2021 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC) | 978-1-6654-1130-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/UrgentHPC54802.2021.00007

# Evaluating policy-driven adaptation on the Edge-to-Cloud Continuum

Daniel Balouek-Thomert, Ivan Rodero, Manish Parashar
{daniel.balouek, ivan.rodero, parashar}@utah.edu
SCI Institute, University of Utah, UT, USA

***Abstract*—Developing data-driven applications requires developers and service providers to orchestrate data-to-discovery pipelines across distributed data sources and computing units. Realizing such pipelines poses two major challenges: programming analytics that reacts at runtime to unforeseen events, and adaptation of the resources and computing paths between the edge and the cloud. While these concerns are interdependent, they must be separated during the design process of the application and the deployment operations of the infrastructure. This work proposes a system stack for the adaptation of distributed analytics across the computing continuum. We implemented this software stack to evaluate its ability to continually balance the computation or data movement's cost with the value of operations to the application objectives. Using a disaster response application, we observe that the system can select appropriate configurations while managing trade-offs between user-defined constraints, quality of results, and resource utilization. The evaluation shows that our model is able to adapt to variations in the data input size, bandwidth, and CPU capacities with minimal deadline violations (close to 10%). This constitutes encouraging results to benefit and facilitate the creation of ad-hoc computing paths for urgent science and time-critical decision-making.**

***Index Terms*—Cloud computing, Edge computing, Computing Continuum, Decision Model**

## I. Introduction

Urgent science refers to a class of time-critical scientific applications that leverage distributed data sources to facilitate important decision-making in a timely manner. Examples of urgent science span various domains ranging from applications that aim to improve quality of life, monitor civil infrastructures, respond to natural disasters and extreme events, and accelerate science. The exponential growth of available digital data sources coupled with pervasive access to nontrivial computing capabilities and the availability of sophisticated data analytics has the potential to enable end-to-end workflows that combine these elements to the model, manage, control, adapt and optimize sub-systems of interest.

However, while our capacity for collecting data is expanding dramatically, our ability to manage, manipulate, and analyze this data, transform it into knowledge and understanding, and integrate it with practice has not kept pace. For example, most popular data analytics solutions, including those based on AI/ML, are cloud-based and require transporting data from often distant edge devices to a central location for processing. This limits the amount of data that we can process and our ability to analyze and transform this data into knowledge in a timely manner.

Moving away from traditional centralized Cloud models, the use of resources between the Edge and Core of the infrastructure allows distribution of analytics while preserving low latency, high availability, and privacy. This aggregation of heterogeneous resources along the data path from the Edge to the Cloud also referred to as the Edge-to-Cloud Continuum, or the Computing Continuum [1], [2], can be harnessed to support urgent analytics. For instance, in the case of ML-based analytics, data inference can be executed on the Edge in real-time, and only relevant or pre-processed data is forwarded to the Cloud. Furthermore, intermediate results can be aggregated from multiple Edge devices (i.e., in the Fog), allowing fast decision-making based on location or on utility functions.

Realizing this vision of the computing continuum for urgent science poses challenges associated with the variability of the computational environment and the ability to ensure end-to-end guarantees for the applications. This extreme heterogeneity, in the capabilities and capacities of systems and services, is furthered coupled with extreme uncertainty arising from variabilities in the availability and quality of data, resources, and services. Addressing this heterogeneity requires application formulations and programming abstractions that allow developers to expose natural flexibilities and tradeoffs, and define policies and mechanisms that can drive runtime adaptions. The timeliness of urgent decisions also makes providing end-to-end guarantees critical and challenging.

To leverage the computing continuum for data-driven workflows, we propose a system stack for modeling and deploying data-driven applications that react to events happening at runtime. The objectives of this model are *1)* to facilitate the implementations of policies that ensure performance guarantees at runtime and, *2)* to deploy them with simplicity atop the Edge-to-Cloud Continuum.

To enable this approach, we evaluate a set of adaptation policies (e.g., *cloud-only*, *edge-only*, *hybrid*, *resource utilization maximization*, *hard deadline*), which aim to describe the reactions of the application and to make more explicit the user expectations and constraints in terms of response time, solution quality, etc. according to what is currently available in a dynamic computation and communication environment. These policies further drive the reconfiguration of applications, resources, and services, including, for example, provisioning new resources and services, adapting scheduling strategies, selecting appropriate systems services, and/or adjusting ap-

978-1-6654-1130-1/21/$31.00 ©2021 IEEE
DOI 10.1109/UrgentHPC54802.2021.00007

plication parameters as necessary.

The contributions of this paper are:

- an implementation of a unified framework for continuum computing built on our prior research [2] – (Section III);
- an experimental validation of a disaster recovery workflow under different policies on an Edge-to-Cloud infrastructure emulated atop the Grid'5000 large-scale experimental testbed (Sections IV and V);

## II. Motivation

In this section, we introduce the concept of urgent science and the earthquake early warning use case that motivated this work. We then discuss the challenges associated with executing distributed analytics on the Edge-to-Cloud Continuum.

### A. Urgent Science

As available data increases in scale, heterogeneity, and richness, data-driven urgent workflows are enabling new and transformative applications across many disciplines. These workflows aim to process machine-generated data in a timely manner to identify context-sensitive features and events and produce critical insights. For example, emerging urban mobility applications rely on traffic sensors' processing large amounts of traffic data in real-time to identify and alleviate traffic congestion. Similarly, autonomous cars using sensors such as LiDAR and cameras are expected to gather and process, in a timely manner, many TB's of data per day [3]. Many of these workflows involve complex models that are based on the efficient and time-constrained fusion of data from different domains.

Urgent science workflows present additional requirements and constraints due to the nature and distribution of the data, the complexity of the models involved, the stringent error thresholds, and the strict time constraints. The execution of these workflows has to balance the need for a large amount of computational power to reduce errors while ensuring the timely processing and assimilation of essential data streams [4], [5].

### B. Earthquake Early Warning

Earthquakes are amongst the most destructive natural disasters. Networks of distributed seismic instruments on various scales are used for earthquake detection. *Earthquake Early Warning* (EEW) systems provide earthquake alerts before the shaking damage of a seismic event reaches sensitive areas, giving governments and communities a time window of seconds to minutes to take protective actions.

EEW can be described as a *classification problem* in which high-frequency seismic data streams from multiple sensors are processed to infer classes indicating the magnitude of the seismic event in a timely manner. Traditionally, EEW is executed in a fully centralized fashion with data from sensors being sent to Clouds.

At present, there are EEW systems operational in several countries [6]. The Japan Meteorological Agency (JMA) began operating a system in 2007 that consists of over 4,000 contributing stations, with a typical station interval of about 20 km, and performed efficiently when an earthquake hit Japan in 2011 [7]. Mexico implemented a seismic alert system in 1991, allowing for a warning time of 58 to 74 seconds [8]. Notable related efforts include novel algorithms recently developed to locate earthquakes and to calculate their magnitudes using P- and S-wave energy [9]. Recently, ShakeAlert proposed to detect and disseminate EEW alerts using smartphones, relying on the fact that they have become ubiquitous to the public [10], [11].

In a previous work [12], we proposed moving part of the sensor data processing towards the Edge to speed up detection while further enhancing network usage reduction, fault tolerance, and idle machines usage. The Distributed Multi-Sensor Earthquake Early Warning (DMSEEW) is a two-step ensemble method for earthquake detection. DMSEEW takes sensor-level class predictions (normal activity, medium earthquake, or large earthquake) based on the data gathered by each sensor, aggregates them using a bag-of-words representation, and uses it to predict the final earthquake category. A high-level illustration of DMSEEW is presented in Figure 1.

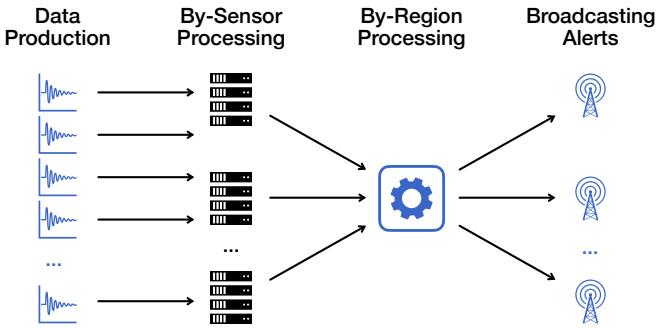



Fig. 1: An illustration of the Distributed Multi-Sensor Earthquake Early Warning use-case (DMSEEW) [12]. Seismic sensors located in the Edge send measurements to gateways in the network which pre-process the data. Those pre-processed data are sent to cloud servers which complete the data processing and eventually broadcast earthquake alerts.

### C. Leveraging the Computing Continuum

The Computing Continuum represents a fluid integration of the computational, storage, and network resources located at the edges, in the cloud, and in-between [1]. The data is generated at the edges by sensors, scientific instruments, and personal devices. Edge devices are usually limited in computational power and storage; their principal function is to collect data and transmit them for analysis. In-transit nodes are then in charge of performing aggregation, filtering, or preprocessing along the data path. Finally, far from the data, the cloud provides the abstraction of unlimited resources in well-provisioned datacenters.

Traditional solutions focus on one particular dimension at a time. Hence, most of the solutions are built on the

premise that data ingestion, management, and processing can be done in the cloud, without the use of edge and in-transit tiers. Several edge-based middleware solutions exist in the literature; however, they often lack a uniform programming model framework for resource management, data processing, and application servicing between edge and cloud [13], [14].

Choosing where to execute a function in a heterogeneous and distributed system depends on a large number of factors ranging from the performance of a given implementation on a given resource to the variability of the data input size, network links, and computing resources. Figure 2 shows the difference between an "all-cloud" or "all-edge" placement of the workflows for the dataset used in our evaluation (see Section IV). Beyond the gap in performance due to the different capacities of the resources, this result assumes that resources are available at full capacity during the lifecycle of the application, which rarely occurs in practice. This motivates the need for adaptive policies that dynamically (and opportunistically) use a combination of available resources between the edge and the cloud to provide performance guarantees and reactions to unforeseen events.

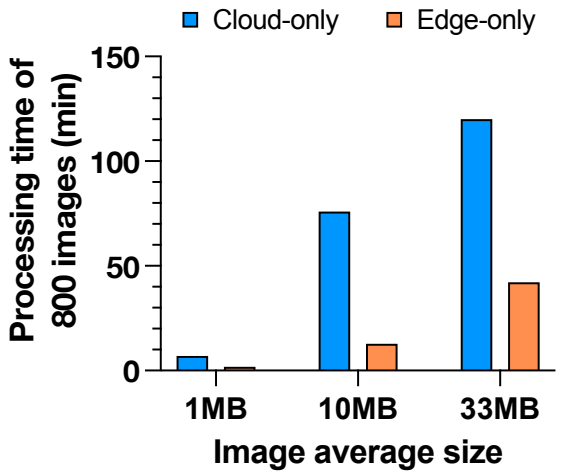



Fig. 2: Processing time for the disaster response dataset (800 images) using either 8-core Cloud or single core Edge resources.

### D. Challenges

Leveraging the computing continuum to support distributed analytics and urgent workflows effectively leads to multiple challenges.

**Extreme heterogeneity and uncertainty.** The continuum represents extreme heterogeneity in the capabilities and capacities of systems and services. Addressing this heterogeneity requires application formulations and programming abstractions that allow developers to expose natural flexibilities and tradeoffs and define policies and mechanisms that can drive runtime adaptions.

**Balancing requirements and expectations with constraints and cost.** Mapping user expectations and constraints in terms of response time, solution quality, data resolution, cost, energy, etc. with what is possible in a dynamic computation and communication environment during execution warrants the design of an autonomic control plane. Such a control plan must be able to address cost/benefit tradeoffs at runtime in a cross-layer manner and drive the autonomic reconfiguration of applications, resources, and services.

**Ensuring time-critical decision making.** While the seamless and ephemeral composition of data, resources and services towards enabling novel and impactful application is compelling, it also makes providing end-to-end guarantees critical and challenging.

## III. System Stack

In this section, we briefly describe our unified framework for continuum computing designed to enable data-driven workflows. The framework provides the essential capabilities and services that are necessary for executing workflows on an infrastructure that integrates resources across the continuum. These services include resource discovery, mapping of computations to resources, and runtime management and control. Furthermore, at the application level, the framework enables developers to express requirements and constraints associated with stages of the workflow. Prior work on this framework has been presented in [2].

An overview of the framework architecture is presented in Figure 3 and is described below. This architecture targets end-to-end applications that are implemented as a workflow of services spanning data producers located at the edges, in-network resources, and data consumers. An application consists of a workflow of services (represented as different shapes) that match user expectations and quality tradeoffs with their available service implementation (represented as different sizes). It is beneficial for the application logic to exploit the available configurations of the infrastructure and influence the operator placement and service configurations. For example, if all applications use only core services, congestion and load balancing issues may affect the overall performance of the system.

The framework allows for the separation of concerns between the domain scientists (familiar with the applications) and the DevOps (familiar with the platforms). By describing the flexibilities and constraints of the applications (particularly the reaction to adopt with regards to the relative state of resources), the domain scientist allows for operations of deployment and reconfiguration that considers variables of latency, quality of service, and throughput during the lifecycle of an application.

### A. Building blocks

**R-Pulsar** is an edge-based middleware for enabling data-driven workflows that span edge and cloud resources. It proposes an event-driven programming model based on control-flow constructs to facilitate the definition of data-processing

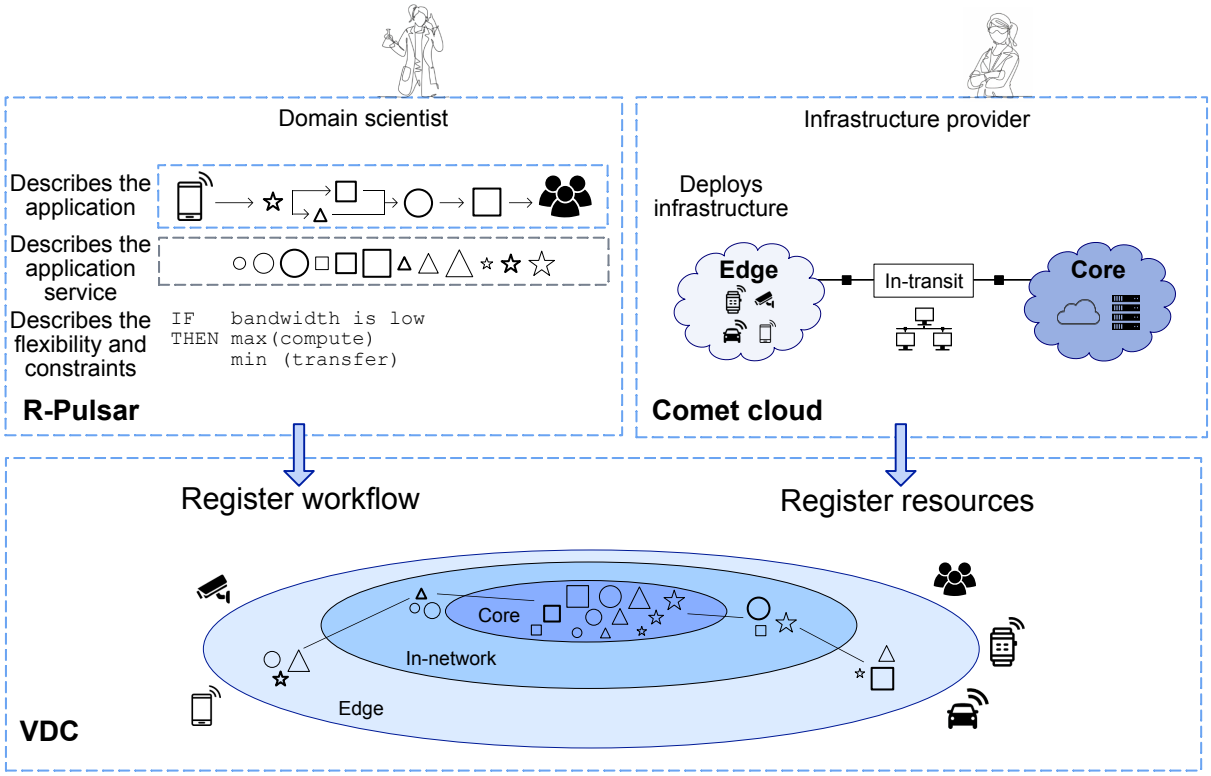



Fig. 3: High-level view of the unified framework. The domain scientist is in charge of describing the flexibilities and constraints of the application. The infrastructure provider deploys resources and instantiates services.

requirements. R-Pulsar allows developers to trigger topologies based on the availability of resources and content of data [14]. Triggers can be values, trends (statistical), etc. This allows the orchestration of computations through user-defined rules for deciding what, when and where data get collected and processed. R-Pulsar has been shown to improve metrics such as end-to-end latency, bandwidth consumption, and messaging costs for IoT-based workflows [15].

**The Virtual Data Collaboratory (VDC)** is a data-intensive cyberinfrastructure focused on providing large-scale federated data management, analysis resources, and tools for scientific applications [16]. VDC leverages regional data transfer nodes (DTNs) and federated data resources to support data services and enable collaborative data-intensive workflows. In the context of this work, VDC exposes data producers and containerized agents for virtual and physical resources.

**CometCloud** enables the dynamic composition of infrastructure services across multiple (academic and commercial) providers. It enables on-demand scale-up, scale-down, and scale-out based on dynamic deadline-, budget-, and workload-based constraints in a multi-cloud environment [17], [18].

## IV. Evaluation Methodology

To illustrate the performance and reactivity of our system stack, we used it to enable a comparative analysis of the five adaptative policies using scenarios associated with the variability of resources across the Computing Continuum. The goal of this evaluation is to show that: *a)* Our framework can provide programming support and resource management for data-driven workflows in real environments *b)* Our framework is able to identify trade-offs between the quality and cost of computations *c)* Our framework reacts to the variability of the environments and allows for the execution of flexible user-defined policies

### A. Implementation aspects

*a) Experimental platform:* In order to deploy the different components of the framework and run the experiments on top of Grid'5000 [19], we rely on EnosLib [20], a library able to set up the environment, deploy configurations and gather metrics and results. This library was designed for experiment-driven research and facilitates the repeatability of the experiments so that comparison between the different policies remains fair.

### B. Baselines

The default Cloud and Edge policies, acting as baselines in our system, are further denoted as Cloud-only and Edge-only and will be compared to three different policies: *Continuum*, *Dynamic*, *Deadline*. They are all defined using our framework but differ in their objective functions.

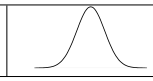
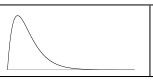
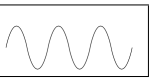
| | Data Input size | Bandwidth | CPU capacity |
|---|---|---|---|
| Distribution |  |  |  |

TABLE I: Probability distributions of the data input size, bandwidth and CPU capacity values during the execution.

*a) Continuum:* This policy maximizes the utilization of edge resources, network links, and cloud resources.

*b) Dynamic:* The objective of this policy is to maximize the utilization of resources while using approximation mechanisms to render products within acceptable ranges of accuracy.

*c) Deadline:* The objective of this policy is to enforce an arbitrary deadline. It allows for approximations mechanisms, sampling techniques, and allows for discarding of data when necessary.

We compare the performance of the five policies in four different setups. Each setup represents a different environment in terms of the variability of the data input size, the available bandwidth between the edge and core, and the available processor capacity at the edge, as presented in Table I.

*d) Constant environment:* This environment models a fixed input size, where the images are the biggest available in the dataset (33.8 MB). Data transfers and processing operations at their maximum capacity.

*e) Variability in the data input size:* This environment models the variation in data input size that occurs at runtime. We use a normal distribution with expectations of [33, 3.3, 1] representing 3 different size factors of the data, and a standard deviation of 0.5.

*f) Variability in the data input size and available bandwidth:* This environment models two variations: the input size as described below, and the available bandwidth. We use a Chi-Squared distribution that results in a constrained bandwidth early in the experiments before slowly recovering towards the maximum bandwidth available.

*g) Variability in the data input size, available bandwidth, and CPU capacity:* This environment models the two variations described below and the sinusoidal distribution of the CPU capacity, oscillating between 30 and 70%. This aims at modeling the cyclic activity of an edge resource when periodically running the decision engine/processing events, and waiting for events.

### C. Environment

We implemented the architecture described in Section III and deployed it on the *nova* cluster from the Grid'5000 experimental testbed. Among these nodes, we emulate one edge node with 1 core and one core node with 8 cores. The capacity of each type of node is described in Table II. The network connection between nodes of the same and different types have a delay and bandwidth as given in Table III. EnosLib enables, among others, the scripting of network emulation features that allows the specification of Edge-to-Cloud communication constraints (delay, loss, and bandwidth).

The Edge-to-Cloud network characteristics emulate a 4G mobile network configuration. We describe in Table III the values of bandwidth, minimum latency, and jitter of our network configuration.

| Layer | Cores | RAM (GiB) | Storage |
|---|---|---|---|
| Edge | 1 | 2 | 2GB |
| Cloud | 8 | 64 | 500GB |

TABLE II: Resource capacity of Edge and Cloud nodes.

### D. Performance Metrics

Each experiment gathers various metrics from the infrastructure. We introduce the used performance metrics.

*The time to completion* measures in minutes the time elapsed between the production of the first image of a given batch by the sensor and the processing of the data (raw file or summary) associated with the last image of the same batch in the cloud. This metric contains the processing time on each type of resource and the Edge-to-Cloud transfer time.

*The compute time* measures in minutes the time spent processing the images of a batch. It Is a sum of any processing occurring on any resource of the Continuum. It also contains the overhead associated with the decision-making process of the policies.

*The data transfer time* measures in minutes the time spent transferring data across the Continuum.

*The data quality* quantifies the variation of average system F1-score when using approximation mechanisms.

*The acceptance rate* quantifies the number of images that were processed in a given batch. This metric only applies to the deadline policy where data products can be discarded at the source to help enforce the user-defined deadline.

The following section presents the validation of our system stack and the performance of the different policies using a real workload (see Section 2).

### E. Datasets

We use data from the Hurricane Sandy dataset, extracted from the office for coastal management [1]. The goal of this application is to provide emergency services to quickly and efficiently determine whether building conditions are safe for evacuees to return after a natural disaster. First, the workflow captures LiDAR images of the affected zones using a drone, then it pre-processes the images to determine the appropriate response. Finally, if needed, a change analysis is performed using historical data pulled from the cloud.

We process the images using a customized implementation of the Canny edge detection algorithm. The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images [21]. The Canny edge detection algorithm can be broken down into 5 steps: *Noise reduction*; *Gradient calculation* to detect the edge intensity and direction; *Non-maximum suppression* to thin out

[1] https://chs.coast.noaa.gov/htdata/lidar1$_z$/$geoid18$/$data$/1436/

| | Bandwidth | Minimum latency | Jitter |
|---|---|---|---|
| **4G** | 15Mbps | 100 ms | 400ms (normal) |

TABLE III: Edge-to-Cloud network configuration used for the evaluation

the edges; *Double threshold* to discard weak or non-relevant pixels, and edge Tracking to highlight strong pixels.

This generic workflow and synthetic dataset is representative of the Earthquake Early Warning motivating use case. However, we chose the Canny edge algorithm due to its ease to apply approximation through substitution (using simpler tasks) and discarding (not executing a subset or certain redundant tasks), which can be applied to different tasks (both at function and input parameters) [22]. This facilitates the reduction in quality of the different images and the deployment on constrained hardware thanks to the numerous implementations available.

### F. Experimental setup

Each experiment of the evaluation consists of: *(i)* deploying the system stack and necessary infrastructure on top of Grid'5000, *(ii)* configuring the network, and the machines, *(iii)* deploying the application on that infrastructure, *(iv)* running the application for 30 batches of images, and *(v)* gathering logs and metric measurements. These steps are repeated for each of the five policies. Hence, in total, the evaluation consists of 20 experiments. Each experiment is run 10 times.

## V. Evaluation Results

We first assess the performance of the policies in a constant environment. Then, we successively add variations in the data input size, bandwidth, and CPU capacities, highlighting the dynamicity that occurs at runtime across the computing continuum. Finally, we dissect the behavior of each policy in four different cases.

### A. Constant Environment

We start by presenting the performance of Cloud-only and Edge-only, the bound policies that are used as a basis of comparison to the other policies throughout this section. The main characteristic of those scenarios is that placement of functions and data transfers are unchanged during the execution. Cloud-only and Edge-only will, respectively, always transfer then compute in the core or compute all data products at the edge then send a summary of findings at the core. Due to that, the main performance bottleneck for these policies will be the CPU capacity at the edge, and the available bandwidth across the continuum, as observed in Figure 4. It is worth mentioning that the summary of findings only consists of the number of potentially damaged buildings. The size of the summary is voluntarily negligible compared to the transfer of an image, generating less data load on the network. All scenarios present a constant performance, in accordance with the lack of injected variability in the environment. Notice that the scale is different on the Cloud-only results: it takes close

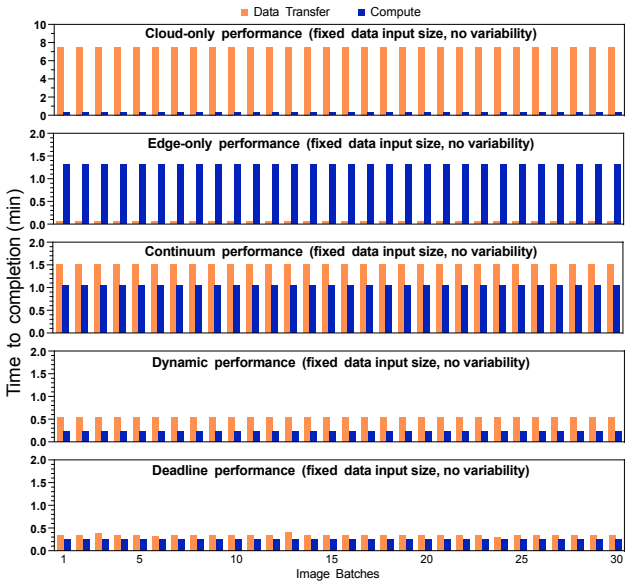



Fig. 4: Time to completion resulting from processing 30 batches of images on each policy with a constant environment.

to 8 minutes for each batch, while the Edge-only policy needs close to 1.3 minutes to process a batch, both relying mainly on one type of resource. The Continuum policy balances the usage of data transfers and edge computations with a slightly higher completion time per batch. The clustered columns represent the time spent (in parallel) on each operation, independently of the location of resources. The dynamic policy presents an improvement of 40% compared to the Edge-only baseline. The repartition of operation is explained by the pre-processing of images to reduce their quality to the minimum acceptable level (I.e. 60%). Finally, the deadline policy is set with the hard constraint of 30 seconds to compute each batch of images. This forces the decision-making engine to, in order, use the minimum quality possible, and then, reduce the number of images processed in each batch. Many class of applications tolerates sampling or are tolerant to errors, but the interactions required to guarantee the proper functioning are out of the scope of this article. We implement this policy as a proof of concept to show the versatility of our decision-making engine and its ability to beat the user-defined deadlines in an edge environment. In our context, images are discarded strictly based on estimated computations and transfer times of a given batch.

### B. Variable Environment

The second set of experiments introduces variability in the data input size, bandwidth, and CPU capacities. We show that the performance is directly affected by the environment and that our online decision-making engine is able to feed the policies with ongoing events to adapt the function placement and data quality to the objectives. The extent of that adaptation

scheme is further discussed using the results of the continuum, dynamic, and deadline policies.

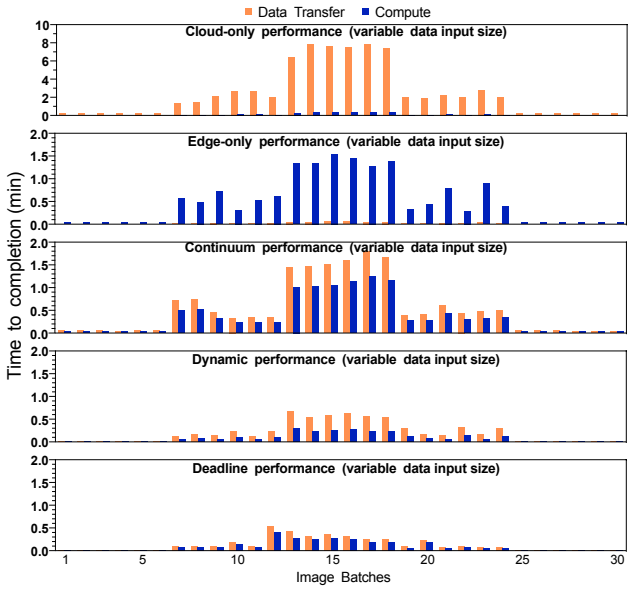



Fig. 5: Time to completion resulting from processing 30 batches of images on each policy with a variable input data size.

*1) Input Size:* We illustrate the input data size scenario in Figure 5. The main observation from this set of charts is the overall average in performance which resembles the normal distribution of the input data size. We observe slight degradations in all scenarios due to the online processing of events (less than 5% on all experiments.) When dealing with smaller images, all policies have similar behavior and performance due to the overprovisioning of resources. As the size of images fluctuates, the continuum policy maintains a maximization of resource utilization by constantly adjusting the number of raw images to be sent in order to use the network links at capacity. The dynamic and deadline policies react well to the changes of data size, however, we observe two spikes for batches 11 and 12 for both policies. Particularly, on the deadline policy, we observe a spike on batch 11 while it does not happen on the next batches with similar or greater input size. Similar behavior is observed on batch 12, however, these appear to be outliers among the other data points. Overall, the system adjusts well when the environment presents frequent input size changes during the execution of the application.

*2) Bandwidth capacity:* We illustrate the bandwidth capacity scenario in Figure 6. This scenario cumulates the input size changes described previously with a network contention scenario early in the experiment. Starting with the Cloud-only policy, the performance strongly resembles the distribution of network capacity events due to the "forward-then-compute" nature of the policy. The Edge-only scenario indicates no effect from the network contention: the edge resources continue

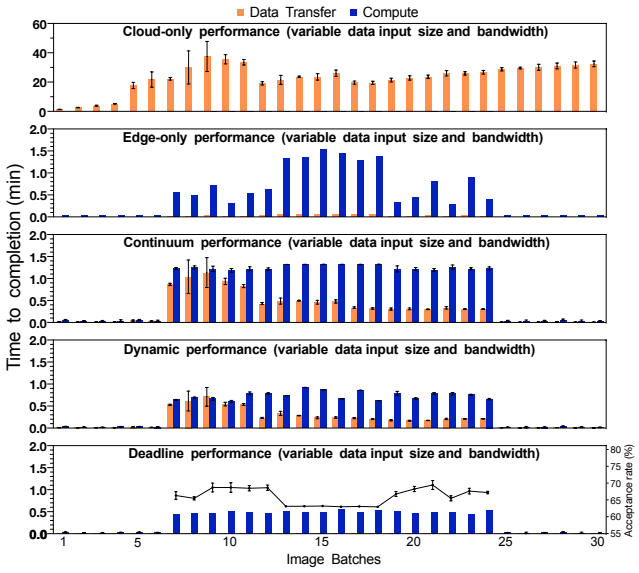



Fig. 6: Time to completion for processing 800 images on each policy with variable input data size and bandwidth capacity.

to operate, reducing data products to summary in order to cope with the minimal bandwidth available. Because of the limited network. the continuum policy tends to rely more on edge resources, resulting in an improvement of the overall performance compared to the previous experiment with little overhead. Similar behavior is observed for the dynamic policy, while the deadline relies on edge resources, approximation, and discarding to enforce the deadline. We observe 6 violations of deadline between 2% and 6%, with an average acceptance rate of 35%. At this particular point, the decision-making engine seems to fail to properly estimate the time required to pre-process this new batch of images, resulting in some miscalculations for bounding the operations. As the network is too constrained, many images are dropped resulting in fewer images processed and less information sent.

*3) CPU capacity:* We illustrate the CPU capacity scenario in Figure 7. This scenario cumulates the input data size and bandwidth variations, along with an external cyclic utilization of edge resources. This last scenario aims at modeling the variability and uncertainty of the Computing Continuum. This results in a higher number of events to manage, but also more complex decisions to apply for the policies. Overall, we observe a significant performance degradation for every policy. While Cloud-only performance remains unchanged due to no use of edge resources, Edge-only, and Dynamic observe more than 200% of performance degradation. However, we can observe that the Dynamic policies adjust quickly to the periodic variations of the CPU. The deadline policy remains able to enforce the user-defined deadline (5 violations of deadline between with a maximum of 10.4%) with an acceptance rate comprised between 47.27% and 53.21%.

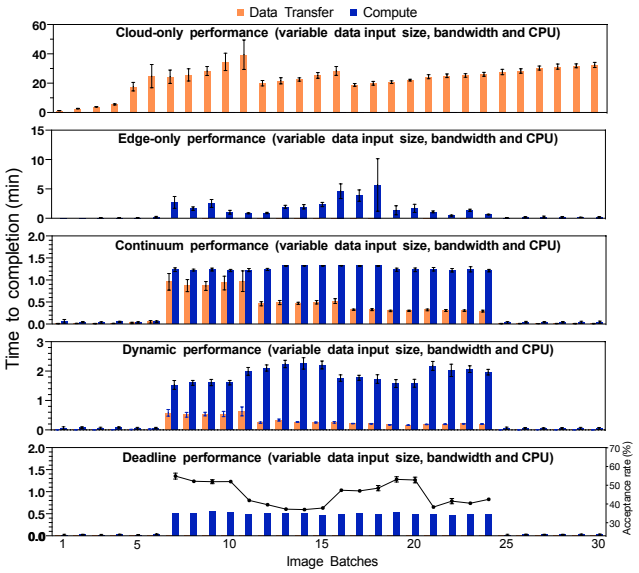



Fig. 7: Time to completion resulting from processing 30 batches of images on each policy with variable input data size, bandwidth capacity and CPU capacity

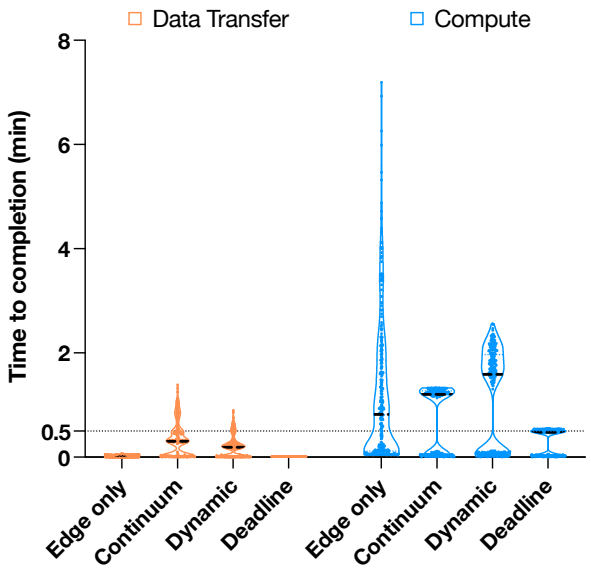



Fig. 8: Repartition of data points for each policy. Closer data points for a given policy indicate a low variability of performance.

In Figure 8, we plot the performance associated to each experiments using a violin/box-plot representation. One can notice the consistency of the deadline policy among highly variable environments. On the other hand, there is a vast distribution of points in the Cloud-only policies, only confirming that the Cloud alone can not provide guarantees of performance or deadlines under a high variability of resources. These results help us clarify the role of the programming support to implement resources during design, and the resource management to cope with variability at runtime. Despite using a sequential approach to manage events within time windows, our system stack enables the execution and adaptation of the application under a high variability of events.

### C. Discussion

We can extrapolate some insights and takeaways from this evaluation to the Earthquake Early Warning use case. The main takeaway from the performance evaluation of these five policies is the importance of the synergy between workloads, applications, and infrastructures. Another important insight refers to the difference between Deadline and Dynamic scenarios. If the application can tolerate the observed acceptance rates, Edge can be an interesting option in particular for scenarios where the network is very constrained such as in the 2G and 3G network infrastructures, or under intermittent connectivity. It is important to highlight that, as real-time is an essential constraint for EEW systems, in this work, we focus on the latency of the applications for justifying the proposed Continuum computing approach. Nevertheless, as discussed in other works , relying on the computing continuum may also add features such as resilience, privacy, or network cost reductions.

## VI. Related Work

The Computing Continuum aggregates the architectural and algorithmic challenges of its subcomponents while presenting new challenges related to their integration and overall management [1], [2]. As data analytics based on AI/ML techniques are becoming an increasingly important component of data-driven application workflows, several studies have identified model optimization and ML inference as the main vectors driving the use of resources at the edge [23], [24]. Furthermore, execution mechanisms that leverage data parallelism (partitioning data among units) and model parallelism (distributing the intelligence among units) have proven effective for processing data streams [25], [26].

*a) Edge-enhanced architectures and system:* While many edge-based stream analytics systems focus on dead-line driven processing [27], bandwidth-limited scheduling [28], [29] and real-time processing [30], [31], major stream processing frameworks rely on data being moved to the cloud and are often agnostic to the specific requirements of devices [32]. Collaborative approaches for inference leverage hybrid edge-to-cloud infrastructures based on constraints such as the size of input data, the model to be executed, and trade-offs between the inference accuracy and network latency and bandwidth [33], [34]. Other graph-based approaches track pipelines and map them to geographically distributed analytic engines ranging from small edge-based engines to powerful multi-node cloud-based engines [35], [36]. Finally, distributed deep learning has motivated the use of computing hierarchies by splitting neural network [37] or using sparse updates [38]

to aggressively reduce communication costs. However, these works are either not resilient/sensitive to topology changes [39] or rely on unsuitable resource management features for service delivery [40].

*b) Resource management:* Current resource management systems for automating deployment, scaling, and management of applications are not equipped to support the extreme uncertainty arising from variability in the availability and quality of data, resources, and services [37]. Several initiatives exist in the literature to manage massively distributed resources [41], [42], [43], but they either operate edge infrastructures as data center environments with WAN links or consist in deploying hierarchical managers [44].

*c) Meeting System Level Objectives (SLOs):* Recent work in Function-as-a-Service research have investigated function placement and optimization in heterogeneous computing environment. Kumar and al. propose Delta, a high-level scheduler able to profile function performance using predictive models [45]. Their approach present significant potential for executing complex tradeoffs under different connectivity and computational capabilities. Other works aims at maximizing utilization of serverless functions by regulating the resource usage of executions when task load increases [46] or using historical data to predict whether a task will meet its objectives [47]. While these works deal with aspects of efficient function scheduling in heterogeneous environments, their focus is not on managing user expectations.

Our work attempts at articulating the extreme heterogeneity and uncertainty of resources with the requirements and constraints of data-driven applications. Prior works are either application-specific and lack abstractions for expressing objectives for time-critical operations or focused on a single environment. The proposed work for programming flexible analytics aims for a general approach with few assumptions on the data and resources capabilities.

## VII. Conclusion

Our society relies intensively on digital technologies for decision-making that impact our economy, culture, and lifestyle in several domains. The computational ecosystem that supports these analytics has become highly heterogeneous and geographically distributed, bringing significant challenges associated with the complexity and sustainability of distributed analytics.

In spite of these important advancements, there is still a critical gap in the knowledge base that pertains to the loosely coupled solutions that are enabling developers to express what data and services to run, where to run them, and how to run them across the Computing Continuum. These limitations prevent developers from reasoning on massively distributed resource capabilities without prior or advanced knowledge of the targeted infrastructure dynamics.

In this paper, we present a framework for deploying and evaluating policies between edge and cloud resources. We show that our approach is able to cope with the variability of the Computing Continuum, and adapt the configuration of flexible data flows, allowing their customization to fit the computing resources' capabilities, dynamic application requirements, and desired quality and performance objectives.

This model can be extended to other applications in which adaptation levers and flexibilities exist, and presents encouraging results for tackling distributed data-driven analytics across the continuum. We are now working on a probabilistic approach of the decision-making model that could compensate for missing or incomplete data at runtime. Also, we are looking at the integration of logging and metrics management into our rule-based engine for more diverse adaptation policies.

## Acknowledgements

This research is supported in part by the NSF under grants numbers OAC 1640834, OAC 1835692, and OCE 1745246. The research was conducted as part of the SCI Institute at the University of Utah.

## References

[1] P. Beckman, J. Dongarra, N. Ferrier, G. Fox, T. Moore, D Reed, and M. Beck. Harnessing the computing continuum for programming our world, 2019.

[2] D. Balouek-Thomert, E. G. Renart, A. R. Zamani, A. Simonet, and M. Parashar. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. *The International Journal of High Performance Computing Applications*, 33(6):1159–1174, November 2019.

[3] K. Winter. For Self-Driving Cars, There's Big Meaning Behind One Big Number: 4 Terabytes.https://newsroom.intel.com/editorials/self-driving-cars-big-meaning-behind-one-number-4-terabytes/, 2017.

[4] Siew Hoon Leong and Dieter Kranzlmüller. Towards a general definition of urgent computing. *Procedia Computer Science*, 51:2337 – 2346, 2015. International Conference On Computational Science, ICCS 2015.

[5] Gordon Gibb, Rupert Nash, Nick Brown, and Bianca Prodan. The technologies required for fusing hpc and real-time data to support urgent computing. In *2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC)*, pages 24–34. IEEE, 2019.

[6] R. M. Allen, P. Gasparini, O. Kamigaichi, and M. Böse. The Status of Earthquake Early Warning around the World: An Introductory Overview. *Computing in Science Engineering*, 80(5):682–693, 2009.

[7] K. Doi. The operation and performance of Earthquake Early Warnings by the Japan Meteorological Agency. *Soil Dynamics and Earthquake Engineering*, 31(2):119–126, 2011.

[8] J. M. Espinosa-Aranda, A. Cuellar, A. Garcia, G. Ibarrola, R. Islas, S. Maldonado, and F. H. Rodriguez. Evolution of the Mexican Seismic Alert System (SASMEX). *Seismological Research Letters*, 80(5):694–706, 2009.

[9] Yih min W. and Ta liang T. A virtual sub-network approach to earthquake early warning. *Bull. Seism. Soc. Am*, pages 2008–2018, 2002.

[10] K. Rochford, J. A. Strauss, Q. Kong, and R. M. Allen. Myshake: Using human-centered design methods to promote engagement in a smartphone-based global seismic network. *Frontiers in Earth Science*, 6:237, 2018.

[11] M. D. Kohler, D. E. Smith, J. Andrews, A. I. Chung, R. Hartog, I. Henson, D. D. Given, R. de Groot, and S. Guiwits. Earthquake Early Warning ShakeAlert 2.0: Public Rollout. *Seismological Research Letters*, 91(3):1763–1775, 04 2020.

[12] K. Fauvel, D. Daniel Balouek-Thomert, D. Melgar, P. Silva, A. Simonet, G. Antoniu, A. Costan, V. Masson, M. Parashar, I. Rodero, and A. Termier. A Distributed Multi-Sensor Machine Learning Approach to Earthquake Early Warning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[13] Bin Cheng, Gürkan Solmaz, Flavio Cirillo, Ernö Kovacs, Kazuyuki Terasawa, and Atsushi Kitazawa. Fogflow: Easy programming of iot services over cloud and edges for smart cities. *IEEE Internet of Things Journal*, 5(2):696–707, 2017.

[14] E. G. Renart, D. Balouek-Thomert, and M. Parashar. An edge-based framework for enabling data-driven pipelines for iot systems. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 885–894, 2019.

[15] E. G. Renart, A. da Silva Veith, D. Balouek-Thomert, M. Dias de Assuncao, L. Lefèvre, and M. Parashar. Distributed Operator Placement for IoT Data Analytics Across Edge and Cloud Resources. In *CCGrid 2019 - 19th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing*, pages 1–10, Larnaca, Cyprus, May 2019.

[16] M. Parashar, A. Simonet, I. Rodero, F. Ghahramani, G. Agnew, R. Jantz, and V. Honavar. The Virtual Data Collaboratory: A Regional Cyberinfrastructure for Collaborative Data-Driven Research. *Computing in Science Engineering*, 22(3):79–92, May 2020.

[17] J. Diaz-Montes, M. AbdelBaky, M. Zou, and M. Parashar. Cometcloud: Enabling software-defined federations for end-to-end application workflows. *IEEE Internet Computing*, 19(1):69–73, 2015.

[18] M. AbdelBaky and M. Parashar. A general performance and qos model for distributed software-defined environments. *IEEE Transactions on Services Computing*, pages 1–1, 2019.

[19] Desprez and al. Adding virtualization capabilities to the grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, pages 3–20, Cham, 2013. Springer International Publishing.

[20] Ronan-Alexandre Cherrueau, Marie Delavergne, Alexandre Van Kempen, Adrien Lebre, Dimitri Pertin, Javier Rojas Balderrama, Anthony Simonet, and Matthieu Simonin. EnosLib: A Library for Experiment-Driven Research in Distributed Computing. *IEEE Transactions on Parallel and Distributed Systems*, September 2021.

[21] William McIlhagga. The canny edge detector revisited. *International Journal of Computer Vision*, 91(3):251–261, 2011.

[22] Parul Pandey and Dario Pompili. Exploiting the untapped potential of mobile distributed computing via approximation. *Pervasive and Mobile Computing*, 38:381–395, 2017. Special Issue IEEE International Conference on Pervasive Computing and Communications (PerCom) 2016.

[23] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[24] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and Faraz Hussain. Machine learning at the network edge: A survey, 07 2019.

[25] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and Ng Andrew. Deep learning with cots hpc systems. In *International conference on machine learning*, pages 1337–1345, 2013.

[26] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[27] Ali Reza Zamani, Mengsong Zou, Javier Diaz-Montes, Ioan Petri, Omer Rana, Ashiq Anjum, and Manish Parashar. Deadline constrained video analysis via in-transit computational environments. *IEEE Transactions on Services Computing*, 13(1):59–72, 2017.

[28] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. Edge analytics in the internet of things. *IEEE Pervasive Computing*, 14(2):24–31, 2015.

[29] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 426–438, 2015.

[30] Camille Bailas, Mark Marsden, Dian Zhang, Noel E O'Connor, and Suzanne Little. Performance of video processing at the edge for crowd-monitoring applications. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 482–487. IEEE, 2018.

[31] Peng Liu, Bozhao Qi, and Suman Banerjee. Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st international workshop on edge systems, analytics and networking*, pages 1–6, 2018.

[32] E. G. Renart, D. Balouek-Thomert, and M. Parashar. Challenges in designing edge-based middlewares for the internet of things: A survey, 2019.

[33] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1421–1429, 2018.

[34] Seungyeop Han and al. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *14th Annual Intl. Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 123–136, 2016.

[35] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu. Deepwear: Adaptive local offloading for on-wearable deep learning. *IEEE Transactions on Mobile Computing*, 19(2):314–330, 2020.

[36] Nisha T. and al. ECO: Harmonizing edge and cloud with ml/dl orchestration. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, 2018.

[37] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 328–339. IEEE, 2017.

[38] Zeyi Tao and Qun Li. esgd: Communication efficient distributed deep learning on the edge. In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[39] Zhenyu Wen, Pramod Bhatotia, Ruichuan Chen, Myungjin Lee, et al. Approxiot: Approximate analytics for edge computing. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 411–421. IEEE, 2018.

[40] Hyunseok Chang, Adiseshu Hari, Sarit Mukherjee, and TV Lakshman. Bringing the cloud to the edge. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 346–351. IEEE, 2014.

[41] Open Source Edge Cloud Computing Architecture - StarlingX, Nov 2020. [Online; accessed 1. Nov. 2020].

[42] KubeEdge. KubeEdge, Nov 2020. [Online; accessed 1. Nov. 2020].

[43] kubernetes sigs. kubefed, Nov 2020. [Online; accessed 1. Nov. 2020].

[44] Ronan-Alexandre Cherrueau, Marie Delavergne, Adrien Lebre, Javier Rojas Balderrama, and Matthieu Simonin. *Edge Computing Resource Management System: Two Years Later!* PhD thesis, Inria Rennes Bretagne Atlantique, 2020.

[45] Rohan Kumar, Matt Baughman, Ryan Chard, Zhuozhao Li, Yadu Babuji, Ian Foster, and Kyle Chard. Coding the computing continuum: Fluid function execution in heterogeneous computing environments. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 66–75. IEEE, 2021.

[46] Amoghvarsha Suresh and Anshul Gandhi. Fnsched: An efficient scheduler for serverless functions. In *Proceedings of the 5th International Workshop on Serverless Computing*, pages 19–24, 2019.

[47] Chavit Denninnart, James Gentry, and Mohsen Amini Salehi. Improving robustness of heterogeneous serverless computing systems via probabilistic task pruning. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 6–15. IEEE, 2019.