 IEEE Fog World Congress (FWC)* (2017), IEEE, pp. 1–6.

[48] McKerracher, J., Mukherjee, P., Kalyanam, R., and Bagchi, S. Fresco: A public multi-institutional dataset for understanding hpc system behavior and dependability. In *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration*. 2025, pp. 1–6.

[49] Mechalikh, C., Safavifar, Z., Golpayegani, F., et al. Quality matters: A comprehensive comparative study of edge computing simulators. *Simulation Modelling Practice and Theory 138* (2025), 103042.

[50] Meskar, E., and Liang, B. Magiks: Fair multi-resource allocation game induced by kalai-smorodinsky bargaining solution. *IEEE Open Journal of the Communications Society 3* (2022), 797–810.

[51] Microsoft Azure. Azurepublicdataset. https://github.com/Azure/AzurePublicDataset, 2020.

[52] Moustafa, N. A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society 72* (2021), 102994.

[53] Nardini, G., Sabella, D., Stea, G., Thakkar, P., and Virdis, A. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access 8* (2020), 181176–181191.

[54] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., and Ghorbani, A. A. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors 23*, 13 (2023), 5941.

[55] Núñez, A., Vázquez-Poletti, J. L., Caminero, A. C., Castañé, G. G., Carretero, J., and Llorente, I. M. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing 10*, 1 (2012), 185–209.

[56] Pati, A., Parhi, M., and Pattanayak, B. K. Iot-fog-edge-cloud computing simulation tools a systematic review. *International Journal of Smart Sensor and Adhoc Network 3*, 2 (2022).

[57] Paul, A. K., Choi, J. Y., Karimi, A. M., and Wang, F. Machine learning assisted hpc workload trace generation for leadership scale storage systems. In *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing* (2022), pp. 199–212.

[58] Peyman, M., Copado, P. J., Tordecilla, R. D., Martins, L. d. C., Xhafa, F., and Juan, A. A. Edge computing and iot analytics for agile optimization in intelligent transportation systems. *Energies 14*, 19 (2021), 6309.

[59] Plus, C. Googletaskusagetracereader.java, 2025. Accessed: 2025-07-31.

[60] Pucher, A., Gul, E., Wolski, R., and Krintz, C. Using trustworthy simulation to engineer cloud schedulers. In *2015 IEEE International Conference on Cloud Engineering* (2015), IEEE, pp. 256–265.

[61] Qayyum, T., Malik, A. W., Khattak, M. A. K., Khalid, O., and Khan, S. U. Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access 6* (2018), 63570–63583.

[62] Raith, P., Rausch, T., Furutanpey, A., and Dustdar, S. faassim: A trace-driven simulation framework for serverless edge computing platforms. *Software: Practice and Experience 53*, 12 (2023), 2327–2361.

[63] Reiss, C., Wilkes, J., and Hellerstein, J. L. Google clusterusage traces: format+ schema. *Google Inc., White Paper 1* (2011), 1–14.

[64] Robinson, S. Exploring the relationship between simulation model accuracy and complexity. *Journal of the Operational Research Society 74*, 9 (2023), 1992–2011.

[65] Rosendo, D., Costan, A., Valduriez, P., and Antoniu, G. Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. *Journal of Parallel and Distributed Computing 166* (2022), 71–94.

[66] Rosendo, D., Mattoso, M., Costan, A., Souza, R., Pina, D., Valduriez, P., and Antoniu, G. Provlight: Efficient workflow provenance capture on the edge-to-cloud continuum. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)* (2023), IEEE, pp. 221–233.

[67] Sabnis, A., and Sitaraman, R. K. Tragen: a synthetic trace generator for realistic cache simulations. In *Proceedings of the 21st ACM Internet Measurement Conference* (2021), pp. 366–379.

[68] Samani, Z. N., Mehran, N., Kimovski, D., Hammer, J., and Prodan, R. Edge infrastructure traces, Nov. 2022.

[69] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (2011), pp. 1–14.

[70] Sibai, F. N., Asaduzzaman, A., and El-Moursy, A. Characterization and machine learning classification of ai and pc workloads. *IEEE Access 12* (2024), 83858–83875.

[71] Silva Filho, M. C., Oliveira, R. L., Monteiro, C. C., Inácio, P. R., and Freire, M. M. 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)* (2017), IEEE, pp. 400–406.

[72] Sonmez, C., Ozgovde, A., and Ersoy, C. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies 29*, 11 (2018), e3493.

[73] Sridharan, S., Heo, T., Feng, L., Wang, Z., Bergeron, M., Fu, W., Zheng, S., Coutinho, B., Rashidi, S., Man, C., et al. Chakra: Advancing performance benchmarking and co-design using standardized execution traces. *arXiv preprint arXiv:2305.14516* (2023).

[74] Tanaka, R., Papadimitriou, G., Viswanath, S. C., Wang, C., Lyons, E., Thareja, K., Qu, C., Esquivel, A., Deelman, E., Mandal, A., et al. Automating edge-to-cloud workflows for science: Traversing the edge-to-cloud continuum with pegasus. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (2022), IEEE, pp. 826–833.

[75] Teixeira Sá, T., Calheiros, R. N., and Gomes, D. G. Cloudreports: An extensible simulation tool for energy-aware cloud computing environments. In *Cloud computing: Challenges, limitations and R&D solutions*. Springer, 2014, pp. 127–142.

[76] Toczé, K., Schmitt, N., Kargén, U., Aral, A., and Brandić, I. Edge workload trace gathering and analysis for benchmarking. In *2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC)* (2022), IEEE, pp. 34–41.

[77] Van Le, T., Ioini, N. E., Pahl, C., and Barzegar, H. R. Edge computing simulation platforms: A technology survey. In *Advances in Service-Oriented and Cloud Computing: International Workshops of ESOCC 2020, Heraklion, Crete, Greece, September 28–30, 2020, Revised Selected Papers 8* (2021), Springer, pp. 18–28.

[78] Wen, S., Han, R., Qiu, K., Ma, X., Li, Z., Deng, H., and Liu, C. H. K8ssim: A simulation tool for kubernetes schedulers and its applications in scheduling algorithm optimization. *Micromachines 14*, 3 (2023), 651.

[79] Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., and Ding, Y. {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)* (2022), pp. 945–960.

[80] Weng, Q., Yang, L., Yu, Y., Wang, W., Tang, X., Yang, G., and Zhang, L. Beware of fragmentation: Scheduling {GPU-Sharing} workloads with fragmentation gradient descent. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)* (2023), pp. 995–1008.

[81] Wette, P., Dräxler, M., Schwabe, A., Wallaschek, F., Zahraee, M. H., and Karl, H. Maxinet: Distributed emulation of software-defined networks. In *2014 IFIP Networking Conference* (2014), IEEE, pp. 1–9.

[82] Wickremasinghe, B., Calheiros, R. N., and Buyya, R. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *2010 24th IEEE international conference on advanced information networking and applications* (2010), IEEE, pp. 446–452.

[83] Yang, L., Wang, Y., Yu, Y., Weng, Q., Dong, J., Liu, K., Zhang, C., Zi, Y., Li, H., Zhang, Z., et al. {GPU-Disaggregated} serving for deep learning recommendation models at scale. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)* (2025), pp. 847–863.

[84] Zakarya, M., Gillam, L., Khan, A. A., and Rahman, I. U. Perficientcloudsim: a tool to simulate large-scale computation in heterogeneous clouds. *The Journal of Supercomputing* (2020).

[85] Zhu, Y., Zhang, W., Chen, Y., and Gao, H. A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking 2019*, 1 (2019), 274.

Notes