Cloud, Fog or Edge: Where to Compute?

Dragi Kimovski, Roland Mathá, Josef Hammer, Narges Mehran, Hermann Hellwagner and Radu Prodan
Institute of Information Technology (ITEC), University of Klagenfurt

***Abstract*—The computing continuum extends the high-performance cloud data centers with energy-efficient and low-latency devices close to the data sources located at the edge of the network. However, the heterogeneity of the computing continuum raises multiple challenges related to application management. These include where to offload an application – from the cloud to the edge – to meet its computation and communication requirements. To support these decisions, we provide in this article a detailed performance and carbon footprint analysis of a selection of use case applications with complementary resource requirements across the computing continuum over a real-life evaluation testbed.**

1 Introduction

The advent of fog and edge computing has prompted predictions that they will take over the traditional cloud for information processing and knowledge extraction at a large scale. Notwithstanding the fact that fog and edge computing have undoubtedly large potential, these predictions are probably oversimplified and wrongly portray the relations between fog, edge and cloud computing. Concretely, fog and edge computing have been introduced as an extension of the cloud services towards the data sources, thus forming the *computing continuum*.

The computing continuum enables the creation of a new type of services, spanning across distributed infrastructures, for autonomous vehicles, smart cities, and content delivery, among other applications. These services have a large spectrum of requirements, burdensome to meet with "distant" cloud data centers. For instance, they may need low-latency connections for fast decision making close to the data sources and substantial computing resources for complex data analysis. The computing continuum provides a vast heterogeneity of computational and communication resources, which have the potential to meet these demands.

The heterogeneity of the computing continuum raises multiple application management challenges, such as where to offload an application from the cloud to the fog or to the edge. These issues primarily concern the large diversity of the devices, which range from single-board computers such as Raspberry Pis to powerful multi-processor servers. This poses the following dilemma of many practitioners and researchers:

> *Should we use devices accessible with low latency and with limited resource availability, or a high-*

performance cloud at the expense of high communication delay?

To answer this question it is essential to characterize the performance of the resources. Existing literature [1], [2], including the DeFog benchmark suite, addresses this problem by conducting performance analysis of cloud services and to some extent of edge infrastructures. Nevertheless, these approaches (*i*) consider the edge and the cloud resources in isolation, (*ii*) provide only quantitative analysis of the performance without offloading recommendations, (*iii*) evaluate a limited number of devices, and (*iv*) do not consider the environmental impact in terms of CO_2 emissions for executing the applications.

We present in this article a performance characterization and an analysis of the CO_2 emissions of the resources across the computing continuum. Our main goal is to support the decision process for offloading an application to fog or edge resources by considering the application characteristics. For this purpose, we deployed a real testbed named *Carinthian Computing Continuum* (C^3) that aggregates a large set of heterogeneous resources. We base the analysis on three complementary applications widely utilized by industry and research: video encoding, machine learning and in-memory data analytics. We conclude by providing recommendations on where to compute applications across the computing continuum.

2 Carinthian Computing Continuum

Figure 1 depicts the top-level view of the Carinthian Computing Continuum. The C^3 testbed includes a heterogeneous set of resources, distributed across different control domains, including public providers such as Exoscale Cloud[1] and Amazon Web Services (AWS), and research institutions such as University of Klagenfurt[2]. We utilize the ASKALON cloud application computing environment [6] with the MAPO resource provisioning algorithm [5] to deploy the applications across the C^3 testbed. Furthermore,

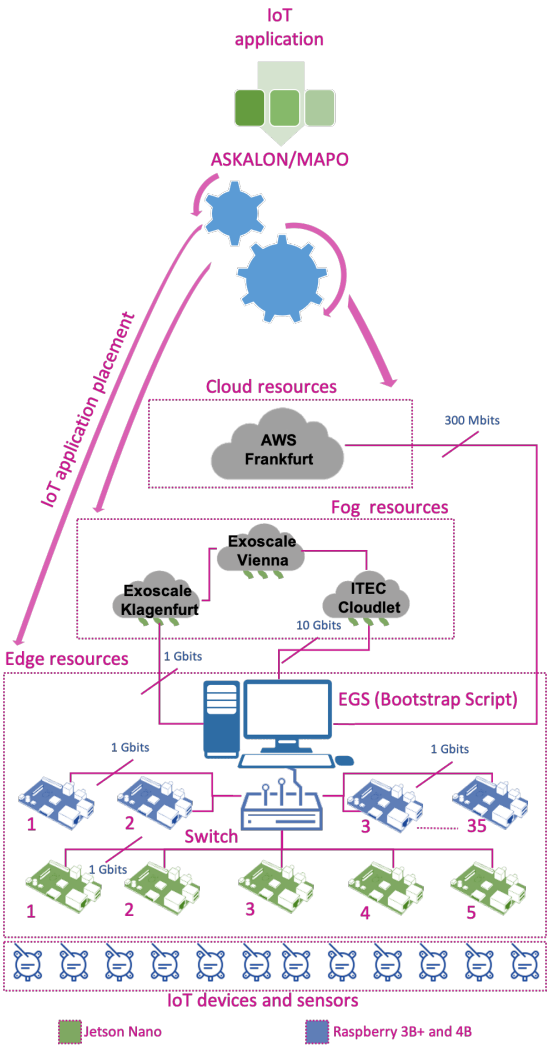


Figure 1: The C^3 testbed.

we employ a bootstrapping script that automatically configures the resources in the testbed [3]. Table 1 summarizes the resource characteristics of the C^3 testbed.

We classify the resources in the C^3 testbed into three layers: *cloud layer*, *fog layer* and *edge layer*.

2.1 Cloud layer

The cloud layer is the uppermost layer of the C^3 testbed. It contains high-performance resources consolidated in vast data-centers, provisioned on-demand as virtual machine instances. As the C^3 testbed resides in Klagenfurt (Austria), we complement it with the

[1] https://www.exoscale.com
[2] https://itec.aau.at

[3] https://github.com/josefhammer/c3-edge

Table 1: Description of the resources available in the C^3 testbed.

Conceptual layer	*Device / Instance type*	*Architecture*	*(v)CPU*	*Memory [GiB]*	*Storage [GiB]*	*Network*	*Physical processor*	*Clock [GHz]*	*Operating system*
Cloud layer	`AWS t2.micro`	64-bit x86	1	1	32	Moderate	Intel Xeon	$\leq$ 3.1	Ubuntu 18.04
	`AWS c5.large`		2	4		$\leq$ 10 Gbps	Intel Xeon Platinum 8000 series	$\leq$ 3.6	
	`AWS m5a.xlarge`		4	16			AMD EPYC 7000 series	$\leq$ 2.5	
Fog layer	`Exoscale Tiny`	64-bit x86	1	1	32	$\leq$ 10 Gbps	Intel Xeon	$\leq$ 3.6	Ubuntu 18.04
	`Exoscale Medium`		2	4					
	`Exoscale Large`		4	8					
	`ITEC Cloud Instance`		4	8			Intel Xeon Platinum 8000	$\leq$ 3.1	
Edge layer	`Edge Gateway System`	64-bit x86	12	32	32	$\leq$ 10 Gbps	AMD Ryzen Threadripper 2920X	$\leq$ 3.5	Ubuntu 18.04
	`Raspberry Pi 3B`	64-bit ARM	4	1	64	$\leq$ 1 Gbps	Cortex - A53	$\leq$ 1.4	Pi OS Buster
	`Raspberry Pi 4`			4			Cortex - A72	$\leq$ 1.5	
	`Jetson Nano`			4			Tegra X1 and Cortex - A57	$\leq$ 1.43	Linux for Tegra R28.2.1

geographically closest European AWS cloud data center located in Frankfurt (Germany).

We carefully selected three instance types based on the `x86-64` architecture that offer to the C^3 testbed a balance of compute, memory, and networking resources for a broad set of applications: general purpose (`t2.micro`), and compute-optimized (`c5.large` and `m5a.xlarge`).

2.2 Fog layer

The fog layer comprises computing infrastructures consolidated in small data-centers in close vicinity to the data sources. This layer comprises resources from two providers in the C^3 testbed [4]: Exoscale and University of Klagenfurt. We allocate these providers in the fog layer as a result of the low round-trip communication latency ($\leq$ 7 ms) and high bandwidth ($\leq$ 10 Gbps). The Exoscale cloud comprises data centers in Vienna and Klagenfurt (Austria). We selected three computing optimized `x86-64` instances from the Exoscale cloud offering: `Tiny`, `Medium` and `Large`. University of Klagenfurt provides a private cloud infrastructure operated by `OpenStack v13.0` and `Ceph v12.2` with one computing optimized instance type described in Table 1.

2.3 Edge layer

This layer encompasses edge resources, such as single-board computers, directly connected to the IoT devices and sensors. An Edge Gateway System (EGS) controls the edge layer, and is the entry point to the other resources available on this level. The EGS supports 10 Gbps Ethernet, dual band PCIe WiFi 5 (802.11ac) and a 150 Mbps LTE 2600 MHz connection. A layer-3 HP Aruba switch with 48 1 Gbps ports connects the EGS to the single-board computers with a latency of 3.8 µs and an aggregate data transfer rate of 104 Gbps. The edge layer also contains 35 physical nodes based on either Raspberry Pi 3B or Pi 4B. Besides, the testbed contains five Jetson Nano devices, each equipped with a general purpose GPU. The edge layer has 1 Gbps Ethernet, Wi-Fi and LTE network connection interfaces.

3 Benchmark applications

We selected three representative application classes with complementary requirements to evaluate the computational performance and the CO_2 emissions of the computing continuum.

3.1 Video encoding

Video encoding allows transmission of video content with different qualities over limited and heterogeneous communication channels. It compresses an original raw video to reduce its effective bandwidth consumption, while maintaining a subjective high quality for viewers. Video encoding has wide fields of applications, including content delivery (live and on-demand video streams), traffic control and surveillance. The video encoding applications have high processing and throughput requirements.

3.2 Machine learning

Machine learning is a branch of artificial intelligence that explores approaches for enabling systems to learn from data, identify patterns and make decisions. Its vast field of application includes automated control in manufacturing, adaptive traffic planning and smart health-care diagnosis, among others. Machine learning, in general, has high processing and operating memory requirements.

3.3 In-memory analytic

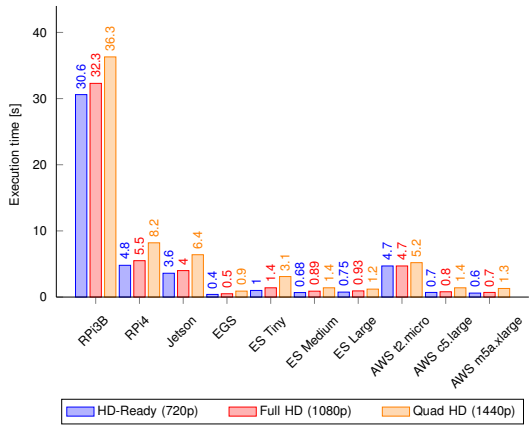
In-memory analytic is essential for efficient low-latency decisions on devices with limited resources. It explores data manipulation such as inspecting, filtering and transforming, and enables efficient extraction of knowledge and non-biased decision-making. Its fields of application include smart cities, healthcare and recommender systems. The in-memory analytic applications require large memory capacity and strict communication latency.

4 Performance evaluation

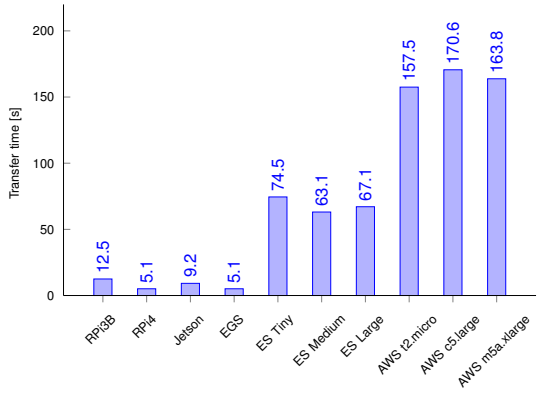
4.1 Video encoding

We evaluate the encoding performance of the computing continuum using FFmpeg version 3.4.6 with the most popular H.264/MPEG-4 video encoder[4] deployed by more than 90% of the video industry[5]. We perform the encoding on a raw video segment with length of 4 s and size of 514 MB, available in the Sintel[6] video-set. The video segment is encoded in three resolutions (HD-ready, Full HD and Quad HD) with data rates of 1500, 3000, and 6500 kbps.

Figure 2 depicts the average encoding time and transfer time, from the video source (located at the University of Klagenfurt) to the encoding device or instance, for a single raw video segment in the three resolutions. The standard deviation ranges from 1.3% for the AWS `m5a.xlarge` instance to 3.6% for the Raspberry Pi 3B devices. We observe that the older generation single-board computers (Raspberry Pi 3B) have a significantly higher encoding time than the other resources. However, the Raspberry Pi 3B devices provide lower transfer times than the cloud instances and are suitable for video-on-demand services employing offline encoding. The Raspberry Pi 4 and the Jetson Nano devices efficiently perform video encoding and provide low transfer times. In some cases, Jetson Nano was capable of encoding up to 20% faster than the AWS `t2.micro` instance with significantly lower transfer times. The remaining cloud and fog resources showed similarly video encoding performance in the range between 0.5 s to 1.3 s. Nevertheless, the cloud and fog resources have limited effective throughput causing higher raw video transfer times. However, the cloud resources are suitable for live video streaming due to the low encoding times. Overall, the EGS achieved the lowest encoding and transfer time due to the low utilization rate and its high computing and networking capabilities.



(a) Average encoding time.



(b) Average raw video segment transfer time.

Figure 2: Average encoding performance of a 4 s long video segment with the x264 codec and FFmpeg 3.4.6.

Recommendation. We recommend executing video-on-demand encoding at the edge using the latest generation of single-board computers or dedicated systems, as they significantly reduce the raw video trans-

[4] https://trac.ffmpeg.org/wiki/Encode/H.264
[5] https://www.itu.int/rec/T-REC-H.264-201906-I/en
[6] https://media.xiph.org/sintel

fer time. Cloud and fog devices (i.e., close-by servers, small data centers) are more suitable for continuous live stream encoding if the effective incoming and outgoing throughput is sufficient and the delay incurred by the transport is tolerable.

4.2 Machine learning

We use TensorFlow Core version 2.3.0 to evaluate machine learning performance. We created two training and validation scenarios for feature identification in a set of images:

- A *quantum neural network* using the MNIST data-set[7] limited to 20000 samples with a size of 3.3 MB. The scenario creates a neural network with two layers and 128 outputs from the previous layer to the next. We conduct five iterations to reach a feature identification accuracy of 90%.
- A *convolutional neural network* using the Kaggle data-set[8] with a size of 218 MB. The minimum required accuracy is 80%. The convolutional network has three layers with a kernel size of three. Each layer uses increasingly higher filter sizes in the range [32, 64, 128]. After each layer, we use a max-pooling sample-based discretization process to reduce the spatial dimensions. We repeat the training five times.

Figure 3 analyzes the average execution time for training the two neural network types and the transfer times of the training data from centralized storage to the device or instance that performs the training. The standard deviation ranges from 1.2% for the Raspberry Pi 4 devices to 5.4% for the AWS `t2.micro` instance. The evaluation shows that the less complex quantum neural network requires a relatively lower training time across all resources. The old generation single-board computers show again a lower performance, and their suitability for training heavily depends on the size of the training data and the model. The other fog and edge devices provide similar performance to the cloud resources. The single-board computers provide lower training performance for the convolutional network. The only exception are the Jetson Nano devices able to train the convolutional network up to four times faster than the Raspberry Pi devices. In general, the EGS provides the lowest training time among all devices. The training data transfer time has limited influence on the training process, especially for the quantum neural network. While the training data transfer time is significantly higher for the convolutional neural network, the cloud and fog resources outperform the edge devices, except EGS.

Recommendation. We recommend the model training with large data-sets and multiple layers in the cloud or on dedicated systems (such as EGS), whenever possible. We recommend offloading to the edge only when the training data is of limited size, or when the neural network has few layers.

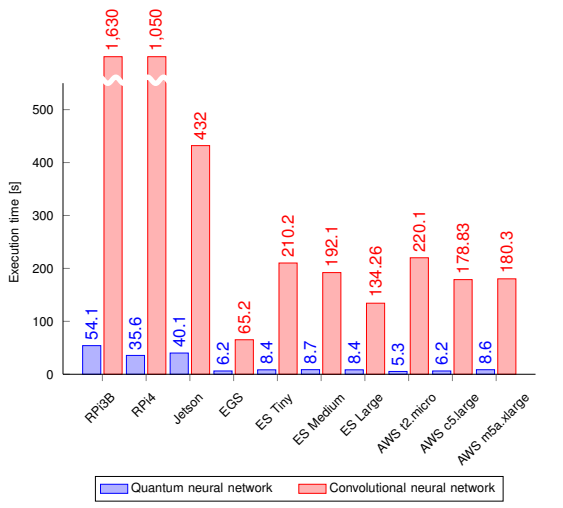
4.3 In-memory data analytics

The in-memory data analytics evaluation explores two scenarios using Apache Spark version 2.4.6:

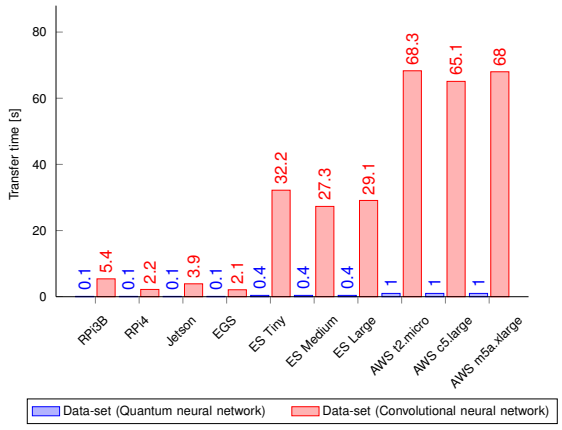
- *Collaborative data filtering* aims to fill missing entries for improved recommendation of movies to consumers. The model uses the alternating least squares algorithm and a data-set[9] of movie preferences with a size of 31.6 kB. We trained the model over the available data-set with a cold start strategy that randomly divides the data into training and validation sets.
- π *estimation* is a memory and computationally intensive task that estimates the value of π by distributing the work among multiple Spark executors. This enables us to evaluate the computational and memory performance of the distributed memory computing continuum for complex tasks.

Figure 4 shows the average execution time of the in-memory collaborative data filtering and the π estimation. The standard deviation ranges from 1.3% for the AWS

[7] http://yann.lecun.com/exdb/mnist/
[8] https://www.kaggle.com/tags/animals
[9] https://github.com/apache/spark/blob/master/data/mllib/als/sample_movielens_ratings.txt



(a) Average training time.



(b) Average training data transfer time.

Figure 3: Average training and data transfer times of two neural network types.

`m5a.xlarges` instance to 4.6% for the Exoscale `Tiny` instance. The AWS and Exoscale cloud instances perform better than the EGS and the single-board computers for the π calculation thanks to their larger memory size and the more efficient memory controllers. The collaborative filtering shows the same trend and the Exoscale instances in Vienna show the best performance. The data transfer time of the collaborative filtering is negligible due to its small size.

Recommendation. We recommend fog instances for collaborative data filtering, due to the relatively small difference in the execution time compared to the cloud. The edge devices can be a reasonable option for applications with soft constraints on the data filtering time. Finally, we recommend executing compute intensive in-memory processing (e.g., π estimation) in the cloud or offloading to fog devices with good memory management.

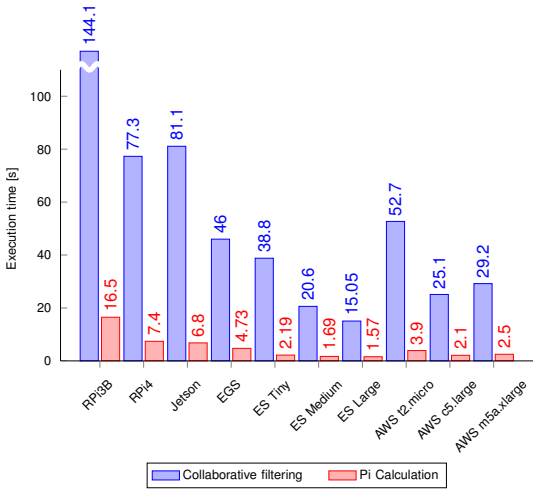


Figure 4: Average execution time for in-memory collaborative data filtering and π estimation using Apache Spark.

4.4 Network performance

Furthermore, we evaluate the network performance of each instance and device in the C^3 testbed by measuring the effective downlink throughput with the iPerf3[10] tool over TCP and the round-trip latency by sending ICMP echo requests from a device registered in the University of Klagenfurt network.

Figure 5 shows the average results with a standard deviation between 0.5% for EGS to 15% for the Exoscale `Tiny` instance. The single-board computers and edge devices provide a 10 times higher throughput and 20 times lower latency.

Recommendation. The edge and fog resources are most suitable for applications that generate frequent input and output requests with larger data sizes.

4.5 Carbon emission

We evaluate the power consumption of the physical devices used for the convolutional

[10]https://iperf.fr/

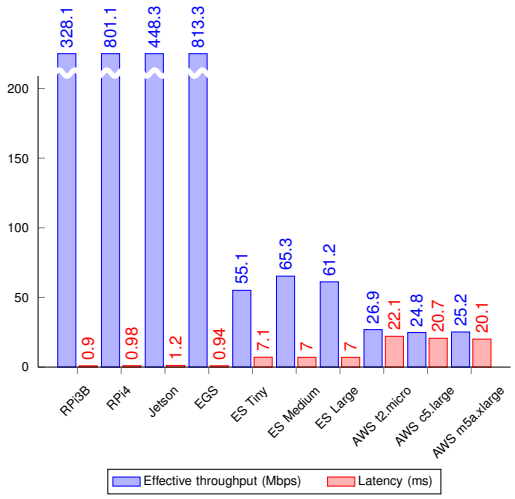


Figure 5: Round-trip communication latency and effective throughput measured with iPerf3 and ICMP echo request.

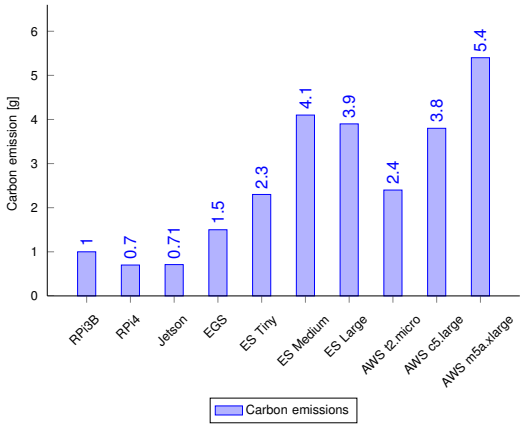


Figure 6: Carbon footprint for training a neural network with accuracy of above 80%.

neural network training in TensorFlow. We use a digital multimeter to physically measure the average electrical current during training on the edge and fog resources. We rely on an AWS research report to approximate the power consumption of the fog devices and cloud instances provided by AWS and Exoscale for different utilization rates [7]. We estimate the carbon emission directly correlated with the power consumption [3], based on the grams of CO_2 emissions for producing one kW h of energy in the European Union.

Figure 6 shows that the edge devices emit up to six times less carbon during training. We therefore expect to reduce the carbon emissions by 1000 kg per year by offloading the computation from the cloud to the edge, which is equivalent to a 5517 km-long travel with a gasoline vehicle.

Recommendation. We recommend offloading applications with soft execution time constraints (e.g., video-on-demand, data filtering, model training with small data-sets) to the edge devices. This reduces the energy costs of service providers and the carbon footprint.

5 Conclusion

In this article we provide a set of recommendations for practitioners on where to offload their applications across the computing continuum, summarized in Table 2. We formulate the recommendations based on a systematic performance and carbon footprint analysis of a selected set of applications on a heterogeneous set of devices and cloud instances across the computing continuum. For this purpose, we deployed a representative testbed called Carinthian Computing Continuum that spawns across a three-layered conceptual architecture. Our results revealed that to reduce the network traffic over the computing continuum it is recommended to offload to edge and fog resources, while we advocate the cloud for lower execution times. Lastly, for decreasing the CO_2 emissions, with an acceptable computational performance penalty, we recommend edge resources.

REFERENCES

1. McChesney, J., Wang, N., Tanwer, A., de Lara, E., Varghese, B. (2019, November). DeFog: fog computing benchmarks. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (pp. 47-58).
2. Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., Hu, K. (2019, April). An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (pp. 3-18).
3. Moro, A., & Lonza, L. (2018). Electricity carbon intensity in European Member States: Impacts on GHG emissions

Table 2: Recommendations for application offloading across the computing continuum.

Requirement / *Application*	*Low network load*	*Low execution time*	*Low CO_2 emissions*
Video encoding	Edge/Fog	Cloud	Edge
Machine learning	Edge	Cloud/Fog	Edge
In-memory analytic	Cloud/Fog	Cloud	Edge

of electric vehicles. Transportation Research Part D: Transport and Environment, 64, (pp. 5-14).

4. Kimovski, D., Ijaz, H., Saurabh, N., Prodan, R. (2018, May). Adaptive nature-inspired fog architecture. In 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC) (pp. 1-8).
5. Mehran, N., Kimovski, D., Prodan, R. (2019, October). MAPO: A Multi-Objective Model for IoT Application Placement in a Fog Environment. In Proceedings of the 9th International Conference on the Internet of Things (pp. 1-8).
6. Fard, H. M., Prodan, R., Barrionuevo, J. J. D., Fahringer, T. (2012, May). A multi-objective approach for workflow scheduling in heterogeneous environments. In 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012) (pp. 300-309).
7. Cloud computing server utilization. https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment, 2020. [Online; accessed 11-November-2020].

Dragi Kimovski is a tenure-track researcher at the Institute of Information Technology (ITEC), Klagenfurt University. He earned his PhD in 2013 from Technical University of Sofia (Bulgaria). He was assistant professor at the University for Information Science and Technology in Ohrid (North Macedonia) and senior researcher at the University of Innsbruck (Austria). His research interests include fog and edge computing, multi-objective optimization, and distributed storage.

Roland Mathá is a PhD student with an MSc degree in computer science from the University of Innsbruck (Austria) in 2014. His interests include cloud simulation, workflows, and multi-objective optimization.

Josef Hammer studied computer science in Austria and Australia and received his MSc degree from the Klagenfurt University. After having spent ten years at CERN and in the automotive industry, he currently pursues a PhD degree in computer science at ITEC, Klagenfurt University. His research focus is on edge computing in connection with 5G mobile networks.

Narges Mehran is a PhD student at ITEC, Klagenfurt University. She received her MSc degree in computer architecture from the University of Isfahan (Iran) in 2016. Her research interests include cloud, fog and edge computing for future Internet applications.

Hermann Hellwagner is professor at and the chair of ITEC, Klagenfurt University. He received his PhD degree from the University of Linz (Austria) in 1988. His research interests include distributed multimedia systems, information-centric networking, edge computing, and communication in UAV swarms.

Radu Prodan is professor in distributed systems at ITEC, Klagenfurt University. He received his PhD degree in 2004 from the Vienna University of Technology and was Associate Professor until 2018 at the University of Innsbruck (Austria). His research interests include performance and resource management tools for parallel and distributed systems.