The SPEC-RG Reference Architecture for The Compute Continuum

Matthijs Jansen
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.s.jansen@vu.nl

Auday Al-Dulaimy
Mälardalen University
Västerås, Sweden
auday.aldulaimy@mdu.se

Alessandro V. Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdu.se

Animesh Trivedi
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.trivedi@vu.nl

Alexandru Iosup
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.iosup@vu.nl

***Abstract*—As the next generation of diverse workloads like autonomous driving and augmented/virtual reality evolves, computation is shifting from cloud-based services to the edge, leading to the emergence of a cloud-edge *compute continuum*. This continuum promises a wide spectrum of deployment opportunities for workloads that can leverage the strengths of cloud (scalable infrastructure, high reliability) and edge (energy efficient, low latencies). Despite its promises, the continuum has only been studied in *silos* of various computing models, thus lacking strong end-to-end theoretical and engineering foundations for computing and resource management across the continuum. Consequently, developers resort to ad hoc approaches to reason about performance and resource utilization of workloads in the continuum. In this work, we conduct a first-of-its-kind systematic study of various computing models, identify salient properties, and make a case to unify them under a *compute continuum reference architecture*. This architecture provides an end-to-end analysis framework for developers to reason about resource management, workload distribution, and performance analysis. We demonstrate the utility of the reference architecture by analyzing two popular continuum workloads, deep learning and industrial IoT. We have developed an accompanying deployment and benchmarking framework and first-order analytical model for quantitative reasoning of continuum workloads. The framework is open-sourced and available at https://github.com/atlarge-research/continuum.**

***Index Terms*—Compute continuum, reference architecture, edge computing, resource management, offloading, benchmark**

I. Introduction

Cloud computing is one of today's most successful computing paradigms [1]. Cloud developers can summon a large fleet of servers and infrastructure services (storage, network, resource managers), and deploy complex, scalable workloads on them with a few clicks [2], [3]. Traditionally, workloads are executed in the cloud or in a limited capacity at the user, located on resource-constrained endpoint devices at the far end of the network such as sensor nodes, smart devices, and IoT devices [4]. To enable new generations of workloads with strict performance and privacy requirements, the cloud-centric view of computation is shifting outwards towards the *edge*, close to users. Edge computing offers low latency, energy-efficient data processing by processing data in-the-field, close to where it is generated using decentralized, heterogeneous, and mobile computing devices often with limited resources. Domains with edge workloads include the Internet of Things (IoT) [5], self-driving vehicles [6], smart farming [7], smart industry [8], mobile gaming [9], analytics [10], and machine learning [11]. With edge computing connecting cloud and user, workloads previously deployed as cloud-only or endpoint-only are now distributed in compute, data, and state and across a compute continuum [12] of cloud, edge, and endpoint devices, leveraging the best of both worlds: the high-performance, scalable network-storage infrastructure and high reliability of clouds with low-latency, privacy-preserving computation of edge. Figure 1 provides an overview of this continuum.

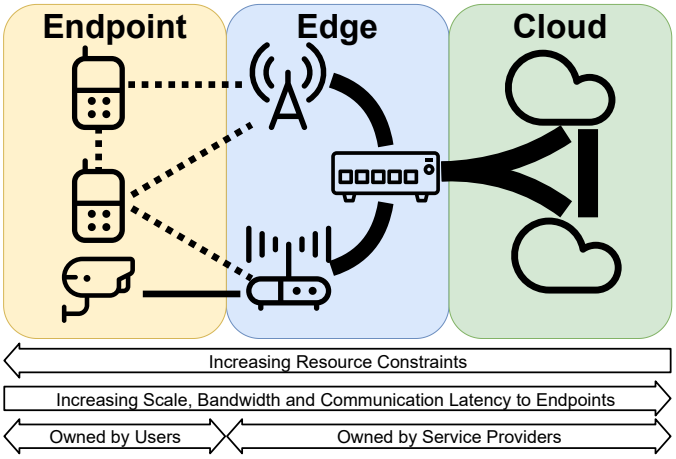

Fig. 1. An overview of the compute continuum (key properties shown as arrows at the bottom) with endpoints, edge servers, and cloud infrastructure.

Though promising, the continuum also presents unique challenges to workload developers and infrastructure providers. Unlike clouds, edge computing lacks standardization of development guidelines and foundational infrastructure services like resource managers, scalable storage, or automatic workflow managers that help with workload deployments [13]–[16]. Hence, developers must decide by themselves how to manage

Key insight: Existing computing models for task offloading are often presented in isolation, while there is significant overlap in concerns addressed by these models. By considering cloud, edge, and endpoint computing models as part of a unified, continuous computing model—the compute continuum—developers are no longer limited to the single set of constraints from current isolated computing models.

resources, split workload, and offload computation with tasks (either in parts or as a whole) that traditionally run in the cloud to edge servers and endpoints [17]. However, making such decisions is non-trivial as there is a large design space of choices. For example, task offloading includes, but is not limited to, offloading from cloud to edge [18], offloading from edge to cloud [19], compute management among different edge devices [20], and compute management between edge and endpoint devices [17].

Computing models like fog, mist, or mobile cloud computing typically address these choices by capturing pertinent workload developments and deployment guidelines for a specific offloading model. As we will illustrate in §II, such fragmented views lead to *silos* of knowledge and miss an opportunity to develop an *end-to-end* reasoning framework regarding workload splitting, infrastructure services, task offloading, and resource management. A further consequence of such fragmentation is the lack of any deployment, analysis, and benchmarking framework with strong theoretical and engineering foundations that can help developers assess their decisions' quality (§VI). As a result, continuum workload developers make development and deployment decisions ad hoc, thus creating complexity and a general confusion regarding *what is the compute continuum* and *what should a developer or infrastructure provider know about it before developing or supporting compute continuum-ready applications*.

In this paper, we take a step back and systematically study the evolution of the compute continuum and various associated computing models. Our study of 17 compute continuum models reveals the *primary insight* that even though these computing models are often presented in isolation, there is a significant overlap in terms of concerns addressed, used mechanisms, and application domains addressed by these different computing models. As a result, we synthesize common characteristics addressed by these models and select five representative models that each cover a distinct part of the compute continuum design space (Table I): Mist computing [21], edge computing [22], multi-access edge computing [23], fog computing [24], and mobile cloud computing [25]. Thus, we make a case that developers and infrastructure providers should not consider cloud, edge, and endpoint computing models in isolation but as parts of a unified, *continuous computing model*, captured as the SPEC-RG compute continuum reference architecture[1]. The unified reference architecture identifies the key building blocks of any continuum deployment and associated key concerns. We map the computing models and two related use cases (deep learning and industrial IoT) to this reference architecture to demonstrate its completeness, comprehensiveness, and usefulness.

The reference architecture also gives us a blueprint to design and implement a workload deployment and benchmarking framework by defining compute continuum components and their responsibility split. The framework leverages virtual machines (KVM) and container technologies (Kubernetes) and allows a developer to configure various typical network (bandwidth, latency, jitter, packet drops), storage (capacity, bandwidth), and compute configurations (CPU type, speed, number of cores, memory) found in the compute continuum. Supporting these configurations with any number of machines allows a developer to quickly explore the performance landscape of various compute continuum deployments in only a few lines of code and collect performance statistics to analyze whether a particular offloading model is suitable for their needs. The performance reports from the deployment framework are verified using a first-order analytical framework. The unique combination of a conceptual model (the Reference Architecture) with practical (deployment and benchmarking framework) and analytical (first-order model) frameworks enables developers to reason about their compute continuum workloads *with* strong theoretical and engineering foundations — a feat that is not possible today.

Our key contributions in this paper include:

1) (*survey*) To the best of our knowledge, we present the most comprehensive survey on computing models with 17 models, synthesize their salient properties, and identify opportunities for unification (§II).
2) (*conceptual*) We make a case for the compute continuum and propose a design of a unified reference architecture. Our reference architecture is the first to consider the *entire edge-cloud compute continuum*. We use this new architecture to synthesize two domain-specific architectures for deep learning and industrial IoT (§III).
3) (*experimental*) To aid compute continuum developers, we present an open-source workload deployment and benchmarking framework. We show how developers can explore the large continuum design space in a few lines of code to examine various trade-offs (§IV).
4) (*analytical*) We enhance and verify the performance analysis capabilities of the framework by formulating an analytical performance model for exploring workload deployment scenarios in the continuum (§V).

[1]Established in 2011, the Cloud Group of the SPEC Research Group focuses on general and specific performance issues associated with cloud operation, from traditional to new performance metrics, from workload characterization to modeling, from concepts to tools, from performance measurement processes to benchmarks. The work presented here is part of a larger, long-term activity within this group, focusing on understanding the systems principles of cloud and edge computing. The activity has started in 2019 and has resulted in several publications, which are available online with open-source artifacts. The group agrees with this article's publication under the current co-authorship.

TABLE I
COMPARISON OF KEY CHARACTERISTICS OF 5 COMPUTING MODELS FOR TASK OFFLOADING BETWEEN CLOUD, EDGE, AND ENDPOINT. MC: MIST COMPUTING; EC: EDGE COMPUTING; MEC: MULTI-ACCESS EDGE COMPUTING; FC: FOG COMPUTING; MCC: MOBILE CLOUD COMPUTING.

Key Characteristic (KC)	MC	EC	MEC	FC	MCC
KC1: Compute source	Endpoint	Endpoint	Cloud	Cloud	Endpoint
KC2: Data source	Endpoint	Cloud, endpoint	Endpoint	Cloud, endpoint	Endpoint
KC3: Offload target	Endpoint	Edge	Edge	Edge	Cloud
KC4: Architecture	Peer-to-peer	Distributed	Distributed	Hierarchical	Distributed
KC5: Offload service	Compute, storage	Compute	Compute	Compute, storage	Compute, storage
KC6: Compute capacity	Low	Moderate	Moderate	Moderate to high	High
KC7: Network latency	Low	Low	Low	Low to moderate	High
KC8: Network type	Wired, wireless	Wired, wireless	Wireless	Wired, wireless	Wired
KC9: Operator	User	User, cloud provider	Network provider	Cloud provider	Cloud provider
Popular use case	Peer-to-peer IoT processing	Real-time data processing	Mobile, network-aware data processing	Low latency cloud service provisioning	Compute- and storage-intensive tasks

II. A CASE FOR THE UNIFIED CONTINUUM MODEL

Due to the historical and diverse nature of outside-the-cloud computing environments, many detailed but selective computing models have been developed to guide developers and infrastructure providers in deploying and supporting their workload. Each model has its assumptions on available infrastructure and workload demands, and therefore, finding the right computing model for an application deployment requires a careful analysis of characteristics desired by users and offered by computing models. In this section, we first synthesize key characteristics of interest to developers and infrastructure providers, then list how various computing models deliver these properties, and finally, make a case for unifying end-to-end concerns. Past survey works in this field also analyze these models but fall short of providing an end-to-end unification argument [26]–[29].

A. Synthesizing Key Characteristics

We start by synthesizing and identifying 9 unique key characteristics (KC) that a compute continuum developer or infrastructure provider should know about (Table I). Based on the computing model and the workload, some or all of these key characteristics can be of interest to a developer or provider.

KC1. Compute source: shows where requests for computing are generated. The source could be cloud, edge, or endpoint, and is different from the offload target.

KC2. Data source: determines where the data is generated or stored before offloading.

KC3. Offload target: identifies where computing operations and data are offloaded to in the continuum.

KC4. Architecture: determines the structure of devices participating in offloading. The options are P2P, distributed (a cluster of devices in a single layer of the continuum), or hierarchical with multiple sub-layers.

KC5. Offload service: captures the type of services being offloaded, e.g., computing requests or data.

KC6. Compute capacity: captures the compute capacity of the offload target. In a typical setup, endpoint devices have low energy, power-efficient CPUs (e.g., mobile phones, cameras), edge servers have moderate, and cloud servers are the most powerful machines.

KC7. Network latency: determines how efficiently entities involved in the continuum can communicate with each other, measured from data source to offload target.

KC8. Network type: can be wired (Ethernet, USB) or wireless (WiFi, Bluetooth, cellular, WANs). The network type also determines what kind of mobility a developer can expect in the deployment.

KC9. Operator: identifies the stakeholder running the infrastructures and development of workloads.

B. Key Characteristics with Existing Computing Models

Having synthesized key characteristics, we now focus on existing computing models and how they consider the aforementioned characteristics. We study 17 unique computing models in our survey that leverage a combination of cloud, edge, and endpoint devices, and enable workload offloading between these devices. Based on the amount of overlap (if the model has been superseded or merged), and the current interest (is part of active academic/industrial interest), we narrow down the discussion to five specific models while giving a rationale for the process.

1. Mist Computing is concerned with forming a peer-to-peer (P2P) network of user-operated endpoint devices, where resource-constrained and/or busy endpoints can offload workload to nearby endpoints with more aggregate processing power or storage capacity [21]. Mist computing is a representative of a larger group of P2P computing models for endpoint devices such as mobile crowdsourcing [30], peer-to-peer computing [31], mobile crowd computing [32], mobile ad hoc cloud computing [33], and transparent computing [34], with each having its assumption on how the P2P network is formed, the goal of the offloading, and possible interaction with a centralized cloud or edge component.

2. Edge Computing enables the offloading of computation and related data from endpoints to a cluster of edge devices through low-latency networks to enhance previous endpoint-only applications. For applications that share data and state between tenants, like federated learning and online gaming, data can also flow from cloud to edge [35]. Related to edge computing is edge-centric computing, which focuses on applications with human interaction.

3. Multi-access Edge Computing: Previously known as mobile edge computing [36], MEC is concerned with augmenting cellular and wireless infrastructure (4G, 5G) with computing capacity to process user data from multiple endpoints close to users instead of in the cloud [23], [37]. Contrary to edge computing, where edge devices are owned by users or cloud providers, edge devices in MEC are exclusively owned by network providers and standardized under the European Telecommunications Standards Institute (ETSI).

4. Fog Computing extends cloud services to edge to accelerate former cloud-only applications by leveraging the edge's low latency to endpoints [5], [38]. Fog computing leverages multiple hierarchical edge clusters: From resource-constrained devices near users to micro data centers near the cloud, providing a trade-off between compute capacity and network latency. Offloading services to micro data centers is the focus of cloudlet computing, a precursor of fog computing [17].

5. Mobile Cloud Computing: Endpoints offload compute tasks and related data for workloads without strict latency constraints directly to the cloud to benefit from the cloud's cheaper and more stable resources [25]. Even selected latency-constrained applications can directly offload to the cloud as the average latency between cloud and endpoint is decreasing due to better networking infrastructure and more cloud data centers [39]. Mobile grid computing is a precursor of mobile cloud computing, leveraging grids instead of clouds [40]. In situ computing explicitly focuses on preprocessing data on endpoint devices before offloading to reduce network load [41]. Osmotic computing combines mobile cloud computing and edge computing, advocating for a dynamic and seamless offloading process to cloud or edge [22], [42].

C. A Case for the Unification

In the past sections, we have synthesized key characteristics (*what a developer should know*) and identified how existing computing models provide (or lack) guidelines for these key characteristics for compute continuum workloads. Based on this discussion, we make a case for the unification of different computing models based on three primary arguments. First, finding a suitable model now requires comparing requirements from a particular workload to the features provided by the five models (Table I) and selecting the best fit. This approach of considering each model in isolation offers a restricted view of the task offloading design space. For example, data processing applications may require computing services in the edge and storage services in the cloud [7]: Edge computing offers the former, and mobile cloud computing offers the latter, but no computing model explicitly offers these services combined. This shows that choosing an existing computing model with fixed characteristics for a particular workload deployment has major limitations that create *silos* of knowledge, and hence, confusion for users. Second, computing models often only focus on compute/task management-related responsibilities. However, in practice, data and runtime state management and resource allocation with multi-workload/tenant situations are equally important [13]. For example, mist computing gives guidelines regarding how to establish and push computation in a P2P network of endpoint devices but provides little guidance on how data sharing is done. Lastly, providing end-to-end data provenance and privacy properties is challenging without building an end-to-end view of the workload. For example, in multi-access edge computing, where network providers own a part of the infrastructure and resources (not the workload developers), it would necessitate cooperation with the providers to ensure a secure resource allocation and isolated execution, a property challenging even in cloud computing [43], [44].

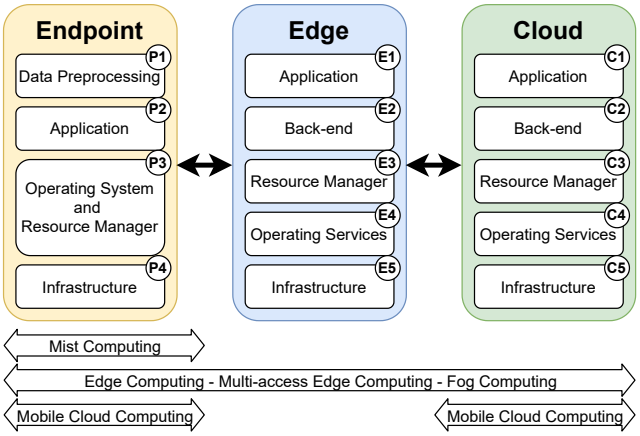

Fig. 2. Reference architecture for the compute continuum. The computing models are mapped to the parts of the architecture relevant to them.

To close the knowledge gaps, in this work, we propose to take an *ab initio* approach for designing a unified continuum computing model and an associated unified reference architecture in the next section.

III. The SPEC-RG Compute Continuum Reference Architecture

Having made a case for a unified computing model, we present our unified reference architecture for the compute continuum in Figure 2. A computing model typically has a reference architecture that describes the components a computing model operates on and the interactions between these components. All five selected models have reference architectures, with the most prominent architectures listed in §VI. The goal of building a unified reference architecture is to abstract away the specifications of the underlying hardware and peculiarities of specific workloads and discuss more broadly the foundational building blocks of the compute continuum regarding compute, data, and resource management.

A. Overview and Design Process

Overview: The SPEC-RG reference architecture consists of three tiers of systems: cloud, edge, and endpoint, with their associated components and responsibilities, as shown in Figure 2. The three-tier system is taken from the developer's view of compute offloading and data processing: endpoint devices are the last connected components in the architecture that create data streams for long-term storage and processing in cloud data centers, located at the top of the architecture,

or require data hosted in the cloud for content delivery networks [45] and applications with shared state [35]. In between, the edge represents multiple hops of processing and storage capabilities. In explaining the reference architecture, we will borrow well-defined cloud-centric components and extend them to include the expanded continuum. Lastly, we show the utility of the proposed reference architecture by *mapping* it into two concrete continuum workloads: machine learning and industrial IoT in §III-E.

Design process: The primary requirement from a reference architecture is that it should capture various deployment models that we discussed in the previous section. We start by analyzing and identifying commonalities and differences between the five selected computing models and use the overlap between the models as the basis of the architecture, with the differences between the models becoming implementation details of the components [28]. For example, all selected computing models (some explicitly, others implicitly) use resource managers to manage the complex stack of cloud, edge, and endpoint resources and the networks connecting them, but the specific implementation of these resource managers differs widely between workload deployments. As such, resource managers should be a core component of the architecture, while the implementation of the resource manager should be an implementation detail of the component. By creating a unified architecture for the edge continuum, we present a single platform for research into cloud, edge, and endpoint resources, unifying previous research efforts and allowing exploration in new research directions. We are the first to create such a unified end-to-end architecture, extending previous work which looked at computing models in isolation (§VI).

In the next sections, we discuss (i) the design of the architecture's components; (ii) the rationale behind their position in the reference architecture; and (iii) how to offer compute, data, and resource management services.

B. Endpoint Components

Endpoints are the last hop of processing and connectivity to users with the following key features: First, they often are data sources, e.g., user input through online gaming [9], environment sensing [7], or video surveillance [11]. Second, they are resource and energy-constrained, e.g., IoT devices such as sensors, and embedded systems [17]. Lastly, they may be mobile, e.g., smartphones, where location-sensitive services must migrate to the device's location [46]. To support the unified computing model, we have split up endpoint responsibilities into the following four components:

P1. Data Preprocessing: As data generators, we identify endpoints to be perfectly positioned to run early workload-specific data preprocessing logic such as filtering, tracking, aggregation, etc. This preprocessing aims to reduce the amount of data pushed to edge and cloud servers. Preprocessing capabilities can be built-in as a hardware accelerator or a software component available for use in high-level applications, thus freeing them to do low-level data preparations. For example, in video surveillance, camera endpoints can support built-in object tracking or face detection [47], as can be the case for self-driving vehicles [6]. The key decisions here for developers are: (1) does data need preprocessing; (2) can it be preprocessed at the endpoint itself, or must it be offloaded; (3) when and how to offload compute/data for preprocessing and collect results; and (4) can preprocessing be helped with hardware acceleration for energy efficiency.

P2. Application: Applications contain user-defined logic that runs on the endpoint to process incoming data (from edge/-cloud or post-preprocessing stage) and make decisions. Typical logic in the application can be running heuristics, making offloading decisions, monitoring performance metrics, and/or triggering changes in the application deployments. For example, an application can join a P2P network with other endpoints to do mist computing or display model data rendered by cloud servers in an online gaming setting (data flow from cloud to endpoint). The key decisions here for developers are: (1) in which direction compute and data offloading happens - P2P or hierarchically to edge/cloud; (2) when and how to offload compute/data for processing and collecting results; (3) what are favorable conditions for offloading and how to quantify them; and (4) how to reserve and allocate local (endpoint) and further (edge, cloud) resources.

P3. Operating System and Resource Manager: In contrast to edge/cloud servers, endpoints are typically single-tenant [48]; hence, OS-level resource managers (RMs) and multiplexers are often enough to meet the workload needs. Common examples of OS-level RMs are Android, iOS, TinyOS, and QNX [49]. These RMs are optimized for deployed hardware and scenarios such as the presence or absence of specific hardware features (virtualization, secure enclaves), energy management, high priority for user interactions, etc. The key decisions here for developers are: (1) what kind of resources does a workload need - CPU, memory, hardware accelerators, networking capacity; (2) how to reserve/allocate resources using the OS-level RMs; and (3) do RMs meet the functional (performance SLAs) and non-functional (cost, fault handling, data corruption) requirements of the workload?

P4. Infrastructure: Infrastructure includes all physical computing, memory, network, and storage resources available to the operating system. These resources can be general purpose (e.g., smartphones) or specialized for mobility, energy efficiency, etc. (e.g., IoT, embedded hardware). Unlike cloud data centers, the infrastructure at the endpoint can be owned and operated by application users and developers besides cloud providers. [5]. The key decisions here for developers/providers are: (1) what are the most popular workloads, computing models, and deployment modes leveraging the endpoints; (2) is there sufficient raw network, storage, and processing capabilities available; (3) what is the appropriate cost and scaling model (horizontal or vertical scaling); and (4) how to integrate and manage infrastructure in a unified continuum [50].

C. Edge Components

The reference architecture in Figure 2 shows that edge and cloud share the same high-level design. This similarity

is because both can run multi-tenant workloads on shared infrastructure and so need to offer similar services. However, uniquely at the edge, there should be support for application offloading both vertically (cloud to edge and back) and horizontally (from one edge system to another) [20]. The presence and absence of such capabilities determine what computing model one can use at the edge, e.g., edge clusters can not communicate among themselves with edge computing, while they can with fog computing. These unique capabilities are implemented in the edge components. To encompass these possibilities, we define these components for edge systems:

E1. Applications are at the first step from the endpoints, and hence they are in the best position to make decisions regarding placements, offloading, and scheduling of application components to meet workload-specific objectives. For example, applications can either do processing at the edge or preprocess data and offload more complex workloads to the cloud [19]. The key decisions here for developers are: (1) in which direction does data and compute offloading happen (cloud to edge, or vice versa); (2) what are the location, data movement, and mobility-related requirements; and (3) are back-end and operating services provided services enough to meet the workload demands.

E2. Back-end: Represents more general-purpose application execution frameworks such as TensorFlow Lite and WebAssembly runtime [51]. These back-end frameworks typically can manage application-specific memory, storage, communication, and workflows. The key decisions here for developers are: (1) does the back-end supports the desired offloading model; and (2) what kind of computing, storage, and networking resources does a back-end need.

E3. Resource Manager: Manages an edge system's application-independent *physical and virtual resources* (such as virtual machines and containers). These resource managers can be local or distributed, and their architecture determines what computing model one can run on the infrastructure. For example, KubeEdge is a hierarchical edge resource manager for Kubernetes, which does two levels of independent allocation for cloud- and edge-related resources. The key decisions here for developers/providers are: (1) what execution environment does an application want; (2) what are the scalability requirements; (3) what are the environment allocation overheads; and (4) what are data privacy and security level restriction in the scheduling of resources.

E4. Operating Services: Provides support to build distributed applications, and their responsibilities include (but are not limited to) communication [52], metadata management [53], consensus services [54], monitoring [55], storage services [56], etc. The key decisions here for developers/providers are: (1) is there an operating service for different kinds of communication and coordination patterns; (2) are services protected and isolated from tenants; (3) how do services handle resource allocation for themselves; and (4) what are scalability, performance, and fault-tolerance requirements for such services.

E5. Infrastructure: Similar to endpoints, compute, memory, and storage resources are provided to the resource manager. However, unlike endpoints, resources are split into physical and virtual resources. Bare-metal deployments give users direct access to hardware, while virtualization technologies like virtual machines and containers abstract physical resources away to provide more flexibility and security for a slight performance penalty. The key decisions for developers/providers are: (1) how to split (or multiplex) software and hardware resources securely between tenants; and (2) is there enough cooling, energy, and infrastructure to meet the demands.

D. Cloud Components

We identify two critical roles for cloud in the compute continuum: The first is that of a central controller that manages multiple compute continuum resources and schedules workload on them [55], [57]. The controller has a global overview of the endpoint, edge, and cloud resources. Developers upload continuum workloads to the cloud, which the cloud controller can schedule on specific edge systems. The global overview allows the central controller to schedule work more efficiently than a decentralized approach where each edge system manages itself. However, maintaining a global overview and scheduling workload from the cloud adds extra overhead to the offloading process [18]. The second role is that of an offloading target that uses cloud computing to provide much more computing and storage capacity than edge [58]. Resources in the edge are limited [59], so cloud computing can be used to run resource-intensive applications like deep learning instead. However, applications that require low end-to-end latency can not leverage the cloud due to the increased communication latency compared to the edge [12]. The role of offloading target in the continuum is key to both edge and cloud; therefore, they share the same high-level design in our architecture. The components from edge, described in §III-C, can also be applied to cloud. Liu et al. already provide a reference architecture for cloud computing [60].

E. Domain-specific Architectures

We create architectures for deep learning (Figure 3) and industrial IoT (IIoT, Figure 4) use cases and provide example systems for each component to demonstrate how the unified reference architecture can be used to explore design trade-offs for application deployments in the compute continuum.

Deep Learning: An important trend for deep learning in the continuum is that model training and inference tasks are split across multiple tiers of devices, as deep learning applications train neural networks with large memory footprints on data generated on endpoints. Deciding where and how to deploy these tasks presents complex trade-offs and requires careful analysis [19]. For example, cloud services like AWS SageMaker (component C2 in Figure 3) offer high-performance managed machine learning training in the cloud but require possibly privacy-sensitive data to be moved to public infrastructure. Federated learning provides a solution as users offload anonymized information of their privately trained model to the cloud instead of the data used to train the model.

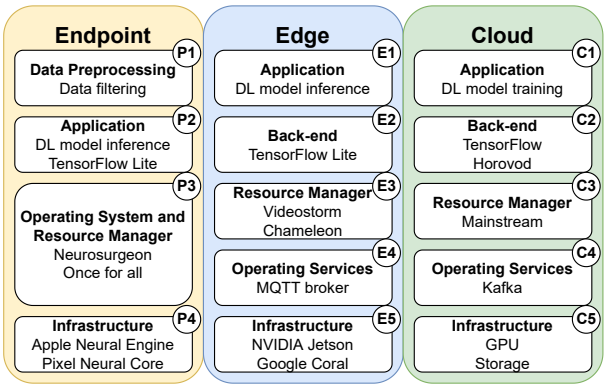

Fig. 3. Deep learning architecture. Examples include Neurosurgeon [19], Once for all [61], Videostorm [62], Chameleon [63], and Mainstream [64]

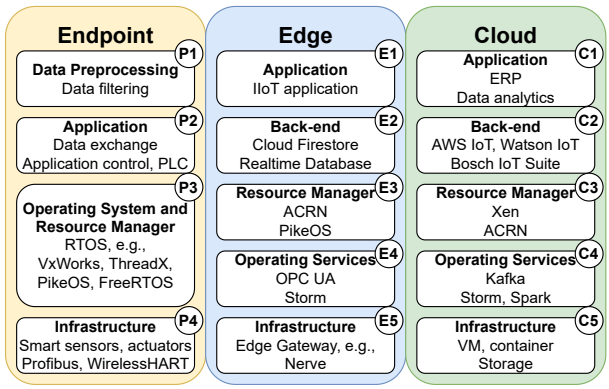

Fig. 4. Industrial IoT (IIoT) architecture. Examples include Cloud Firestore [67], ACRN [68], and PikeOS [69].

While model inference is much less compute and storage intensive than model training, it still requires the use of specialized deep learning frameworks like Neurosurgeon [19] (P3) and models like MobileNet [65] to be deployable on constrained edge or endpoint devices. These specialized frameworks and models present a complex trade-off between response time and model accuracy: By lowering the compute and storage requirements for model inference, model accuracy drops, but the application can be deployed on constrained devices close to the user, lowering the response time to the user. For recommender systems, real-time user feedback may be required, so response time is preferred above model accuracy, while for video analytics the opposite may apply.

Industrial IoT: In IIoT, endpoint devices generate large amounts of data that often need to be processed in real-time, with strict requirements for privacy and durability [66]. These endpoints are connected to Programmable Logic Controllers (PLCs, component P2 in Figure 4); these are control systems for local control without support for advanced processing due to resource limitations. Thus, offloading to remote devices over a fieldbus or via wireless communication is required (P3).

As many industry deployments include a vast amount of sensors and actuators, processing and storing large amounts of generated data requires extensive resources. Therefore, public cloud offerings like AWS IoT and Bosch IoT suite (C2) can be a good fit for many deployments [66]. However, reliable real-time processing guarantees may be broken if the network infrastructure between the endpoints and cloud can not support the large data streams. Moreover, offloading sensitive data to third parties may introduce security and privacy concerns. On-premise cloud devices are therefore often used in IIoT deployments, delivering more stable performance by eliminating possible connectivity issues to remote clouds and guaranteeing data privacy. In addition to clouds, edge gateways (E5) can be used to offer prompt response time and increased security.

In conclusion, with the example architectures for deep learning and IIoT we show that our uniform architecture helps developers and infrastructure providers navigate the compute continuum without being restricted to hardware and software solutions from a single computing model, but can freely combine solutions instead.

IV. Deployment and Benchmarking Framework

In the design of our reference architecture, we showed that by changing the implementation of components in our architecture, we can create deployments for any task offloading computing model. To demonstrate this mechanic in practice, we present Continuum, an infrastructure deployment and benchmarking framework for the compute continuum in this section and an accompanying analytical model in §V. The combination between infrastructure deployment and benchmarking, both highly configurable through a list of parameters, allows us to perform a systematic quantitative exploration of the compute continuum deployment space by performing performance analysis on architecture components. We attempt to answer three fundamental questions here:

1) *First, does our framework allow exploring different complex deployments in the compute continuum?* We explore different computing models in the framework using a few lines of changes in the configuration setup with parameters derived directly from the reference architecture (Table II and example Listing 1)).
2) *Secondly, how can our framework help explore design trade-offs?* Using the framework, we provide a breakdown of end-to-end latency (compute, communication) with an aggregation factor for various cloud, edge, and endpoint offload targets (Figures 6 and 7).
3) *Lastly, does the analytical model provide exploratory guidelines in line with the empirical evaluation?* We explain our first-order analytical model, corroborate its predictions with empirical evaluations, and show its offloading guidelines as a heatmap (Figure 8).

The framework and model are open-sourced and available at https://github.com/atlarge-research/continuum.

A. Framework Design and Interaction

Performing experiments in the continuum is challenging due to the wide range of networks and hardware available (Figure 1). Currently, no physical infrastructure is available that allows the exploration of all task offloading computing models and related deployments, only a selection [70]. As an alternative, devices and networks can be virtualized to

TABLE II
SELECTION OF PARAMETERS OFFERED BY THE FRAMEWORK.

Parameter	Architecture Component	Description
Data generation frequency	P1	Rate at which data is generated at endpoints.
Application	P1, P2, E1, E2, C1, C2	Application to deploy and benchmark.
Resource manager	P3, E3, E4, C3, C4	Resource manager to deploy in the continuum.
Hypervisor	P4, E5, C5	Virtual machine provider, e.g., QEMU.
Devices per tier	P4, E5, C5	Number of cloud, edge, and endpoint devices to emulate.
Cores per device	P4, E5, C5	Number of CPU cores to assign to each VM.
Quota per CPU	P4, E5, C5	Use part of a CPU core to emulate slower hardware.
Network per tier	P4, E5, C5	Throughput and latency between emulated devices.
Machine addresses	P4, E5, C5	Machine IP addresses when doing emulation across multiple physical machines.

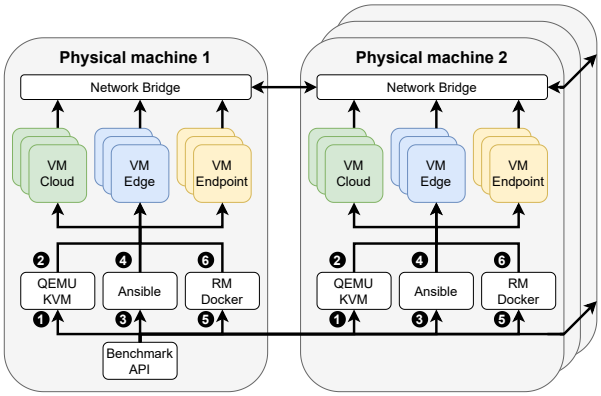

Fig. 5. Design of our continuum benchmarking framework.

```
[infrastructure]
hypervisor = qemu
thread_pinning = True

# VM settings for cloud, edge, endpoint
devices_per_tier = 10,0,40
cores_per_device = 4,0,1
quota_per_cpu = 1.0,0,0.5

# Latency (ms): average,variability
cloud_to_cloud = 1,0
cloud_to_endpoint = 45,5

# Throughput (Mbit): average
cloud_to_cloud = 1000
cloud_to_endpoint = 8

machine_address = 192.168.1.1,192.168.1.2

[benchmark]
use_benchmark = True
data_generation_frequency = 5
application = image_classification
resource_manager = kubernetes
```

Listing 1. Example configuration file for Figures 6 and 7.

emulate the compute continuum on commodity hardware. This emulation offers a flexible and easy-to-use method to explore the continuum's deployment space, as emulated hardware is configurable, unlike physical hardware. Furthermore, such an emulated environment can be used to benchmark various deployment scenarios using components from our reference architecture. We have created such an emulated deployment and benchmarking framework and present its design in Figure 5. Our framework allows the emulation of many virtual machines across many physical machines, ranging from commodity hardware to specialized edge or endpoint hardware. This flexibility allows our framework to function as an emulated benchmarking environment on general-purpose hardware and as a benchmark on physical compute continuum resources. The framework offers a set of simple and intuitive parameters that can be explored and mapped to the reference architecture to understand their implications on workload performance, as shown in Table II and abridged in Listing 1.

On the infrastructural level (components P4, E5, and C5 in Figure 2), users can specify the number of required cloud, edge, and endpoint devices (line 6) that should be emulated, the specifications of the emulated devices (cores, CPU share/quota, memory, storage, lines: 7-16), and the networks connecting the devices (component ❶ in Figure 5). We currently leverage the Linux QEMU/KVM hypervisor [71] (v6.0) with virtual machines (VMs) and tools (`tc` and `blkiotune`) to emulate a large continuum setup (❷) on a 3-machine Xeon Silver 4210R CPU cluster connected with a 1 Gbps link. For the next step, operating services and resource managers are installed in the provisioned VMs (P3, E3, E4, C3, C4) through Ansible [72] (❸ and ❹). We currently support Kubernetes (v1.21) as cloud resource manager, KubeEdge (v1.8) as edge resource manager, and OpenFaaS for serverless resources. On the application level (P1, E1, C1), we support data-processing applications with data generation at the endpoints and processing options at an endpoint, edge, and cloud (❺ and ❻).

B. Offload Model Design Space Exploration

This section illustrates how a developer explores the continuum design space using the aforementioned simple configurations with an image-processing workload as an example. In this workload, images are generated at endpoints (e.g., camera, thermal sensors), and can be processed at endpoints, edge, or cloud nodes with different CPU processing capabilities. We fix the image generation rate to five images per second, and set the bandwidth between the endpoint and its offloading targets to 8 Mbit/s, a representative throughput value for 4G networks [73].

Table III shows how varying different parameters in listing 1 allows us to explore four computing offload models on

TABLE III
PARAMETERS FOR THE DEPLOYMENTS USED IN OUR EVALUATION.

Parameter	Cloud	Edge-Large	Edge-Small	Mist
Resource manager	Kubernetes	KubeEdge	KubeEdge	-
Worker location	Cloud	Edge	Edge	Endpoint
Workers	10	10	10	10
Worker cores	4	4	2	2
Worker quota	1.0	1.0	0.75	0.5
Endpoints per worker	4	4	2	1
Network latency (ms)	45	30	7.5	7.5
(#cloud, #edge, #endpoint)	(11, 0, 40)	(1, 10, 40)	(1, 10, 20)	(0, 0, 20)

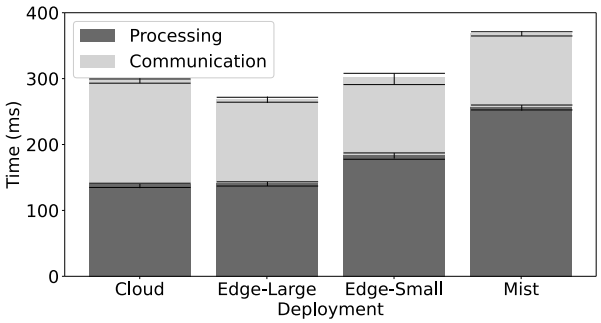

Fig. 6. Breakdown of the end-to-end latency per deployment.

our 3-machines cluster. For example, with *cloud* offloading, images are transmitted from the endpoint to a cloud cluster with 10 workers, each with 4 cores and full CPU quota (1.0), and each worker serves 4 endpoints (controlled via parameters `devices_per_tier`, `cores_per_device`, `quota_per_cpu`). There are 11 cloud nodes in total (10 workers and one controller), zero edge nodes, and 40 endpoints. In contrast, with *edge offloading*, workers can be placed at the edge using the parameters `devices_per_tier` and `quota_per_cpu` for large and small CPU configurations. Lastly, to capture *mist computing*, all image processing tasks are offloaded to other endpoint devices, thus marking the cloud and edge devices as zero (setting `devices_per_tier` as `0, 0, 40`). The Continuum framework also allows us to specify the network and storage properties.

For these four computing models, figure 6 shows our results. The x-axis shows the offloading model. The y-axis shows the end-to-end image processing time (in milliseconds, lower is better) split between computation and communication components. We report the average and standard deviation values over three runs. There are three main observations here. First, our framework allows explorations of these different offloading models with less than 5 lines of configuration changes between them (total size 50 lines, not shown). Second, we see a latency reduction between Cloud and Edge-large devices with equal CPU capabilities (the first two bars from the left) due to a decrease in communication time, as edge nodes are closer to the endpoint than cloud nodes. Lastly, as we move closer to the data source (Edge-small and Mist), we receive a gradual decrease in computing capabilities (0.75 and 0.5 CPU fractions) while still reducing communication latencies. However, gains from the latency reduction can not offset the overheads in computing due to slower processors; hence, the

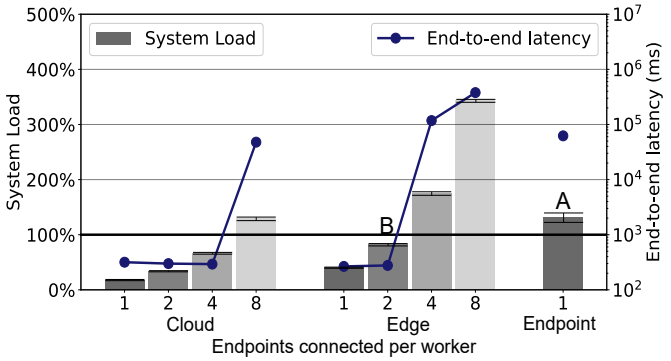

Fig. 7. System load when processing data, with more endpoints connected to one processing device. Experiments A and B are used for the performance model examples in Equations 3 and 4, respectively.

end-to-end processing latencies increase. Depending on the workload requirements, a developer can decide which model is of interest to them. For example, if the cutoff latency is 300ms, then a user must only consider Cloud or Edge-large, not Edge-small and Mist offloads.

In the second experiment, we explore the cardinality and location of data aggregation (P1, E1, and C1 boxes) by varying how many endpoints an offload target can serve. We identify this limit by defining a "system load" property, which is calculated by comparing the number of images processed (*compute capacity*) to images offloaded (*compute demand*) per second. A load of more than 100% implies more offloading requests are coming in than the worker device can process, resulting in queuing delays. Such setups are common where a developer needs to identify the best place for data aggregation to balance out computation and communication capabilities in the continuum. Close-to-source aggregation at endpoints reduces network transmission size but at the expense of using slower processors. We benchmark a single offload task with an aggregation cardinality of 1, 2, 4, and 8 connected endpoints. Figure 7 shows our results. The x-axis shows the aggregation granularity (cardinality and location), the left y-axis (the bar) shows the system load, and the right y-axis (log-scale, not starting from 1) shows the end-to-end latencies in milliseconds. Looking at the systems loads, we can observe that cloud-based aggregation can support up to 4 endpoints/task, while edge only supports 2 endpoints/task, after which the system load increases beyond 100% because a queueing delay of image processing requests is introduced in the end-to-end latencies. The graph also shows an endpoint-only bar that represents a locally-processed solution that is non-viable as the system load is more than 100%, thus failing to deliver real-time processing because of queueing delays.

The key takeaway messages with these experiments are that the open-source Continuum framework allows (i) a *comprehensive exploration* of various compute offloading models with an expressive list of parameters; and (ii) *a deep exploration* of specific setups such as aggregation point offloading and cardinality.

V. Analytical Performance Model

For our final contribution, we enhance the performance analysis capabilities of our compute continuum benchmark by introducing a simple-to-use, first-order analytical performance model similar to a Roofline model in Figure 8. The model predicts if applications can be offloaded to cloud or edge, should be processed locally on endpoints, or are not suited for deployment in the compute continuum. The performance model is part of the open-source benchmark suite.

A. Offload Model and Heatmap

To predict if applications can be executed in the continuum, we need to verify if the available network and compute resources satisfy an application's data and processing requirements. Equations 1 and 2 describe our performance models for local execution on endpoints and offloaded processing on edge or cloud. Only when the deployment meets all data and processing requirements will the models deem viable execution in the continuum, marked as 1 in the equations.

We assume long-running data processing applications are used with predictable and periodical offloading patterns, such as the machine learning application used in the benchmark evaluation, so we ignore startup and clean-up overheads when offloading tasks. This scenario is common in edge data processing as sensors and other data-generating devices like cameras are in constant use.

$$Local = \begin{cases} 0 & \text{if } (T_{proc} \times R) > (C_e \times Q_e) \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

$$Offload = \begin{cases} 0 & \text{if } (T_{proc} \times R \times E) > (C_o \times Q_o) \\ 0 & \text{if } (T_{pre} \times R) > (C_e \times Q_e) \\ 0 & \text{if } D > B \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where:

T_{proc}	Processing time per data element (sec)
T_{pre}	Preprocessing time per data element (sec)
C	CPU cores of endpoint e or offloading target o
Q	CPU quota of endpoint e or offloading target o
R	Data element generation rate (Hz)
E	Endpoints connected to offloading target
D	Data generated per second per endpoint (Mbps)
B	Bandwidth to offloading target (Mbps)

The formula captures compute capacity, represented as various calculations involving the CPU capacity C and quota Q at the offload targets, and compute demand, captured as the time to process each data element (T_{proc} and T_{pre}) with data element generation rate P. In the simplest terms, whenever demand (workload property) exceeds capacity (infrastructure property), that configuration setup is not viable (marked as 0, and the red zone in Figure 8). More specifically, with endpoints generating R images per second, each image should be processed, either locally or on an offload target, before the next image is generated. This guarantees real-time processing as there are no workload queues building up.

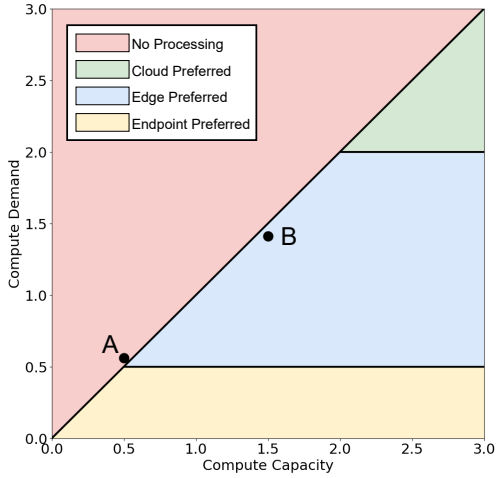

Fig. 8. Exploring preferred deployment models for the model examples from Equations 3 (A) and 4 (B). The division between cloud, edge, and endpoint preferred is configurable and based on computing capacity differences.

Equation 2 also identifies a non-viable case where data generation rates from endpoints are higher than the available bandwidth. For viable configurations, there is at least one viable processing location: Either locally on the endpoint (Eq. 1), or remotely on a cloud or edge by offloading (Eq. 2).

B. Verifying Empirical Results

We focus on two results marked as *A* (endpoint-only) and *B* (offloaded) in Figure 7. We use the parameters from these experiments as reported in Table III, including an R of 5. The equation for point A is:

$$\begin{aligned} Local : 0.11 \times 5 &= 1 \times 0.5 \\ 0.56 &> 0.5 \end{aligned} \tag{3}$$

Therefore, showing that local processing on an endpoint is not viable in this configuration because the endpoint can not keep up with the data generation speed. The estimated system load is $(0.56/0.5) * 100 = 112\%$, a slight difference from the actual system load of 131% noted as A in Figure 7, but still reaching the same conclusion. For the offloaded scenario with an aggregation of two endpoints:

$$\begin{aligned} Offload : 0.14 \times 5 \times 2 = 2 \times 0.75 \text{ and } 0.001 \times 5 &= 1 \times 0.5 \\ 1.4 < 1.5 \qquad \text{and} \qquad 0.005 &< 0.5 \end{aligned} \tag{4}$$

The left part of the equation shows that processing can be successfully offloaded to the edge, with an estimated system load of $(1.4/1.5) * 100 = 93\%$. This system load is again not much different from the actual system load of 83%, denoted as B in Figure 7. All three conditions for offloading are passed, showing the viability of this configuration: Offloading processing to the edge, preprocessing on the endpoint (the right part of Equation 4), and available bandwidth (2.7 Mbps required compared to 8 Mbps offered). Hence, our analytical model correctly predicts points A and B, with their position in Figure 8 indicating deployment viability or non-viability.

TABLE IV
SELECTION OF REFERENCE ARCHITECTURES MAPPED TO COMPUTING MODELS FOR TASK OFFLOADING. MC: MIST COMPUTING; EC: EDGE COMPUTING; MEC: MULTI-ACCESS EDGE COMPUTING; FC: FOG COMPUTING; MCC: MOBILE CLOUD COMPUTING. SYMBOLS: ●: PRESENT; ○: NOT PRESENT.

Authors	MC	MEC	EC	FC	MCC
Yogi et al. [74]	●	○	○	○	○
ETSI [37]	○	●	○	○	○
ECC [59]	○	○	●	○	○
IIC [8]	○	○	●	○	○
Intel, SAP [75]	○	○	●	○	○
OpenNebula [76]	○	○	●	○	○
Sittón-Candanedo et al. [77]	○	○	●	○	○
Qinglin et al. [78]	○	○	●	●	○
Willner et al. [79]	○	○	●	●	○
Mahmud et al. [80]	○	○	○	●	○
OpenFog Consortium [81]	○	○	○	●	○
Pop et al. [82]	○	○	○	●	○
Dinh et al. [25]	○	○	○	○	●
Compute continuum (this work)	●	●	●	●	●

VI. RELATED WORK

We have discussed past efforts in building isolated computing models in §III. To the best of our knowledge, only previous work from Qinglin et al. [78] and Willner et al. [79] has discussed the possibility of combining different computing models, however, this is limited to only edge and fog computing. Table IV summarizes previous efforts (and us, the last row) that cover different models in their analysis.

We design and implement a deployment and benchmarking framework for the compute continuum to explore the continuum's deployment space. On infrastructure provisioning and emulation, closest to our work, Symeonides et al. [83] present Fogify, a framework for emulating fog resources, and Hasenburg et al. present a similar framework with MockFog [84]. These systems are part of a larger class of cloud, edge, and endpoint resource emulation and simulation frameworks [85]. All support limited computing models and therefore are restricted in resources and deployments that can be emulated, unlike our framework, which enables emulation of all resources spawning the entire continuum. Additionally, by emulating virtual machines instead of isolation methods such as containers, our framework allows the configuration and benchmarking of architecture components other than applications. For example, users can switch between the Kubernetes and KubeEdge resource managers or add a new resource manager; the same applies to operating services. Table V shows the limitations of related work compared to our framework.

On benchmarking continuum resources and systems, closest to our work, Kimovski et al. [86] propose a benchmarking framework spawning edge, cloud, and fog; we add to it DeFog [87] and DeathStarBench [88]. These benchmark tools present a larger class of benchmarking tools for cloud, edge, and endpoint resources [89]. We improve upon these systems by offering more control over resource allocation (e.g., physical machines, virtual machines, containers) and deployment models (e.g., mist computing, edge computing, etc.) to benchmark and by allowing greater customization of compute and network resources. We argue that the coupling of infrastructure emulation and benchmarking that our framework offers is key in efficiently exploring the design space of continuum deployments and is not present in any related benchmark frameworks (the first two columns in Table V).

TABLE V
COMPARISON OF CHARACTERISTICS OF SELECTED EMULATION AND BENCHMARKING FRAMEWORKS FOR CLOUD AND EDGE.

	Infrastructure		Benchmark			This
Characteristic	[83]	[84]	[86]	[87]	[88]	work
Considers cloud resources	●	●	●	●	●	●
Considers edge resources	●	●	●	●	●	●
Considers endpoint resources	●	●	●	●	●	●
Considers network resources	●	●	●	●	●	●
Configurable compute resources	●	●	○	○	○	●
Configurable networks	●	●	○	○	○	●
Configurable resource managers	○	○	○	○	○	●
Configurable operating services	○	○	○	○	○	●
Application benchmarking	●	●	●	●	●	●
Supports all computing models	○	○	○	○	○	●

On modeling the performance of compute continuum systems, closest to our work, Majeed et al. [90] do performance modeling for workload offloading in the fog; we add to it work on network modeling from Ali-Eldin et al. [91]. These works provide detailed performance models for specific continuum deployments, unlike our first-order model, which can be applied to many models. Furthermore, our performance model is accompanied by an compute continuum benchmark that helps to quickly iterate on application deployment configurations.

VII. CONCLUSION AND ONGOING WORK

In this paper, we have made a case for unifying various past, present, and emerging computing models into a single unified compute continuum. To accomplish this, we provide a detailed analysis of 17 computing models, identify their unique characteristics, and unify them under a reference architecture. The reference architecture provides a conceptual framework for a systematic exploration of the deployment design space at the continuum. We then provide two unique instantiations of the reference architecture with machine learning and industrial IoT continuum workloads. To enhance the exploration guidelines, our reference architecture is accompanied by a deployment framework and first-order analytical model that help continuum developers to reason about their workloads with strong theoretical and engineering foundations. We are working to enhance the framework with cloud infrastructure support, energy modeling, and automatic ML-driven exploration capabilities. The effort to develop the reference architecture is a part of SPEC-RG, and the framework and model are open-source, and under active development: https://github.com/atlarge-research/continuum.

ACKNOWLEDGEMENT

This work is funded by NWO TOP project OffSense (OCENW.KLEIN.209), the EU GraphMassivizer project (No. 101093202), the Swedish Research Council (Vetenskapsrådet) – project "PSI" (No. 2020-05094), and the Knowledge Foundation (KKS) – project "SACSys" (No. 20190021).

References

[1] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 26–32. [Online]. Available: https://doi.org/10.1145/3458336.3465301

[2] S. Eismann, J. Scheuner, E. V. Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "The state of serverless applications: Collection, characterization, and community consensus," *IEEE Trans. Software Eng.*, vol. 48, no. 10, pp. 4152–4166, 2022. [Online]. Available: https://doi.org/10.1109/TSE.2021.3113940

[3] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become: The next phase of cloud computing," *Commun. ACM*, vol. 64, no. 5, p. 76–84, apr 2021. [Online]. Available: https://doi.org/10.1145/3406011

[4] M. Satyanarayanan, G. Klas, M. D. Silva, and S. Mangiante, "The seminal role of edge-native applications," in *3rd IEEE International Conference on Edge Computing, EDGE 2019, Milan, Italy, July 8-13, 2019*, E. Bertino, C. K. Chang, P. Chen, E. Damiani, M. Goul, and K. Oyama, Eds. IEEE, 2019, pp. 33–40. [Online]. Available: https://doi.org/10.1109/EDGE.2019.00022

[5] AWS, "AWS Greengrass," https://aws.amazon.com/greengrass/, 2021, accessed: 2021-05-12.

[6] S. Lin, Y. Zhang, C. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, X. Shen, J. Tuck, R. Bianchini, and V. Sarkar, Eds. ACM, 2018, pp. 751–766. [Online]. Available: https://doi.org/10.1145/3173162.3173191

[7] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. N. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman, "Farmbeats: An iot platform for data-driven agriculture," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, A. Akella and J. Howell, Eds. USENIX Association, 2017, pp. 515–529. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vasisht

[8] M. Tseng, T. Edmunds, and L. Canaran, "Introduction to edge computing in IIoT," Industrial Internet Consortium, Tech. Rep., 2018. [Online]. Available: https://www.iiconsortium.org/pdf/Introduction_to_Edge_Computing_in_IIoT_2018-06-18.pdf

[9] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wirel. Commun.*, vol. 26, no. 4, pp. 178–183, 2019. [Online]. Available: https://doi.org/10.1109/MWC.2019.1800440

[10] P. Patel, M. I. Ali, and A. P. Sheth, "On using the intelligent edge for iot analytics," *IEEE Intell. Syst.*, vol. 32, no. 5, pp. 64–69, 2017. [Online]. Available: https://doi.org/10.1109/MIS.2017.3711653

[11] M. Z. Khan, S. Harous, S. U. Hassan, M. U. G. Khan, R. Iqbal, and S. Mumtaz, "Deep unified model for face recognition based on convolution neural network and edge computing," *IEEE Access*, vol. 7, pp. 72 622–72 633, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2918275

[12] Linux Foundation, "State of the edge 2021," https://project.linuxfoundation.org/hubfs/LF%20Edge/StateoftheEdgeReport_2021.pdf, 2021, accessed: 2021-06-06.

[13] A. Trivedi, L. Wang, H. Bal, and A. Iosup, "Sharing and caring of data at the edge," in *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*. USENIX Association, Jun. 2020. [Online]. Available: https://www.usenix.org/conference/hotedge20/presentation/trivedi

[14] N. Sreekumar, A. Chandra, and J. B. Weissman, "Position paper: Towards a robust edge-native storage system," in *5th IEEE/ACM Symposium on Edge Computing, SEC 2020, San Jose, CA, USA, November 12-14, 2020*. IEEE, 2020, pp. 285–292. [Online]. Available: https://doi.org/10.1109/SEC50012.2020.00040

[15] R.-A. Cherrueau, A. Lebre, D. Pertin, F. Wuhib, and J. M. Soares, "Edge computing resource management system: a critical building block! initiating the debate via OpenStack," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/cherrueau

[16] B. Varghese, E. de Lara, A. Y. Ding, C. Hong, F. Bonomi, S. Dustdar, P. Harvey, P. Hewkin, W. Shi, M. Thiele, and P. Willis, "Revisiting the arguments for edge computing research," *IEEE Internet Comput.*, vol. 25, no. 5, pp. 36–42, 2021. [Online]. Available: https://doi.org/10.1109/MIC.2021.3093924

[17] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009. [Online]. Available: https://doi.org/10.1109/MPRV.2009.82

[18] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018*. IEEE, 2018, pp. 373–377. [Online]. Available: https://doi.org/10.1109/SEC.2018.00048

[19] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. N. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2017, Xi'an, China, April 8-12, 2017*, Y. Chen, O. Temam, and J. Carter, Eds. ACM, 2017, pp. 615–629. [Online]. Available: https://doi.org/10.1145/3037697.3037698

[20] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara, "Cloudpath: a multi-tier cloud computing framework," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose / Silicon Valley, SEC 2017, CA, USA, October 12-14, 2017*, J. Zhang, M. Chiang, and B. M. Maggs, Eds. ACM, 2017, pp. 20:1–20:13. [Online]. Available: https://doi.org/10.1145/3132211.3134464

[21] J. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in fog and mist computing," *Computer*, vol. 48, no. 7, pp. 37–45, 2015. [Online]. Available: https://doi.org/10.1109/MC.2015.207

[22] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. [Online]. Available: https://doi.org/10.1109/MC.2017.9

[23] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2705720

[24] F. Bonomi, R. A. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC@SIGCOMM 2012, Helsinki, Finland, August 17, 2012*, M. Gerla and D. Huang, Eds. ACM, 2012, pp. 13–16. [Online]. Available: https://doi.org/10.1145/2342509.2342513

[25] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wirel. Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013. [Online]. Available: https://doi.org/10.1002/wcm.1203

[26] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, 2019. [Online]. Available: https://doi.org/10.1016/j.sysarc.2019.02.009

[27] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 39:1–39:34, 2018. [Online]. Available: https://doi.org/10.1145/3154815

[28] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 1, pp. 416–464, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2017.2771153

[29] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2705720

[30] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 98–105, 2015. [Online]. Available: https://doi.org/10.1109/MCOM.2015.7060488

[31] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-peer computing,"

https://www.cs.kau.se/cs/education/courses/dvad02/p2/seminar4/Papers/HPL-2002-57R1.pdf, 2002.

[32] N. Fernando, S. W. Loke, and J. W. Rahayu, "Honeybee: A programming framework for mobile crowd computing," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services - 9th International Conference, MobiQuitous 2012, Beijing, China, December 12-14, 2012. Revised Selected Papers*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, K. Zheng, M. Li, and H. Jiang, Eds., vol. 120. Springer, 2012, pp. 224–236. [Online]. Available: https://doi.org/10.1007/978-3-642-40238-8_19

[33] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan, "The case for mobile edge-clouds," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, UIC/ATC 2013, Vietri sul Mare, Sorrento Peninsula, Italy, December 18-21, 2013*. IEEE Computer Society, 2013, pp. 209–215. [Online]. Available: https://doi.org/10.1109/UIC-ATC.2013.94

[34] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, 2017. [Online]. Available: https://doi.org/10.1109/MNET.2017.1700030

[35] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019. [Online]. Available: https://doi.org/10.1109/JSAC.2019.2904348

[36] M. ETSI, "Mobile edge computing (mec); framework and reference architecture," *ETSI, DGS MEC*, vol. 3, 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/mec/001_099/003/01.01.01_60/gs_mec003v010101p.pdf

[37] ——, "Multi-access edge computing (mec); framework and reference architecture," *ETSI, GS MEC*, vol. 3, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf

[38] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2579198

[39] L. Corneo, M. Eder, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, P. Gunningberg, J. Kangasharju, and J. Ott, "Surrounded by the clouds: A comprehensive cloud reachability study," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 295–304. [Online]. Available: https://doi.org/10.1145/3442381.3449854

[40] T. Guan, E. Zaluska, and D. D. Roure, "A grid service infrastructure for mobile devices," in *2005 International Conference on Semantics, Knowledge and Grid (SKG 2005), 27-29 November 2005, Beijing, China*. IEEE Computer Society, 2005, p. 42. [Online]. Available: https://doi.org/10.1109/SKG.2005.10

[41] C. Li, Y. Hu, L. Liu, J. Gu, M. Song, X. Liang, J. Yuan, and T. Li, "Towards sustainable in-situ server systems in the big data era," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, June 13-17, 2015*, D. T. Marr and D. H. Albonesi, Eds. ACM, 2015, pp. 14–26. [Online]. Available: https://doi.org/10.1145/2749469.2750381

[42] M. Villari, M. Fazio, S. Dustdar, O. F. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76–83, 2016. [Online]. Available: https://doi.org/10.1109/MCC.2016.124

[43] V. van Rijn and J. S. Rellermeyer, "A fresh look at the architecture and performance of contemporary isolation platforms," in *Proceedings of the 22nd International Middleware Conference*, ser. Middleware '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 323–335. [Online]. Available: https://doi.org/10.1145/3464298.3493404

[44] K. Razavi and A. Trivedi, "Stratus: Clouds with microarchitectural resource management," in *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*. USENIX Association, Jul. 2020. [Online]. Available: https://www.usenix.org/conference/hotcloud20/presentation/razavi

[45] G. Yadgar, O. Kolosov, M. F. Aktas, and E. Soljanin, "Modeling the edge: Peer-to-Peer reincarnated," in *HotEdge 19*. Renton, WA: USENIX Association, Jul. 2019. [Online]. Available: https://www.usenix.org/conference/hotedge19/presentation/yadgar

[46] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality mmogs," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3132211.3134463

[47] W. Hu, B. Amos, Z. Chen, K. Ha, W. Richter, P. Pillai, B. Gilbert, J. Harkes, and M. Satyanarayanan, "The case for offload shaping," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile 2015, Santa Fe, NM, USA, February 12-13, 2015*, J. Manweiler and R. R. Choudhury, Eds. ACM, 2015, pp. 51–56. [Online]. Available: https://doi.org/10.1145/2699343.2699351

[48] M. Satyanarayanan, W. Gao, and B. Lucia, "The computing landscape of the 21st century," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications, HotMobile 2019, Santa Cruz, CA, USA, February 27-28, 2019*, A. Wolman and L. Zhong, Eds. ACM, 2019, pp. 45–50. [Online]. Available: https://doi.org/10.1145/3301293.3302357

[49] J. L. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *ASPLOS-IX Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, November 12-15, 2000*, L. Rudolph and A. Gupta, Eds. ACM Press, 2000, pp. 93–104. [Online]. Available: https://doi.org/10.1145/378993.379006

[50] R.-A. Cherrueau, A. Lebre, D. Pertin, F. Wuhib, and J. M. Soares, "Edge computing resource management system: a critical building block! initiating the debate via OpenStack," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/cherrueau

[51] P. K. Gadepalli, S. McBride, G. Peach, L. Cherkasova, and G. Parmer, "Sledge: a serverless-first, light-weight wasm runtime for the edge," in *Middleware '20: 21st International Middleware Conference, Delft, The Netherlands, December 7-11, 2020*, D. D. Silva and R. Kapitza, Eds. ACM, 2020, pp. 265–279. [Online]. Available: https://doi.org/10.1145/3423211.3425680

[52] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *J. Open Source Softw.*, vol. 2, no. 13, p. 265, 2017. [Online]. Available: https://doi.org/10.21105/joss.00265

[53] Erwin, "Erwin edge," https://www.erwin.com/products/, 2021, accessed: 2021-05-30.

[54] Z. Hao, S. Yi, and Q. Li, "Edgecons: Achieving efficient consensus in edge computing networks," in *USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2018, Boston, MA, July 10, 2018*, I. Ahmad and S. Sundararaman, Eds. USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/hao

[55] Tencent, Intel, VMware, Huya, Cambricon, Captialonline, and Meituan, "Superedge," https://github.com/superedge/superedge, 2021, accessed: 2021-06-06.

[56] H. Gupta and U. Ramachandran, "Fogstore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access," in *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, DEBS 2018, Hamilton, New Zealand, June 25-29, 2018*, A. Hinze, D. M. Eyers, M. Hirzel, M. Weidlich, and S. Bhowmik, Eds. ACM, 2018, pp. 148–159. [Online]. Available: https://doi.org/10.1145/3210284.3210297

[57] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 1086–1099, 2020. [Online]. Available: https://doi.org/10.1109/TSC.2017.2753775

[58] Microsoft, "Microsoft Azure IoT," https://azure.microsoft.com/en-us/services/iot-hub/, 2021, accessed: 2021-05-18.

[59] E. C. C. (ECC) and A. of Industrial Internet (AII), "Edge computing reference architecture 2.0," ECC, AII, Tech. Rep., 2017. [Online]. Available: http://en.ecconsortium.net/Uploads/file/20180328/1522232376480704.pdf

[60] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," in *World Congress on Services, SERVICES 2011, Washington, DC, USA, July 4-9, 2011*. IEEE Computer Society, 2011, pp. 594–596. [Online]. Available: https://doi.org/10.1109/SERVICES.2011.105

[61] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train

one network and specialize it for efficient deployment," 2019. [Online]. Available: https://arxiv.org/abs/1908.09791

[62] H. Zhang, G. Ananthanarayanan, P. Bodík, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, A. Akella and J. Howell, Eds. USENIX Association, 2017, pp. 377–392. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang

[63] J. Jiang, G. Ananthanarayanan, P. Bodík, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, S. Gorinsky and J. Tapolcai, Eds. ACM, 2018, pp. 253–266. [Online]. Available: https://doi.org/10.1145/3230543.3230574

[64] A. H. Jiang, D. L. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger, "Mainstream: Dynamic stem-sharing for multi-tenant video processing," in *2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*, H. S. Gunawi and B. Reed, Eds. USENIX Association, 2018, pp. 29–42. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/jiang

[65] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4510–4520. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[66] K. Al-Gumaei, K. Schuba, A. Friesen, S. Heymann, C. Pieper, F. Pethig, and S. Schriegel, "A survey of internet of things and big data integrated solutions for industrie 4.0," in *23rd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2018, Torino, Italy, September 4-7, 2018*. IEEE, 2018, pp. 1417–1424. [Online]. Available: https://doi.org/10.1109/ETFA.2018.8502484

[67] Google, "Firebase database," https://firebase.google.com/docs/database/, 2021, accessed: 2021-06-22.

[68] L. F. project, "Acrn," https://projectacrn.org/, 2021, accessed: 2021-06-22.

[69] SYSGO, "Pikeos," https://www.sysgo.com/pikeos, 2021, accessed: 2021-06-22.

[70] B. C. Senel, M. Mouchet, J. Cappos, O. Fourmaux, T. Friedman, and R. McGeer, "Edgenet: A multi-tenant and multi-provider edge cloud," in *EdgeSys@EuroSys 2021: 4th International Workshop on Edge Systems, Analytics and Networking, Online Event, United Kingdom, April 26, 2021*, A. Y. Ding and R. Mortier, Eds. ACM, 2021, pp. 49–54. [Online]. Available: https://doi.org/10.1145/3434770.3459737

[71] F. Bellard, "Qemu, a fast and portable dynamic translator," in *Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference, April 10-15, 2005, Anaheim, CA, USA*. USENIX, 2005, pp. 41–46. [Online]. Available: http://www.usenix.org/events/usenix05/tech/freenix/bellard.html

[72] Hochstein and Moser, *Ansible: Up and Running: Automating configuration management and deployment the easy way*. O'Reilly, 2017.

[73] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g LTE dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference, MMSys 2018, Amsterdam, The Netherlands, June 12-15, 2018*, P. César, M. Zink, and N. Murray, Eds. ACM, 2018, pp. 460–465. [Online]. Available: https://doi.org/10.1145/3204949.3208123

[74] M. K. Yogi, K. Chandrasekhar, and G. V. Kumar, "Mist computing: Principles, trends and future direction," *CoRR*, vol. abs/1709.06927, 2017. [Online]. Available: http://arxiv.org/abs/1709.06927

[75] Intel and SAP, "IoT joint reference architecture from intel and sap," Intel, SAP, Tech. Rep., 2018. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/reference-architectures/sap-iot-reference-architecture.pdf

[76] OpenNebula, "Edge cloud architecture - white paper," OpenNebula, Tech. Rep. 2.2, 2021. [Online]. Available: https://support.opennebula.pro/hc/en-us/article_attachments/360019858317/OpenNebula_-_Edge_Cloud_Reference_Architecture_r2.2_20210507.pdf

[77] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Gener. Comput. Syst.*, vol. 99, pp. 278–294, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.04.016

[78] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86 769–86 777, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2923610

[79] A. Willner and V. Gowtham, "Toward a reference architecture model for industrial edge computing," *IEEE Commun. Stand. Mag.*, vol. 4, no. 4, pp. 42–48, 2020. [Online]. Available: https://doi.org/10.1109/MCOMSTD.001.2000007

[80] M. R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in iot-enabled healthcare solutions," in *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, P. Bellavista and V. K. Garg, Eds. ACM, 2018, pp. 32:1–32:10. [Online]. Available: https://doi.org/10.1145/3154273.3154347

[81] O. C. A. W. Group, "Openfog reference architecture for fog computing," OpenFog Consortium, Tech. Rep. OPFRA001, 2017. [Online]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf

[82] P. Pop, B. Zarrin, M. Barzegaran, S. Schulte, S. Punnekkat, J. Ruh, and W. Steiner, "The FORA fog computing platform for industrial iot," *Inf. Syst.*, vol. 98, p. 101727, 2021. [Online]. Available: https://doi.org/10.1016/j.is.2021.101727

[83] M. Symeonides, Z. Georgiou, D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Fogify: A fog computing emulation framework," in *5th IEEE/ACM Symposium on Edge Computing, SEC 2020, San Jose, CA, USA, November 12-14, 2020*. IEEE, 2020, pp. 42–54. [Online]. Available: https://doi.org/10.1109/SEC50012.2020.00011

[84] J. Hasenburg, M. Grambow, and D. Bermbach, "Mockfog 2.0: Automated execution of fog application experiments in the cloud," *CoRR*, vol. abs/2009.10579, 2020. [Online]. Available: https://arxiv.org/abs/2009.10579

[85] J. Taheri and S. Deng, *Edge Computing: Models, Technologies and Applications*. The Institution of Engineering and Technology (IET), 2020. [Online]. Available: https://digital-library.theiet.org/content/books/pc/pbpc033e

[86] D. Kimovski, R. Mathá, J. Hammer, N. Mehran, H. Hellwagner, and R. Prodan, "Cloud, fog, or edge: Where to compute?" *IEEE Internet Comput.*, vol. 25, no. 4, pp. 30–36, 2021. [Online]. Available: https://doi.org/10.1109/MIC.2021.3050613

[87] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "Defog: fog computing benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC 2019, Arlington, Virginia, USA, November 7-9, 2019*, S. Chen, R. Onishi, G. Ananthanarayanan, and Q. Li, Eds. ACM, 2019, pp. 47–58. [Online]. Available: https://doi.org/10.1145/3318216.3363299

[88] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, B. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espinosa, R. Lin, Z. Liu, J. Padilla, and C. Delimitrou, "An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, I. Bahar, M. Herlihy, E. Witchel, and A. R. Lebeck, Eds. ACM, 2019, pp. 3–18. [Online]. Available: https://doi.org/10.1145/3297858.3304013

[89] B. Varghese, N. Wang, D. Bermbach, C. Hong, E. de Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 66:1–66:33, 2021. [Online]. Available: https://doi.org/10.1145/3444692

[90] A. A. Majeed, P. Kilpatrick, I. T. A. Spence, and B. Varghese, "Modelling fog offloading performance," in *4th IEEE International Conference on Fog and Edge Computing, ICFEC 2020, Melbourne, Australia, May 11-14, 2020*. IEEE, 2020, pp. 29–38. [Online]. Available: https://doi.org/10.1109/ICFEC50348.2020.00011

[91] A. Ali-Eldin, B. Wang, and P. J. Shenoy, "The hidden cost of the edge: a performance comparison of edge and cloud latencies," in *SC '21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021*, B. R. de Supinski, M. W. Hall, and T. Gamblin, Eds. ACM, 2021, pp. 23:1–23:12. [Online]. Available: https://doi.org/10.1145/3458817.3476142

APPENDIX

A. Abstract

This artifact appendix provides all information required to reproduce the experiments presented in §IV and §V. The artifact is open-source code for Continuum, our framework for automated cloud-edge infrastructure deployments and benchmarks in the compute continuum. This paper uses the Continuum framework for all benchmarking and modeling experiments. The appendix contains the following:

1) A description of the artifact
2) Requirements for reproducing the experiments
3) Detailed installation steps for Continuum
4) Scripts to reproduce our experiments and graphs
5) Options for further customization

B. Artifact Description

Continuum is open-source with an MIT license and is available on Zenodo at https://doi.org/10.5281/zenodo.7716297. The framework is also available on GitHub, where the most up-to-date version of the artifact can be found: https://github.com/atlarge-research/continuum/tree/CCGRID2023-Artifact-Evaluation. We provide a more extensive version of this appendix on Zenodo and GitHub. The framework includes all code and scripts required to reproduce the paper's results but excludes software dependencies. More specifically, Continuum provides the following functionalities:

1) Infrastructure deployment: Create virtual compute continuum infrastructure in the cloud or on local hardware.
2) Software installation: Automatically install complex software deployment on the provided infrastructure.
3) Benchmark execution: Execute application- and system-level benchmarks on the continuum deployments.

The repository has the following directory structure:

1) application: The benchmarked applications. Also available on DockerHub.
2) benchmark: Components for the benchmark setup and execution, as well as output processing.
3) configuration: Input configuration files for Continuum.
4) configuration_parser: The configuration parser.
5) docs: Documentation on how to use Continuum.
6) infrastructure: Management of infrastructure providers such as QEMU and GCP.
7) resource_manager: Installation scripts for distributed services and resource management software.
8) scripts: Scripts for replicating paper results.

C. Hardware Requirements

Continuum can emulate compute continuum infrastructure on local hardware using QEMU and in the cloud using Google Cloud Platform (GCP). In this artifact appendix, we describe Continuum's functionalities for GCP, as this requires significantly less setup and hardware from the user. The hardware requirement for Continuum using GCP is any computer with internet access. All experiments have been executed on Ubuntu 20.04; we recommend this operating system to reproduce the experiments. A virtual machine can be used in the absence of a physical device with this operating system. Continuum requires about 150 MB of storage space for the code and up to 2 GB for the containerized applications.

D. Software Requirements

The Continuum framework has the following dependencies and has been tested with the following software versions:

1) Docker 23.0.0: Continuum uses containerized applications for its benchmarks and requires Docker for container management on the host device. Continuum builds an image repository on the local host device before moving the images to GCP.
2) Python 3.8.10: The majority of Continuum's code is written in the Python programming language.
3) Pip3 20.0.2: Python's package manager.
4) Python 3 packages: Numpy 1.21., Pandas 1.3.4, and Matplotlib 3.5.0.
5) Ansible 2.9.6: Continuum uses Ansible for automatic software installation and configuration on GCP virtual machines, managed from the host device.
6) Terraform 1.3.7: Terraform is an infrastructure-as-code service that simplifies deployment on GCP.

E. Getting Started

The following steps show how to run a simple example deployment in 15 minutes. These steps are extensively described in the codebase's documentation.

Main.py is the entry point of the framework. It requires a configuration file that describes what deployment it should create. The configuration file layout is defined in Table 2 of the CCGRID paper. In this example, Continuum creates a Kubernetes cluster on Google Cloud, using two cloud VMs (one for the Kubernetes control plane, one as a Kubernetes worker) and one endpoint VM (the user that offloads data to the cloud). When Continuum is finished, it outputs the results of the performed benchmark to the newly created logs directory. We provide an example output below.

```
# 1: Prepare the hardware environment

# 2: Prepare the software environment
git clone https://github.com/atlarge-research/continuum
git checkout CCGRID2023-Artifact-Evaluation
# - Follow the README

# 3: Start a simple example
cd continuum
python3 main.py configuration/gcp_cloud_kubernetes_benchmark.cfg
```

In this example, one endpoint offloads five images per second to one cloud worker for 300 seconds, then the cloud worker performs image classification on each image and sends the results back to the endpoint. The time between the endpoint application sending the image and the cloud application receiving the image was 107.49 on average; it then took the cloud worker 81.46 ms to process each image on average,

for a total end-to-end latency (the time between the endpoint generating an image and getting the image classification output back) of 191.23 ms per image on average.

```
-----------------------------------
CLOUD OUTPUT
-----------------------------------
worker_id               0
total_time (s)          302.06
delay_avg (ms)          107.49
delay_stdev (ms)        23.92
proc_time/data (ms)     81.46
-----------------------------------
ENDPOINT OUTPUT
-----------------------------------
connected_to            0
total_time (s)          302.45
preproc_time/data (ms)  0.45
data_size_avg (kb)      65.28
latency_avg (ms)        191.23
latency_stdev (ms)      24.44
```

F. Reproduce Figure 6

The following code describes how to reproduce Figure 6, the breakdown of the end-to-end latency per deployment, and takes approximately 1 hour to execute:

```
cd continuum
rm -r logs/*.log
cd scripts
python3 replicate_paper.py Deployments
```

The configurations used for these experiments can be found in the configuration directory. The output can be found in the logs directory and contains a graph similar to Figure 6. Results may differ between the produced graph and the original Figure 6 as we use Google Cloud instead of QEMU. However, the general trend in the results will remain the same. This also applies to Figures 7 and 8.

G. Reproduce Figure 7

The following code describes how to reproduce Figure 7, the system load and end-to-end latency for deployments with varying numbers of endpoints connected per worker, and takes approximately 2 hour to execute:

```
cd continuum
rm -r logs/*.log
cd scripts
python3 replicate_paper.py EndpointScaling
```

H. Reproduce Figure 8

Finally, the following code describes how to reproduce Figure 8 in 1 hour, which visualizes the analytical model.

```
cd continuum
rm -r logs/*.log
cd scripts
python3 replicate_model.py
```

I. Customization

Continuum allows for very fine-grained customization of deployments using the configuration file format. We provide a template in the configuration directory that lists all options that the framework currently supports. Moreover, the framework also contains a suite of configurations that are used to test Continuum's functionalities. These can be used as follows:

```
cd continuum
for i in configuration/tests/terraform/*.cfg;
    do
    python3 main.py $i || break
done
```

J. Future Work

The Continuum framework is part of active research and development. The repository's main branch lists many more functionalities that the framework supports, such as bare-metal and serverless deployments. For more information, see https://github.com/atlarge-research/continuum.