Harnessing the Computing Continuum for Urgent Science

Daniel Balouek-Thomert
Rutgers Discovery Informatics Institute (RDI^2)
daniel.balouek@rutgers.edu

Ivan Rodero
Rutgers Discovery Informatics Institute (RDI^2)
irodero@rutgers.edu

Manish Parashar
Rutgers Discovery Informatics Institute (RDI^2)
parashar@rutgers.edu

ABSTRACT

Urgent science describes time-critical, data-driven scientific workflows that can leverage distributed data sources in a timely way to facilitate important decision making. While our capacity for generating data is expanding dramatically, our ability to manage, analyze, and transform this data into knowledge in a timely manner has not kept pace. This paper explores how the computing continuum, spanning resources at the edges, in the core, and in-between, can be harnessed to support urgent science and discusses associated research challenges. Using an Early Earthquake Warning (EEW) workflow, which combines data streams from geo-distributed seismometers and high-precision GPS stations to detect large ground motions, as a driver, we propose a system stack that can enable the fluid integration of distributed analytics across a dynamic infrastructure spanning the computing continuum.

KEYWORDS

Computing continuum, workflows, machine learning, edge computing, scheduling, urgent science

ACM Reference Format:
Daniel Balouek-Thomert, Ivan Rodero, and Manish Parashar. 2020. Harnessing the Computing Continuum for Urgent Science. In *DCC'20: ACM Workshop on Distributed Cloud Computing, June 08, 2020, Boston, MA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 including those based on Artificial Intelligence and Machine Learning (AI/ML) has the potential to enable end-to-end workflows that combine these elements to 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.

In our earlier work, we proposed the notion of a **computing continuum** based on the fluid integration of resources at the edge, in the core, in the network, and along the data path to support dynamic and data-driven application workflows [1]. Key research questions addressed by this prior work include: (1) how do we determine what, where, and when data gets collected and analyzed based on user objectives and user/system constraints; (2) how do we program services to respond to changes in application objectives, data availability and content; and (3) how does the application react during execution to events of interest and trigger appropriate actions.

Leveraging this vision of the computing continuum for urgent science requires novel application and algorithmic formulations, as well as a software stack that provides programming abstractions and runtime mechanisms for dynamically (and opportunistically) federating appropriate resources, and composing data-driven workflows where workflow composition and execution is defined by the content and location of the data. Furthermore, the execution must continually balance the computation or data movement's cost with the value of the operations to the application objectives.

In this paper, we build on our prior research and envisioned software stack presented in [1] and explore how the computing continuum can be leveraged to support a specific urgent science workflow, i.e., the data-driven Earthquake Early Warning (EEW) application workflow. This workflow combines data streams from geo-distributed seismometers and high-precision GPS stations to detect large ground motions, which can lead to hazards such as tsunamis. Early warnings rely on a two-step ML algorithm that has been validated using real data. The presented software stack allows users to program reactive behaviors and orchestrate their execution across the continuum while balancing computational costs with the application's value.

The rest of this paper is organized as follows. Section 2 presents key research challenges in realizing urgent science and the EEW workflow across the continuum. Section 3 introduces the concept of continuum computing and provides background and related work. Our unified framework is introduced in Section 4, and Section 5 illustrates how it is used to support the EEW workflow. Finally, we discuss open research challenges in Section 6 and conclude the paper.

DCC'20, June 08, 2020, Boston, MA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 URGENT SCIENCE AND THE EARTHQUAKE EARLY WARNING USE CASE

2.1 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 [2]. 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 [3, 4].

2.2 Earthquake Early Warning

Earthquakes are amongst the most destructive natural disasters. Networks of distributed seismic instruments on various scales are used for earthquake detection. When an earthquake occurs, a real-time network system installed to monitor seismic activity in a region can detect it. Typically, a centralized data collection and processing agency alerts relevant stakeholders. This centralized data collection and processing limit the amount of data that can be processed and how fast it can be processed. It motivates leveraging the computing continuum. The goal of distributing the processing is to speed-up detection while reducing data transfer costs (sending only aggregated data to the cloud), fault tolerance (avoid isolation of sensors), and better-utilizing resources (using resources at the edge and in-transit).

When an earthquake strikes, the energy released is sent out as seismic waves in all directions. The fastest waves, the waves that arrive first, do not actually shake the ground in a damaging way. We cannot even feel these primary or P-waves, but sensors can. P-waves travel through Earth's crust around 1.7 times faster than secondary or shear waves (S-waves), propagating through Earth's interior and causing the ground to shake back and forth. Thus, there is a lag between these two types of waves, and using them together allows us to generate warning before the shaking starts and the resulting damage.

Earthquake Early Warning (EEW) systems aim to provide alerts before any damaging effects reach sensitive areas by detecting P-waves before the S-waves arrive. It gives communities, organizations, and governments a valuable time window to take protective actions. Warnings can be sent out to mobile phones, radio, and TV. They can stop traffic and bridges, halt surgeries, and make decisions on critical infrastructures such as gas pipelines. EEW systems usually use hundreds to thousands of sensors scattered in different geographical regions producing 3D *time series*, which indicates the direction of the P-wave ground-motion (east-west, north-south and up-down).

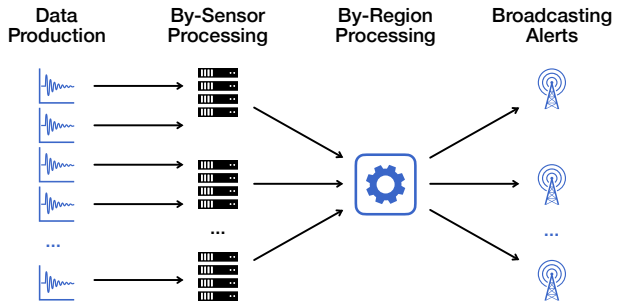

Figure 1: An illustration of the Distributed Multi-Sensor Earthquake Early Warning use-case (DMSEEW) [11]. Seismic sensors located in the Edge send measurements to gateways in the network which pre-process the data. Those pre-processed data are sent to cloud servers which complete the data processing and eventually broadcast earthquake alerts.

At present, there are EEW systems operational in several countries [5]. 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 [6]. Mexico implemented a seismic alert system in 1991, allowing for a warning time of 58 to 74 seconds [7]. Notable related efforts include novel algorithms recently developed to locate earthquakes and to calculate their magnitudes using P- and S-wave energy [8]. Recently, ShakeAlert proposed to detect and disseminate EEW alerts using smartphones, relying on the fact that they have become ubiquitous to the public [9, 10].

The Distributed Multi-Sensor Earthquake Early Warning (DMSEEW) used in this work is a two-step stacking ensemble method for earthquake detection [11]. It relies on the complementary nature of GPS sensors and seismometers to obtain a precise assessment of the magnitude. GPS sensors are better at characterizing high magnitude, while seismometers are better for low magnitude. It has two important data processing steps: a *by-sensor* time-series classification and a *by-region* combination of sensor predictions, as illustrated in Figure 1. This algorithm produces sensor-level class predictions (normal activity, medium earthquake, or large earthquake) based on the data gathered by each sensor. It then aggregates those sensor-level class predictions using a bag-of-words representation to calculate a final prediction for the earthquake category. DMSEEW has been developed, evaluated, and validated using a real-world dataset in collaboration with geoscientists [1]. The ability to fuse data from GPS stations and seismometers differentiate this work from existing methods to detect all the large earthquakes with a precision of 100%, which is critical for any EEW system.

[1] Earthquake Early Warning Dataset. doi: 10.6084/m9.figshare.9758555

3 CONTINUUM COMPUTING: BACKGROUND AND RELATED WORK

3.1 The Continuum Computing Paradigm

The Computing Continuum represents a fluid integration of the computational, storage, and network resources located the edges, in the cloud, and in-between [12]. 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. Figure 2 illustrates the key dimensions of the computing continuum.

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]. Edge resources have limited availability, are distributed geographically, and are often power constrained.

The in-transit tier brings value to applications by allowing quick data triaging and processing while moving the data between geographically distributed nodes. Our previous work integrates approximation and Software Defined Networks (SDN) to support tradeoffs and network management on federated resources. Approximate computing is used to optimize the balance between quality and resource usage. For example, data may be processed with lower accuracy to improve performance when appropriate [15]. SDN approaches enable programmable network management to provision and reconfigure in-transit resources. This flexibility allows for a dynamic architecture for a better fit to application logic and prevents bottlenecks [16].

The Cloud tier handles the heavy data processing using stream processing frameworks and supporting data management systems. Stream processing frameworks are responsible for data transformation, aggregation, and filtering using application logic. Management systems provide key enabling services such as data ingestion systems, key-value stores, or load-balancing and proxy servers. Despite the availability of well-provisioned computing resources in the cloud, the increasing amount of data puts considerable stress on the infrastructure. For example, limited network bandwidth can result in bottlenecks and underutilized cloud resources. Furthermore, cloud providers' current pricing models are based on the number of messages between the edge and the cloud. As a result, transporting all the data to the cloud is expensive (in terms of latency and costs) but may also be wasteful.

Leveraging the computing continuum to support application workflows effectively requires novel solutions for federating infrastructure, programming applications and services, and composing dynamic workflows, which are capable of reacting in real-time to unpredictable data sizes, availability, locations, and rates.

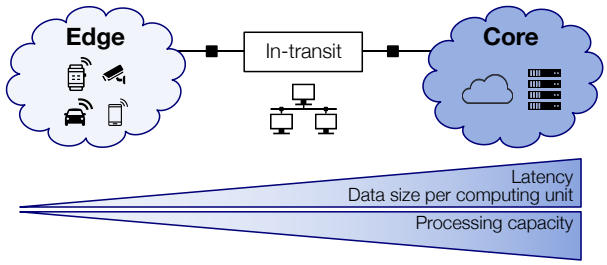

Figure 2: An abstract view of the Computing Continuum. Traversing the data path from the network edges, via in-transit resources, to the core, resources typically increase in scale and processing capacity, but also imply increasing latency and data movement costs. Leveraging the continuum implies effectively using resources along this data path to balance timely processing with processing complexity and quality of results.

3.2 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, 12]. 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 [17, 18]. 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 [19, 20].

Early efforts in distributed training at the edge proposed to train domain-specific models on devices, and only transmit the inferred knowledge for fast inference [21–23]. Collaborative approaches for inference leverage hybrid edge-to-cloud infrastructures to take the offloading decision based on constraints such as the size of input data, the model to be executed, and tradeoffs between the inference accuracy and network latency and bandwidth [24, 25]. 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 [26, 27].

Recent years have seen the availability of a growing number of cloud-based stream processing frameworks. These frameworks rely on data being moved to the cloud and are often agnostic to the specific characteristics and needs of particular data sources/devices [28]. Most of these works are application-specific and lack abstractions for expressing objective for time-critical operations. Our strategy for urgent science across the continuum aims for a general approach with little assumptions on the data and resources capabilities.

4 A FRAMEWORK FOR CONTINUUM COMPUTING

In this section, we describe a 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 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. Finally the framework supports the autonomic management of workflow execution, balancing performance, cost, and other metrics.

4.1 Architecture

A schematic overview of the framework architecture is presented in Figure 3 and is described below.

Infrastructure Layer The infrastructure layer enables the discovery, selection, and monitoring of resources. Resources include data producers and streams (sensors, scientific instruments, IoT devices, etc.) and computing resources and services (cloud services, in-network services, cloudlets, etc.). This layer builds on the Associative Rendezvous (AR) interaction model [29] to enable discovery based on attributes rather than names. Using this model, data producers and services/resource providers use keyword-based profiles to locally describe resources/services as well as availability and access constraints, and consumers use keywords and wildcards to discover and interact with relevant resources. Note that profiles can be locally changed, impacting which resources (data streams, compute) are used.

Federation Layer The federation layer organizes selected resources and services as logical groups based on attributes such as performance and geographical location, and enables these resources to be used to execute the workflow. The logical organization builds on structured peer-to-peer overlays and provides lookup and messaging services within the overlay. The federation layer supports the synthesis of agile execution environments for data-driven workflows using relevant resources across the computing continuum leveraging software-defined environment concepts. These execution environments autonomically evolve to balance performance and costs as workflow requirements as well as the state of the resource change.

Streaming Layer The streaming layer manages workflow execution and the associated data processing. Data-driven workflows may involve merging data from multiple sources, the deployment and configuration of steam processing engines, and the management of the data flow between these engines. The streaming layer provides programming abstractions to express which data streams should be processed and where they should be processed based on data content and using parameters such as available resources, network capacity or application requirements. It also provides runtime mechanisms to support these abstractions and enable workflow execution.

Service Layer The service layer provides the interfaces that expose the services provided by the lower layers that can be used to described and execute workflow. It also provides access to external services such as services for longer-term data storage and management.

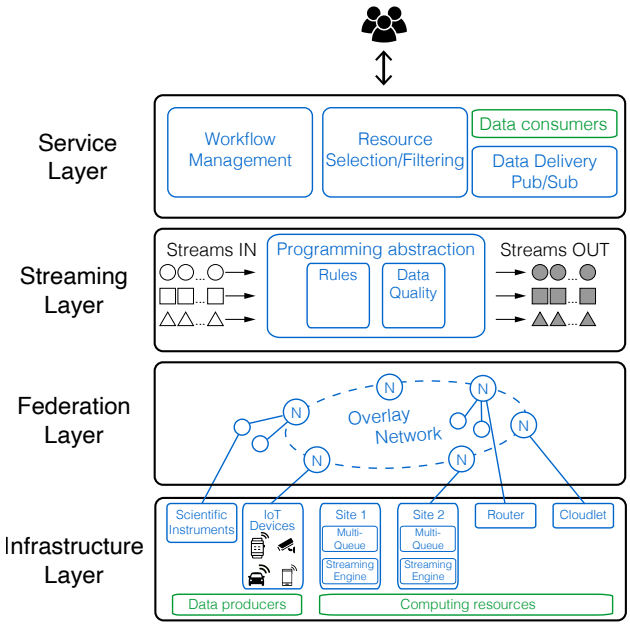

Figure 3: Architecture of the unified framework composed of four layers (extracted from [1]). Each layer exposes operators provided by the building blocks and enabling technologies, to enables data-driven workflows across the computing continuum.

4.2 Building blocks

R-Pulsar is a programming system and edge-based middleware for enabling data-driven workflows that span edge and cloud resources. It supports the definition of data-processing requirements including what data should be process and where, as well as based on data-content, the specification of tradeoffs between the quality and costs. R-Pulsar has been shown to improve metrics such as end-to-end latency, bandwidth consumption, and edge-to-cloud and cloud-to-edge messaging cost for IoT-based workflows [14, 30].

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 [31]. VDC leverages regional data transfer notes (DTNs) and federated data resources to support data services and enable collaborative data-intensive workflows.

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 [32, 33].

5 LEVERAGING THE COMPUTING CONTINUUM FOR EEW URGENT SCIENCE

We use the EEW workflow as a driving use case for urgent science. This model inference workflow can be seen as composed of 3 key steps: data collection at the sensors, time-critical data processing close to the sensors, and time-constrained aggregation and processing of local results at a central computing facility. Early experiments have shown significant variations in the latency measured for processing one seismic event. Performance varies depending on network technology and the mapping of functions to resources along the data path. This is a major challenge for urgent science workflows such as the EEW workflow, where only a couple of seconds can be crucial for people to protect themselves before a strong shaking of the ground [34]. This section describes how the framework presented in Section 4 can be used to increase control over performance and costs.

5.1 Managing data collection at edge sensors

In the EEW workflow, a large number of geographically distributed seismometers and GPS sensors continuously produce data in the form of a 3D time series streamed or queried on demand. For example, in our experimental scenario, we emulate 1,000 sensors deployed on 25 host machines. Each set of 20 sensors composes a geographical region and is connected to one of the 50 Fog Gateway. Each sensor produces 1,200 messages at a frequency of 100Hz, resulting in 12,000,000 messages that have to be processed. This sensor data is then selectively processed close to the sensor location for initial classification, which in our implementation is a Data Transfer Nodes (DTN) located in the network.

The data collection step uses R-Pulsar to program query and store operations on sensor data. Every sensor and computational resource executes an R-Pulsar runtime instance that exposes a semantic profile associated with the data and/or resources. Specifically, sensors declare an advertising profile with the type, frequency, and precision of data they can produce, and requests to be notified when a resource is interested in such data. DTNs declare their interests in terms of the type of sensor data and the region of interest for the first step of classification. The matching of profiles happens dynamically, allowing DTNs to received data streams and trigger data processing either locally or in the cloud.

5.2 Data processing using edge/in-transit resources

The EEW workflow classification step relies on window-based processing to gather measurements streamed by each sensor, and performing classification on this streamed data. Specifically, the DTNs run the WEASEL-MUSE library for Multivariate Time Series classification [35] and Apache Flink for parallel stream processing [36]. Window-based processing stores a finite amount of data at the in-transit resources, to which computations can be applied based on the time that the event occurred on the device producing the data (i.e., *Event time*). As it is only possible to wait for a finite period to gather events within a particular window, *Event time* processing incurs some latency while waiting for out-of-order events [2].

[2]https://ci.apache.org/projects/flink/flink-docs-stable/dev/event_time.html

Our design for fast-decision making relies on an estimation of data processing cost and accuracy to dynamically adjust the mapping of computations. CometCloud aggregates time, estimated accuracy and energy consumption metrics, and interacts with VDC to configure and manage the connectivity between data processing engines.

5.3 Aggregating and processing results at the core

The prediction step filters the data streams by regions and predicts the magnitude of eventual seismic events. This step is typically executed at a central computing facility and relies on Apache Kafka for data ingestion and Apache Flink for data processing. It uses tumbling windows of 1 second to gather predictions from a given region and calculate the final prediction.

Similar to in-transit resources, CometCloud acts with help from VDC services, manages the configuration of stream processing engines (Apache Kafka servers, Flink job managers and Zookeeper) and collects relevant performance metrics. The application's end-to-end performance is tracked for each prediction in a distributed knowledge base, along with associated provenance information. The final predictions are available to the users as well as downstream decision-making systems.

6 SOME RESEARCH CHALLENGES

Extreme heterogeneity and uncertainty. The continuum represents extreme heterogeneity in the capabilities and capacities of systems and services. This heterogeneity 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. For example, the integration of dynamic partial reconfiguration, such as provided by Field Programmable Gate Arrays (FPGAs), requires analysis and understanding of factors that affect usability (overheads, constraints, energy consumption) to benefit and facilitate the creation of ad-hoc clusters in the continuum.

Balancing requirements and expectation 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, satisfying these requirements/constraints 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, including, for example, provisioning new resources and services, adapting scheduling strategies, selecting appropriate systems services, and/or adjusting application parameters as necessary.

Ensuring end-to-end guarantees. 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. Translating (often heterogeneous) local mechanisms for authentication, privacy, provenance, etc., into end-to-end guarantees essential for

enabling applications to take action requires new research and new solutions.

7 CONCLUSION

The proliferation of digital data sources coupled with pervasive access to non-trivial computing capabilities has the potential for enabling new classes of near real-time, data-driven applications and support urgent science.

However, harnessing the computing continuum for urgent science workflows requires understanding the computing resources' capabilities along the continuum, the dynamic application requirements, and desired quality and performance objectives in an integrated manner. In this paper, we presented a conceptual framework for enabling urgent science workflows across the computing continuum. Building on our prior work, we also presented an implementation architecture that includes infrastructure services that automate the discovery of resources, programming abstractions to enable developers to express what data to process, where to process it, and how to manage resources along the data path. We use the early earthquake warning urgent science use case to illustrate the operation of the framework.

As for future work, an interesting direction is to design a general model for balancing computational costs with the value provided to the application in terms of solution accuracy to achieve timely completion of operations.

ACKNOWLEDGMENTS

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 Rutgers Discovery Informatics Institute (RDI^2) at Rutgers, the State University of New Jersey.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[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] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48(4):1–33, 2016.

[16] A. R. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, and M. Parashar. A computational model to support in-network data analysis in federated ecosystems. *Future Generation Computer Systems*, 80:342–354, 2018.

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

[18] 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.

[19] 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.

[20] 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.

[21] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W. Song. Pushing analytics to the edge. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[22] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah. Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 64–76, 2016.

[23] K. Portelli and C. Anagnostopoulos. Leveraging edge computing through collaborative machine learning. In *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 164–169, 2017.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] N. Jiang, C. Schmidt, and M. Parashar. A decentralized content-based aggregation service for pervasive environments. In *2006 ACS/IEEE International Conference on Pervasive Services*, pages 203–212, June 2006.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] F. Tajima and T. Hayashida. Earthquake early warning: what does "seconds before a strong hit" mean? *Prog Earth Planet Sci*, 5(63), 2018.

[35] P. Schäfer and U. Leser. Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*, 2017.

[36] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.