POLITECNICO DI TORINO
Repository ISTITUZIONALE

Balancing Energy Efficiency and Infrastructure Knowledge in Cloud-to-Edge Task Distribution Systems

Original
Balancing Energy Efficiency and Infrastructure Knowledge in Cloud-to-Edge Task Distribution Systems / Galantino, S., Pinto, A., Esposito, F., Manzalini, A., Risso, F.. - (2024), pp. 28-34. (EuroSys '24: Nineteenth European Conference on Computer Systems Athens (GRC) 22 April 2024) [10.1145/3642975.3678965].

Availability:
This version is available at: 11583/2992348 since: 2024-09-10T12:34:03Z

Publisher:
ACM

Published
DOI:10.1145/3642975.3678965

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026



Balancing Energy Efficiency and Infrastructure Knowledge in Cloud-to-Edge Task Distribution Systems

Stefano Galantino
Dept of Control and Computer Engineering, Politecnico di Torino
Torino, Italy
stefano.galantino@polito.it

Andrea Pinto
Computer Science Department, Saint Louis University
Saint Louis, MO, US
andrea.pinto.1@slu.edu

Flavio Esposito
Computer Science Department, Saint Louis University
Saint Louis, MO, US
flavio.esposito@slu.edu

Antonio Manzalini
Innovation Labs, Telecom Italia Mobile (TIM)
Torino, Italy
antonio.manzalini@telecomitalia.it

Fulvio Risso
Dept of Control and Computer Engineering, Politecnico di Torino
Torino, Italy
fulvio.risso@polito.it

ABSTRACT

In the rapidly evolving landscape of distributed computing, maintaining energy efficiency in edge and data center infrastructures has become critical. While the problem has been faced with centralized approaches assuming knowledge of the available underlying infrastructure resources, this paper introduces a distributed task allocation framework emphasizing energy awareness without requiring infrastructure knowledge. The framework is designed to optimize energy consumption in heterogeneous computing environments, leveraging a distributed consensus algorithm that allows nodes to maximize individual or global goals. Each private custom utility function enables a node to carefully determine whether executing a task is efficient, thus ensuring flexibility in the task allocation process based on local preferences. While showcasing the energy efficiency of our framework, we also illustrate that it is not necessary to disclose the underlying infrastructure resources status, ensuring the preservation of potentially sensitive local resources information. Experimental results demonstrate the framework's ability to achieve optimal power consumption outcomes while maintaining privacy, offering a significant advancement over traditional centralized allocation policies and Kubernetes-like scheduling algorithms.

CCS CONCEPTS

• **Hardware** → **Impact on the environment**; • **Information systems** → *Computing platforms*; • **Computer systems organization** → **Cloud computing**; *Heterogeneous (hybrid) systems.*

ACM Reference Format:
Stefano Galantino, Andrea Pinto, Flavio Esposito, Antonio Manzalini, and Fulvio Risso. 2024. Balancing Energy Efficiency and Infrastructure Knowledge in Cloud-to-Edge Task Distribution Systems. In *1st International Workshop on MetaOS for the Cloud-Edge-IoT Continuum (MECC '24), April 22, 2024, Athens, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3642975.3678965



This work is licensed under a Creative Commons Attribution International 4.0 License.
MECC '24, April 22, 2024, Athens, Greece
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0543-4/24/04
https://doi.org/10.1145/3642975.3678965

1 INTRODUCTION

In recent years, containerization has become increasingly popular as a lightweight method for packaging applications in a format that can be used interchangeably, regardless of the underlying infrastructure [1]. This common foundation has spurred the cloud-native revolution, shifting focus from individual servers to entire data centers. Additionally, with the advent of edge and fog computing [2–4], which prioritize geographical proximity, reduced latency, and enhanced privacy, these methods are being adapted for smaller data centers at the network's edge, using consistent principles to promote service flexibility.

Such technological advancement paved the way for the computing continuum (also referred to as edge-to-cloud continuum) [5], in which devices coming from different layers of the computing substrate (i.e., edge, fog, cloud) share computing resources with the other members of the continuum federation in the attempt to host user workloads. Still, despite the development of standard interfaces for application orchestration [6, 7], standard industry practices treat each infrastructure as a series of connected but isolated silos rather than as a single virtual space.

Within the context of the European project FLUIDOS [1], we argued that the effective utilization of resources within a continuum is contingent upon their recognition and exposure as a unified pool. Indeed, the possibility of having heterogeneous and geographically distributed computing resources allows for enhanced allocation policies, including the device location as an additional dimension to the problem. This new spatial awareness fosters enhanced allocation policies that are related to the power consumption of the entire computing infrastructure [8–11]: (*i*) the workload characteristics can be exploited to identify, among the heterogeneous set of devices in the infrastructure, the most suitable (i.e., the most energy-efficient) to host the workload and (*ii*) workloads can be shifted in time and space based on the availability of renewable sources to promote sustainable computing.

[1] https://www.fluidos.eu/

Such goals can be achieved both by centralized and distributed allocation policies. From a functionality perspective, the former can guarantee the best allocation scheme for the submitted workloads but, at the same time, requires a constantly updated knowledge of the infrastructure. The latter, instead, reduces or, in some cases, eliminates the non-trivial need to share the infrastructure nodes utilization status leading to an (sub)optimal allocation. Therefore, we identify the protection of sensitive node utilization metrics as one of the main concerns when choosing the most suitable approach.

To this end, this paper proposes a distributed allocation framework, which relies on a *distributed consensus algorithm* to allocate tasks using private utility functions. The algorithm operates in two phases: (*i*) a task *dispatching* during which each node decides how green it is to execute that task on its available resources using the local private utility function and (*ii*) a *consensus* during which the nodes communicate to each other the output value of the function, deciding for an optimal task allocation. The framework allows each node within the infrastructure to employ a utility function, indicating its willingness to accept (and allocate) a particular task. Later in this paper, we propose a utility function for energy-awareness, still, each node can implement a custom one, reducing the need to disclose information and enhancing its privacy with the infrastructure. Eventually, for each task, the framework will converge on the unique optimal task allocation.

The rest of the paper is organized as follows. Section 2 details the state of the art for distributed energy-aware task allocation. Section 3 provides a high-level overview of the system architecture. Sections 4 and 5 describe the main contribution of this paper, i.e., the model description. Finally, Section 6 validates our proposal, and Section 7 concludes the paper.

2 RELATED WORK

Recent advancements in distributed task allocation have significantly improved mobile edge computing, multi-robot systems, and distributed computing environments. In the realm of multi-robot systems [12] presented a distributed task allocation and scheduling algorithm for missions with tight coupling between tasks, emphasizing the importance of addressing temporal and precedence constraints through a distributed metaheuristic algorithm. Generalizing the scenario, [13] proposed a load-balancing technique that randomly probes two nodes in edge-clouds and selects the one with the less load to place the tasks to provide a QoE guarantee. Finally, focusing on the distributed algorithm, [14] presented an allocation policy based on the CBBA consensus algorithm that, differently from other distributed algorithms like RAFT [15] and Paxos [16, 17], have a convergence guarantee. Still, while being able to provide privacy guarantees, none of the above proposals included energy consumption in the problem formulation.

Including energy into the allocation problem, [18–20] tackled the problem of task offloading in mobile computing. However, they primarily focused on offloading techniques to reduce power consumption for battery-constrained end-user devices, neglecting additional energy requirements on servers. [11] further extended the problem, including an online joint offloading and resource allocation framework to prevent edge devices from exceeding a specified energy budget. Cloud-to-edge infrastructures are typically shared among multiple users thus requiring both to correctly model user mobility[21], and balance between energy-awareness while providing delay-guarantees in task execution [22]. Authors in [23] extended the problem formulation with a time-division multiple-access system for minimizing the weighted sum of mobile energy consumption under the constraint of computation latency. Finally, the energy-aware workload re-distribution has also been addressed, including heterogeneous edge devices in the problem formulation [24], and evaluating the applicability on vehicular networks [25].

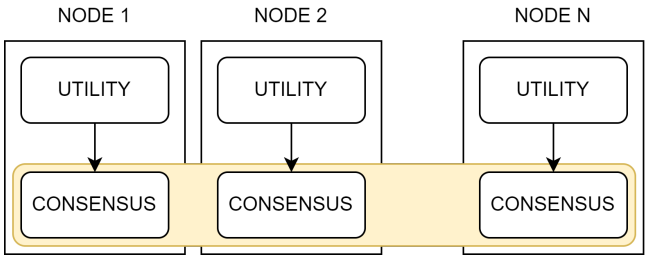


Figure 1: Each node maintains a utility function to determine its suitability for executing a specific task, and the underlying framework analyzes the outputs from these utility functions to converge on an optimal task allocation.

In the present study, we have examined several works related to task allocation in edge-to-cloud infrastructures. However, all of the works presented thus far require varying degrees of knowledge on infrastructure status, compromising the privacy of certain nodes unwilling to share sensitive information. We, therefore, assert that the joint allocation of tasks in distributed edge-to-cloud infrastructures, which considers both energy and privacy requirements, remains largely unexplored.

3 SYSTEM ARCHITECTURE

The edge-to-cloud approach follows a decentralized and peer-based model similar to the Internet. This approach allows diverse participants, including large cloud providers, smaller enterprises linked to specific territories, and even small offices or homeowners, to independently and dynamically determine with whom they want to share resources by choosing who to peer with. After joining this edge-to-cloud infrastructure, each participant in this continuum may pursue individual goals (e.g., maximize resource usage, or profits) or share some common objective (e.g., minimize the overall infrastructure power consumption). It is worth mentioning that if the nodes in the continuum pursue a common goal, some level of information disclosure is required.

On top of this continuum infrastructure, applications must be efficiently allocated (i.e., satisfying all the execution requirements), and nodes should compete to identify the most suitable location to host the execution. In this regard, each node is equipped with its utility function that maps how much a given node is willing to host one application or a part thereof in the case of microservice-based applications. Customizable utility functions can include metrics

such as resource usage and power consumption; however, the output value must fall within a predefined range to ensure fairness in allocation. Nonetheless, multiple nodes can share the same utility function to pursue global objectives or opt for a custom one in the case of local objectives.

We then use a distributed consensus-based RAFT-like algorithm to reach a consensus among the participants in the continuum to select the node with the highest utility value. The RAFT protocol allows the creation of a compute node mesh (see Fig. 1) to exchange the output of the utility functions consistently. This can be achieved through an automatic leader election process that selects the node that is fully responsible for managing log replication on the other servers of the cluster. [2] Our distributed protocol operates in two phases: *task dispatching* and *consensus*. First, each node receives the allocation request of an application. To secure the hosting of the job or part of it (e.g., a subset of constituent microservices), the node shares the output value of its utility function. Specifically, each node checks the value of the current highest utility with the other cluster members, overriding it if higher. At the end of the consensus phase, the information of the node with the highest utility is replicated among all the nodes in the continuum to perform the actual allocation.

4 PROBLEM FORMULATION

We now describe the resource allocation protocol designed to guarantee w.r.t. the optimal allocation. Such a mechanism is designed to clear the resource allocation problem when competing jobs must be concurrently run on the hosting physical infrastructure, participating in what we call a federation $\mathcal{F}$ while minimizing the overall power consumption of the computing infrastructure.

We assume a collection of $\mathcal{N}$ physical nodes, potentially hosting one or more jobs, and we index such nodes with $n \in \mathcal{N}$, where $\mathcal{N} = \{1, \dots, N\}$, with N being the total number of nodes participating in the federation $\mathcal{F}$. Each node is then equipped with computing resources (i.e., CPU), denoted as c_n. It is worth mentioning that most Cloud-based solutions typically describe computing resources in terms of CPU and memory resources; however, since the highest correlation occurs mostly between CPU usage and power consumption, memory requirements will be omitted in the problem formulation without losing generality.

Furthermore, physical nodes are also described using the transfer function $P(x)$, which correlates the CPU usage x with the energy required to sustain such load (the exact formulation of $P(x)$ will be detailed Section 5).

We assume that each hosting node n has a (private) utility function U_n, and we are seeking an allocation solution that maximizes the sum of the utilities of all nodes. Such utility function is a policy of our resource allocation mechanisms and can be tuned to be engineered for various application and infrastructure goals. In the scope of this work, we design the utility function for a generic node n to be as follows:

$$U_n(x) = \frac{1}{P_n(x) - P_n(x_0)} \tag{1}$$

[2] In the RAFT terminology, a cluster comprises the set of nodes involved in the distributed consensus.

where $U_n(x)$ is a function of the CPU usage x. Specifically, upon receiving the request to allocate a task that would increase the resource usage of the node to a value of x, the value of utility is the reciprocal of the difference between the power consumption of the node n at the expected usage x, and the power consumption computed for the prior resource usage x_0 (i.e., the step increase in power consumption). Intuitively, the lower the increase in power consumption to host the requested job, the higher the utility value.

Our primary objective is to determine an allocation of a set of jobs, referred to as $j \in \mathcal{J}$, where $\mathcal{J} = \{1, \dots, J\}$ onto the nodes in $\mathcal{N}$ of the federation ensuring a conflict-free assignment, i.e., satisfying the computing requirements of the job without exceeding the available computing capability c_n of each node n. Indeed, every job j has particular resource needs that must be met during allocation to ensure optimal execution across the nodes' resources. To this end, following the microservice-based approach, we represent each job j as a set of loosely coupled components $\mathcal{M}_j$, and we index such components for each job j with $m_j \in \mathcal{M}_j$, where $\mathcal{M}_j = \{1, \dots, M_j\}$. We assume that each component m_j of the job j has a specific computing resource demand, and we denote it with r_{m_j}. Given such notation, a *conflict-free assignment* is an assignment in which each component m_j of the job request j is mapped to one and only one of the hosting nodes n (note that multiple valid mappings of a job over the topology are possible).

Based on the above notation, we model the (NP-hard) *constrained graph matching resource allocation* problem upon the arrival of the job j with the following:

$$\max_{x} \quad \sum_{n \in \mathcal{N}} \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}_j} U_n(y_j) \tag{2}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}_j} y_j r_{m_j} \leq c_n \qquad \forall n \in \mathcal{N} \tag{3}$$

$$\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_j} y_j = M_j \qquad \forall j \in \mathcal{J} \tag{4}$$

$$\sum_{m \in \mathcal{M}_j} y_j \leq 1 \qquad \forall n \in \mathcal{N}, \forall j \in \mathcal{J} \tag{5}$$

where the allocation variable $y_j \in \{0, 1\}^{N \times M_j}$ is intended to assign job j (i.e., the set of its components) on the node n. Eq. (2) seeks to maximize the node utilities as a function of the allocation variable y_j under the constraint (Eq. (3)), which enforces that the sum of resources assigned to the node n does not exceed the physical capacity c_n for the same node. Finally, the conflict-free assignment is enforced with (Eq. (4)) and (Eq. (5)), respectively stating that all the components of the job j are allocated (i.e., we don't consider a partial allocation as valid) and the same component is not assigned to multiple nodes.

5 SYSTEM MODEL

Beyond performance, CPU architectures vary significantly in terms of power consumption, with some chips being optimized for energy efficiency (for instance, Raspberry Pis) and others prioritizing processing power (like those used in servers). Therefore, it is essential for this study to first establish a baseline for comparing both performance and power consumption across various CPUs.

Table 1: Infrastructure setup.

Device type	CPU model	Count	# CPUs	α	β	δ
SERVER	Intel Xeon Gold 5120	5	28	1-7	150-180	0.5-α
DESKTOP	Intel Core i7-6700	5	8	10-20	10-20	0.2-6
RASPBERRY	ARM Cortex-A72	5	4	0.5-1	2.5-4.5	NA

We started by conducting real measurements to evaluate the power consumption of the devices and then extrapolated the results to obtain an accurate mathematical model. To assess power consumption, the study uses a smart plug connected to a wall outlet and monitors the total power usage of the device as the number of CPU cores allocated gradually increases (following the work in [26]). Using such experimental results, we can define a mathematical model to represent devices' power consumption as a function of the CPU usage x as follows:

$$P(x) = \begin{cases} \alpha x + \beta, & \text{if } 0 < x \leq \rho \\ \delta x + (\alpha\rho + \beta - \delta\rho), & \text{if } \rho < x \leq 2\rho \end{cases} \quad (6)$$

where ρ represents the number of physical cores of the system. In detail, devices experience different power consumption patterns, depending on whether or not the current CPU usage x exceeds the number of physical cores (i.e., logical cores are also required to sustain the load). Specifically, with a CPU usage = 0 all devices experience an idle power consumption β, which is typically more prominent in the case of servers. Until all the physical cores are reserved, the power consumption typically follows a linear increase with a slope value of α. Then, the power consumption keeps increasing linearly but with a lower gradient value ($\delta < \alpha$). The values of α, β, δ are device-specific, but the values for devices belonging to the same class (e.g., server, workstation, Raspberry PI) fall into predefined class ranges.

Next, in the consensus phase, the optimal job energy-aware allocation described in Eq. (2) can be achieved by weighting the components m_j of each job j based on the individual computing requirements r_{m_j} and prioritizing the ones with the lowest demands for the bidding process. As a result, this approach drastically reduces resource fragmentation, allowing the most energy-efficient devices to maximize their efficiency.

6 EXPERIMENTAL EVALUATION

Using a simulation-driven approach, this section details the experimental evaluation of the proposed solution, further validated within the software framework provided by the FLUIDOS project.

6.1 Setup

The simulated infrastructure is composed of 15 devices in total, 5 Desktops, 5 Servers, and 5 Raspberry PIs. TABLE 1 summarizes the main characteristics of the devices and the ranges for the α, β, and δ values used to represent devices' power consumption based on the model described in Eq. (6). Fig. 2 depicts the resulting correlation between the CPU usage and the expected power consumption of the different classes of devices in the infrastructure.

As discussed in the introduction, we aim to achieve the best allocation for infrastructure power consumption while preserving the

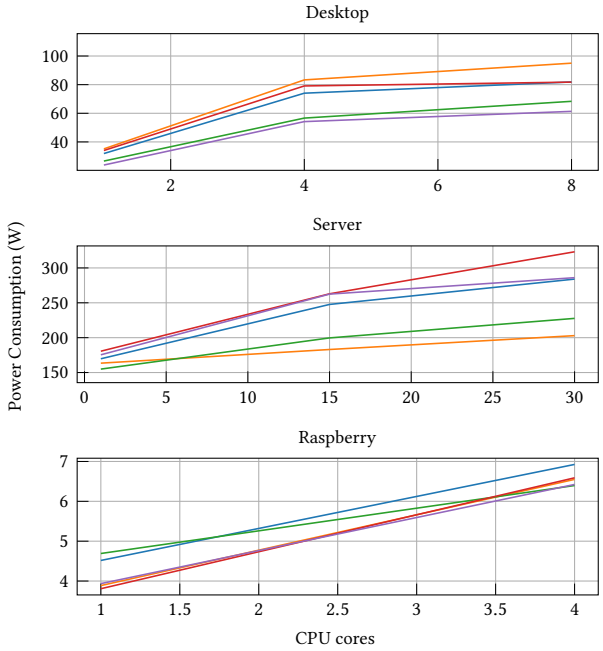


Figure 2: Correlation between CPU usage and power consumption of the different classes of devices included in the simulated infrastructure.

privacy of devices unwilling to share any internal metrics. To this end, throughout the simulation, we tested different configurations, namely DA-n (i.e., distributed-allocation-n), representing scenarios in which all the nodes participate in the process using the utility function described in Eq. (1) (i.e., DA-0), and scenarios in which some of the nodes are not willing to share sensitive information and, instead, rely on custom utility functions (i.e., DA-[2,4,6,8] in which the number represents the number of devices not participating in the energy-aware process, thus introducing noise in the process). Specifically, such nodes implement a random utility function to represent the maximum possible unpredictability.

Results are then compared with the well-known Kubernetes scheduling algorithm (shortened in k8s in the following figures), which is not energy-aware and accounts only for available computing resources in the nodes, and a brute force allocation policy (shortened in BF), which has the full knowledge of the infrastructure and can thus perform the best allocation concerning power consumption.

The submitted workloads consist of 500 tasks, arriving at a rate of one task per second, with CPU needs varying within a specified

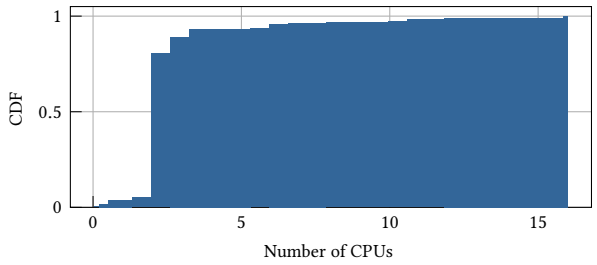


Figure 3: Distribution of the CPU demand of the submitted workloads.

range as shown in Fig. 3. As we can see, the majority of tasks require approximately 2 CPU cores, but a conspicuous set of jobs has a much higher resource demand (up to 15 CPU cores). [3] Instead of each task's CPU demand being uniformly distributed across its components, it is randomly divided in a non-uniform manner across 1 to 6 components (i.e., microservices). This means that for a task with six components, the CPU requirement for each component isn't simply one-sixth of the total demand, but rather, the demands of all components together equal the overall task requirement. Each task is defined by its execution time, and resources are freed up immediately upon completion. To evaluate how the system handles different demand levels, we conducted simulations using varied execution times, ranging from 1 second to 140 seconds.

6.2 Energy-aware allocation

Fig. 4 depicts the power consumption of the different configurations previously detailed, along with the brute force and Kubernetes-like allocation policies as a reference. Specifically, Figs. 4a to 4c represent the result of the allocation varying the duration of the submitted jobs, respectively, with 60, 100, and 140 seconds. Intuitively, increasing the task duration while keeping the same arrival rate results in different pressures for the computing infrastructure.

The results show that if all nodes use the energy-aware utility function presented in this paper (DA-0), it is possible to obtain almost the same results as the optimal brute force approach for the overall power consumption of the infrastructure. The difference with respect to the optimal allocation is always $<<$ 5%, demonstrating that we can always achieve a near-optimal allocation. Instead, if we increase the number of devices not actively participating in the energy-aware allocation, we notice that the gap with the optimal allocation keeps increasing. Still, even with eight devices not participating in the energy-aware allocation (i.e., more than 50% of the devices of the infrastructure), we can always obtain better results than the Kubernetes-like allocation policy. Overall, we can say that, on average, an energy un-aware allocation policy like k8s always requires between 5% and 20% more energy than our energy-aware allocation policy.

It is worth noting that the possible benefits derived from energy-aware allocation strictly depend on the amount of workload submitted to the infrastructure. With highly saturated infrastructures, as the scenario depicted in Fig. 4c, the possible solutions for the allocation problem are extremely limited, resulting in almost comparable results for all the configurations. In fact, if the infrastructure is overloaded with job requests, the allocation problem becomes a simplified version of the knapsack problem (i.e., fit as many jobs as possible with the few available resources). Still, both the brute force approach and our allocation policy could allocate all the jobs in less time than k8s, resulting in the outliers in Fig. 4c (in those time instants, the consumption is = $0W$).

Such results can motivated by the fact that the energy-aware allocation policy can preemptively select only the most efficient nodes for the job execution while leaving the less efficient ones only in case of saturation of resources. In fact, Fig. 5 details the CPU utilization of the different classes of devices in the infrastructure for the case of DA-0 configuration. As we can see, with low saturated infrastructures as in Fig. 5a, our proposal relies almost entirely on servers and Raspberry PIs to host the submitted tasks (being the most energy-efficient devices of the lot). Instead, if we increase the load submitted to the infrastructure as in Figs. 5b and 5c, the system has to select less energy-efficient nodes to host the workload.

As a final comment on the results, although our approach proved some interesting results on this specific setup, the same considerations and benefits can be achieved on any infrastructure with heterogeneous devices, in which it is important to consider not only the processing capability of the different devices but also the energy required to obtain such computation.

7 CONCLUSION

This study presents a distributed framework to optimize energy consumption in heterogeneous computing environments for cloud-to-edge computing environments. Leveraging custom utility functions, the framework demonstrates a promising approach to optimizing task allocation while minimizing energy consumption across the network, bypassing the need to share private resource information. Compared to traditional centralized allocation policies and Kubernetes-like scheduling algorithms, our method offers significant improvements in energy efficiency without compromising the privacy of the compute nodes participating in the federation.

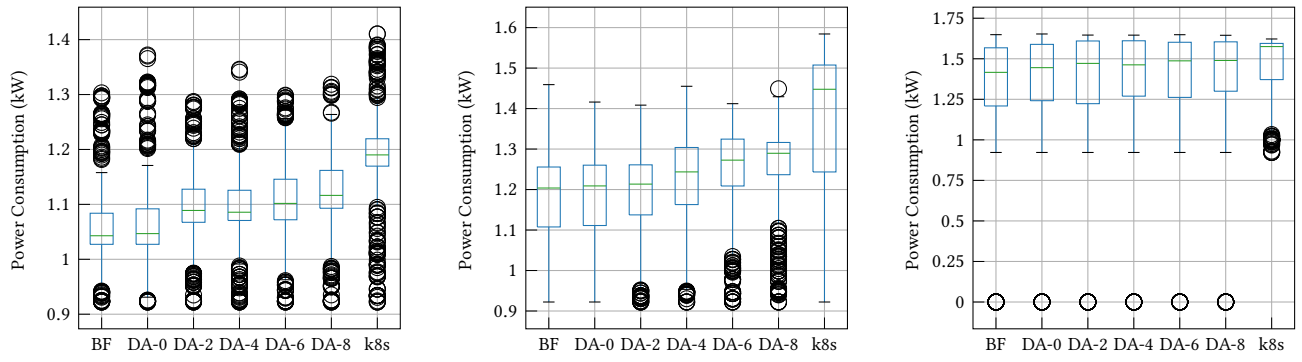
In the forthcoming works, the temporal shifting of workloads shall be evaluated alongside the present geographical shifting to enhance the already encouraging outcomes further. This shall be accomplished by incorporating information on the proportion of energy utilized by various devices (green versus brown energy) to encourage the development of sustainable cloud and edge computing infrastructure. In addition, a trade-off between QoE and energy consumption should be evaluated to provide execution guarantees for high-priority applications.

ACKNOWLEDGMENT

This work was partly supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101070473, project FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System).

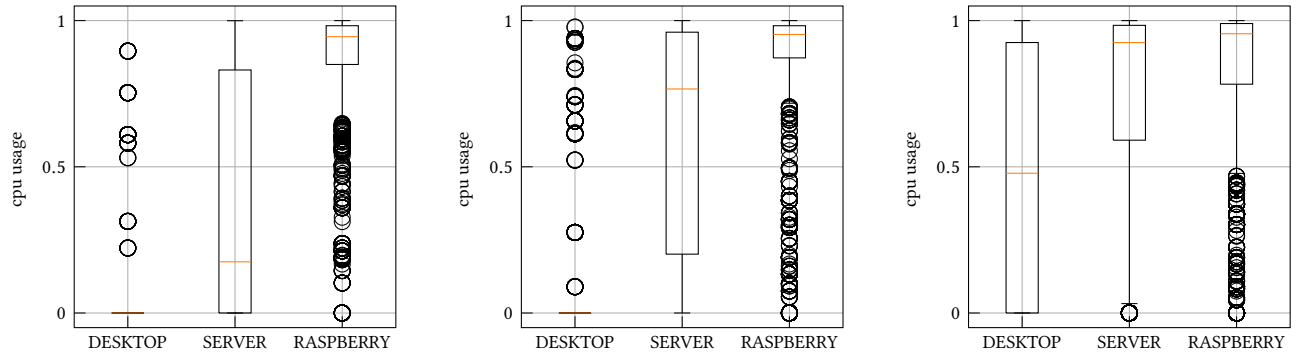
This work has been partially supported by NSF awards 1908574 and 2201536.

[3] The dataset has been characterized starting from the Alibaba's public trace available at https://github.com/alibaba/clusterdata.



(a) Duration range 1-60s. (b) Duration range 1-100s. (c) Duration 1-140s.

Figure 4: Power consumption of the infrastructure varying the duration of the submitted jobs (i.e., the load on the infrastructure) for the different configurations.



(a) Duration range 1-60s. (b) Duration range 1-100s. (c) Duration range 1-140s.

Figure 5: CPU consumption of the infrastructure varying the duration of the submitted jobs (i.e., the load on the infrastructure) for the different classes of devices included in the simulation.

REFERENCES

[1] C. Pahl, "Containerization and the paas cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[2] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," pp. 37–42, 2015.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[5] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing without borders: The way towards liquid computing," *IEEE Transactions on Cloud Computing*, 2022.

[6] D. Milojicic, "The edge-to-cloud continuum," *Computer*, vol. 53, no. 11, pp. 16–25, 2020.

[7] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–21, 2019.

[8] A. A. Alahmadi, T. E. El-Gorashi, and J. M. Elmirghani, "Energy efficient processing allocation in opportunistic cloud-fog-vehicular edge cloud architectures," *arXiv preprint arXiv:2006.14659*, 2020.

[9] M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, and S. Papavassiliou, "Enerdge: Distributed energy-aware resource allocation at the edge," *Sensors*, vol. 22, no. 2, p. 660, 2022.

[10] E. Al-Masri, A. Souri, H. Mohamed, W. Yang, J. Olmsted, and O. Kotevska, "Energy-efficient cooperative resource allocation and task scheduling for internet of things environments," *Internet of Things*, vol. 23, p. 100832, 2023.

[11] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[12] B. A. Ferreira, T. Petrović, M. Orsag, J. R. Martínez-de Dios, and S. Bogdan, "Distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams," *arXiv preprint arXiv:2109.03089*, 2021.

[13] L. Lin, P. Li, J. Xiong, and M. Lin, "Distributed and application-aware task scheduling in edge-clouds," in *2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. IEEE, 2018, pp. 165–170.

[14] J. Turner, Q. Meng, G. Schaefer, and A. Soltoggio, "Fast consensus for fully distributed multi-agent task allocation," in *Proceedings of the 33rd annual ACM symposium on applied computing*, 2018, pp. 832–839.

[15] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX annual technical conference (USENIX ATC 14)*, 2014, pp. 305–319.

[16] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, p. 133–169, may 1998. [Online]. Available: https://doi.org/10.1145/279227.279229

[17] ——, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, 2001.

[18] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on hetnets," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2015, pp. 254–258.

[19] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 638–643.

[20] Y. Zhao, S. Zhou, T. Zhao, and Z. Niu, "Energy-efficient task offloading for multiuser mobile cloud computing," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2015, pp. 1–5.

[21] W. Hua, P. Liu, and L. Huang, "Energy-efficient resource allocation for heterogeneous edge-cloud computing," *IEEE Internet of Things Journal*, 2023.

[22] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in iot-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78 685–78 697, 2019.

[23] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[24] M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, and S. Papavassiliou, "Enerdge: Distributed energy-aware resource allocation at the edge," *Sensors*, vol. 22, no. 2, p. 660, 2022.

[25] A. A. Alahmadi, T. E. El-Gorashi, and J. M. Elmirghani, "Energy efficient processing allocation in opportunistic cloud-fog-vehicular edge cloud architectures," *arXiv preprint arXiv:2006.14659*, 2020.

[26] S. Galantino, F. Risso, V. C. Coroamă, and A. Manzalini, "Assessing the potential energy savings of a fluidified infrastructure," *Computer*, vol. 56, no. 6, pp. 26–34, 2023.