Thea – a QoS, Privacy, and Power-aware Algorithm for Placing Applications on Federated Edges

Paulo Souza†, Carlos Kayser†, Lucas Roges†, Tiago Ferreto†
†School of Technology, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
Email: {paulo.severo, carlos.kayser, roges.lucas}@edu.pucrs.br, tiago.ferreto@pucrs.br

2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) | 979-8-3503-3763-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/PDP59025.2023.00028

***Abstract*—Federations between Edge Computing infrastructure providers represent a promising approach for improving the applications' Quality of Service (QoS) and the infrastructure's resource usage. At the same time, federated edges impose particular provisioning challenges, as data protection policies implemented by certain providers within a federation may conflict with the privacy requirements of services carrying out sensitive information (e.g., databases). In addition, the popularization of complex software architectures (e.g., composite applications) sets strict latency requirements that narrow the provisioning possibilities even further. Previous research efforts targeting federated edges have focused either on coupling with end-user performance requirements (e.g., latency and privacy) or on satisfying infrastructure providers' objectives (e.g., power consumption reduction), but none on balancing both. This paper presents Thea, a novel approach for provisioning composite applications on federated edges which optimizes applications' latency and privacy while reducing the infrastructure's power consumption. Simulated experiments show that Thea can achieve near-optimal results, reducing application latency and privacy issues by 50% and 42.11% and the infrastructure's power consumption by 18.95% compared to state-of-the-art approaches.**

***Index Terms*—Edge Computing, Infrastructure Provider Federations, Composite Application, Privacy, Power Consumption.**

I. Introduction

In the last decade, innovative research outcomes in telecommunications have highlighted the potential market size of real-time applications such as wearable cognitive assistants and autonomous vehicles [1]. However, low latency and high bandwidth are essential elements for such use cases to go mainstream, which calls into question where we should process data. On the one hand, holding back processing on mobile devices such as smartphones incurs overheating and battery draining, which is usually intolerable [2]. On the other hand, offloading processing to the cloud implies dealing with the intrinsic high latency resulting from the physical distance between users and data centers [3]. This challenge gave birth to a new paradigm called Edge Computing [4], which positions computing devices near end users at the Internet's edge.

Whereas the physical proximity between computing resources and end users alleviates the latency and bandwidth issues, it also introduces new challenges. As deploying large-scale data centers in the middle of urban centers is typically unfeasible, most edge infrastructure models count on heterogeneous servers deployed in small spaces with limited power supply interconnected by public networks [5] [6]. As such, edge sites display limited vertical scalability, which contrasts with the cloud's virtually unlimited capacity, and highlights the need for innovative resource management approaches.

The vertical scalability limitations of edge sites motivate infrastructure providers to explore approaches that leverage horizontal scalability. In that regard, several investigations show that providers could achieve better resource utilization by forming edge federations [7] [8] [9]. As large urban centers may house edge servers managed by multiple providers, federated providers can use each other resources, allowing infrastructure operators to rearrange applications to deliver enhanced quality of service (QoS) to end users while cutting investment by disabling resources in undercrowded regions.

Although edge federations widen the provisioning options for edge applications, software developers must build applications that can get the most out of such heterogeneous infrastructure [10]. Given that individual edge servers display limited processing power, investing in developing composite applications becomes the natural course of action. Rather than consolidating all functionality into a monolithic software unit, the features of composite applications are divided into a collection of small and independent services, which can be hosted separately on multiple edge servers [11].

While edge federations and distributed software architectures optimize resource usage in edge infrastructures, provisioning applications in such environments is not trivial. First, for composite applications to deliver satisfactory QoS levels to end users, their services must be close together in the infrastructure [12]. Second, federated providers may implement data protection policies that conflict with the privacy requirements of certain services [13]. While privacy is typically not a critical requirement for ordinary services, it is indispensable for those carrying sensitive information (e.g., databases). Finally, the infrastructure's power consumption should be as lower as possible to meet providers' financial and sustainability goals.

Previous efforts in provisioning composite applications on federated edges consider end-user QoS goals (in terms of latency and privacy) [8] [13] or infrastructure provider interests (in terms of power consumption reduction) [9], but none focus on balancing both. This paper presents Thea, a novel approach

This work was supported by the PDTI Program, funded by Dell Computadores do Brasil Ltda (Law 8.248 / 91). The authors acknowledge the High-Performance Computing Laboratory of the Pontifical Catholic University of Rio Grande do Sul (LAD-IDEIA/PUCRS) for providing support and technological resources for this project.

2377-5750/23/$31.00 ©2023 IEEE
DOI 10.1109/PDP59025.2023.00028

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.

that jointly optimizes the applications' QoS (latency and privacy) and the infrastructure's power consumption. Simulated experiments show that Thea can achieve near-optimal results, reducing application latency and privacy issues by up to 50% and 42.11% and the infrastructure's power consumption by 18.95% compared to state-of-the-art approaches.

The remaining of this paper is organized as follows. Section II presents an overview of federated edges. Next, Sections III and IV describe the considered scenario and our proposal. Then, Section V details the performance evaluation. Section VI discusses the related work and highlights our contributions. Finally, Section VII concludes the paper.

II. Background

The rise of mobile and sensor-rich applications sets low latency and high bandwidth as increasingly necessary features [1]. Although mobile devices have been exhibiting significant increases in computing power, holding processing locally on them is becoming less and less sustainable as battery life stands out as the bottleneck [10]. Worse still, delegating the processing of resource-intensive tasks to the cloud also carries negative consequences, such as reduced application responsiveness, due to the network distance between end users and data centers [3]. Consequently, experts from industry and academia have been looking for alternative approaches to meet the ever-increasing application performance requirements.

The challenge of coupling with the rigid application QoS requirements paved the way for a new paradigm called Edge Computing [4], which positions computing and storage resources at the Internet's edge, close to data sources. In contrast to mobile devices (e.g., smartphones), edge servers can be equipped with more robust power supplies, which makes them suitable for processing resource-intensive tasks. In addition, the network proximity between edge servers and end users puts Edge Computing in the spotlight as a more attractive option than the cloud for latency-sensitive applications.

While the proximity between edge servers and data sources enables low latency and high bandwidth, it also introduces significant operational challenges. As installing mid-to-large edge data centers in the middle of urban centers is typically unfeasible, edge sites commonly comprise several edge servers accommodated in small physical spaces with limited power and cooling supply [5] [6]. As a result, there is great interest in optimizing cost and resource utilization at the edge to make such a paradigm as sustainable and profitable as possible.

Given the resource constraints of edge infrastructures, composite application models are gaining significant attention as an alternative that enables multiple edge servers to cooperate in providing the features of large-scale applications [13]. In such an architecture, applications are split into independent services which communicate through message-passing protocols. A typical execution of a composite application comprises a flow of interactions that starts with the client's request and goes over the application services until the client gets feedback. Figure 1 illustrates the architecture and workflow of a composite application comprised of three services (S1–S3) hosted in different edge servers (ES3, ES1, and ES2, respectively), where services communicate across the network (i.e., traversing links L5, L2, and L1, respectively). Despite the inter-service dependency within the application workflow, each service can be independently managed and scaled, which widens the resource management possibilities.

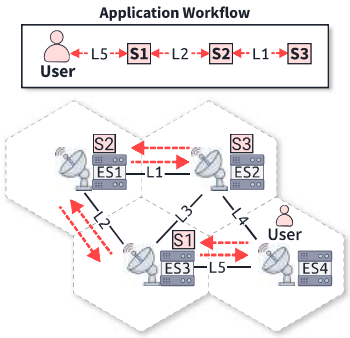


Fig. 1. Composite application architecture and workflow.

Another favorable scenario regards a federation of edge providers, which enhances cooperation and increases the availability of resources [7] [9]. In a federated edge, the multiple infrastructure providers can stipulate contracts to rent each other's resources and ensure smooth execution for applications. Such an arrangement can benefit providers and their clients to accomplish their objectives while provisioning applications in the infrastructure.

Whereas an edge federation allows infrastructure providers to deliver optimized application performance to end users while better using edge resources, it also raises concerns about potential conflicts between the data protection policies implemented by infrastructure providers and the privacy requirements of application services [13]. In such a scenario, resource management techniques need to consider several aspects, from application QoS (which can encompass several metrics such as latency and privacy) to infrastructure providers' objectives such as cost and power consumption reduction.

III. System Model

This section details the composite application provisioning scenario on federated edges tackled in this work. First, we describe the attributes and behavior of the main components in the considered edge infrastructure, including the specification of application performance requirements and the service provisioning process. Finally, we describe our optimization objectives. Table I summarizes the notations.

The edge infrastructure leverages the cellular network [14], where a set of edge servers $\mathcal{E}$ is positioned near base stations $\mathcal{B}$, which are interconnected by a set of network links $\mathcal{L}$. The map comprises a group of hexagonal cells, as in Aral et al. [15], where each cell represents the coverage area of a base station. Whereas base stations provide wireless connectivity to users, network links allow communication between edge servers at different base stations. We model a network link as $\mathcal{L}_f = \{g_f\}$, where g_f represents $\mathcal{L}_f$'s latency.

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF NOTATIONS USED IN THIS PAPER.

Symbol	Description
$\mathcal{P}$	Set of infrastructure providers
$\mathcal{E}$	Set of edge servers
$\mathcal{L}$	Set of network links
$\mathcal{U}$	Set of users
$\mathcal{A}$	Set of applications
$\mathcal{S}$	Set of services
p_i	$\mathcal{E}_i$'s provider identifier
c_i	$\mathcal{E}_i$'s CPU capacity
r_i	$\mathcal{E}_i$'s RAM capacity
q_i	$\mathcal{E}_i$'s static power consumption
m_i	$\mathcal{E}_i$'s max power consumption
$\partial(\mathcal{E}_i)$	$\mathcal{E}_i$'s overall power consumption
g_f	$\mathcal{L}_f$'s latency
u_j	$\mathcal{A}_j$'s user
w_j	$\mathcal{A}_j$'s service chain
b_j	$\mathcal{A}_j$'s communication path
$\delta(\mathcal{A}_j)$	$\mathcal{A}_j$'s latency
λ_j	$\mathcal{A}_j$'s latency SLA
t_k	$\mathcal{S}_k$'s user
h_k	$\mathcal{S}_k$'s CPU demand
d_k	$\mathcal{S}_k$'s RAM demand
ξ_k	$\mathcal{S}_k$'s privacy requirement
$x_{i,k}$	Service placement matrix
$\phi(\mathcal{U}_n, \mathcal{E}_i)$	$\mathcal{U}_n$'s trust in $\mathcal{E}_i$'s provider
$\psi(\mathfrak{B}_1, \mathfrak{B}_2)$	Latency between elements $\mathfrak{B}_1$ and $\mathfrak{B}_2$

Our modeling focuses on federated edges. As such, services are hosted on edge servers managed by a set of providers $\mathcal{P}$. Each provider can define its policies for data protection and privacy maintenance, making room for allocation decisions based on factors like the trust between users and providers and the service privacy requirements. We model an edge server as $\mathcal{E}_i = \{p_i, c_i, r_i, q_i, m_i\}$. Here, p_i references the provider that owns $\mathcal{E}_i$, where c_i and r_i denote $\mathcal{E}_i$'s CPU and RAM capacity, respectively. In addition to the capacity attributes, $\mathcal{E}_i$ displays a power consumption profile represented by attributes q_i and m_i. While q_i denotes $\mathcal{E}_i$'s static power consumption (i.e., power consumption when idle), m_i denotes $\mathcal{E}_i$'s maximum power consumption (i.e., power consumption when fully occupied). Accordingly, $\mathcal{E}_i$'s overall power consumption is given by function $\partial(\mathcal{E})$, which considers both q_i and m_i. This structure allows the incorporation of different power consumption models, as shown in Beloglazov et al. [16].

We consider a set of users $\mathcal{U}$ positioned at pre-defined positions on the map accessing their applications. For simplicity, we create a helper function $\phi(\mathcal{U}_n, \mathcal{E}_i)$, which denotes the trust of user $\mathcal{U}_n$ in the provider that manages an edge server $\mathcal{E}_i$. We model composite applications as Directed Acyclic Graphs (DAGs), where nodes represent services and edges characterize the inter-service data communication. Our modeling assumes that each application is accessed by a single user [13]. An application is represented as $\mathcal{A}_j = \{u_j, w_j, b_j, \lambda_j\}$. Attributes u_j and w_j refer to $\mathcal{A}_j$'s user and $\mathcal{A}_j$'s services list, and λ_j denotes $\mathcal{A}_j$'s latency Service Level Agreement (SLA), which is the application's latency threshold.

As services might be provisioned on different edge servers within the infrastructure, a DAG representing $\mathcal{A}_j$'s communication path (denoted by b_j) starts at the base station to which $\mathcal{A}_j$'s user is connected and travels through the network topology visiting the base stations with the servers that host each of $\mathcal{A}_j$'s services.

The latency between each pair of elements (let us say $\mathfrak{B}_1$ and $\mathfrak{B}_2$) within the network infrastructure (e.g., services, edge servers) is retrieved by $\psi(\mathfrak{B}_1, \mathfrak{B}_2)$, which encapsulates the Dijkstra's Shortest Path Algorithm [17]. In this scenario, $\mathcal{A}_j$'s latency, retrieved by $\delta(\mathcal{A}_j)$, is the sum of the latency of all links in b_j. An application service is modeled as $\mathcal{S}_k = \{t_k, h_k, d_k, \xi_k\}$, where t_k references $\mathcal{S}_k$'s user, h_k and d_k denote $\mathcal{S}_k$'s CPU and RAM demands, respectively, and ξ_k represents $\mathcal{S}_k$'s privacy requirement. The service placement is modeled by $x_{i,k}$, where:

$$x_{i,k} = \begin{cases} 1 & \text{if edge server } \mathcal{E}_i \text{ hosts service } \mathcal{S}_k \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We consider a threefold objective function (Eq. 2), which minimizes latency SLA violations, privacy SLA violations, and edge servers' power consumption, subject to constraints (3) and (4), which ensure that each service is provisioned on only one server and that the servers' capacity limit is respected, respectively. In such a scenario, latency SLA violations occur when the application latencies exceed their latency SLAs. Conversely, privacy SLA violations happen when services are provisioned on edge servers managed by providers whose trust is lower than the service privacy requirements.

$$\text{Min: } \sum_{j=1}^{|\mathcal{A}|}[\delta(\mathcal{A}_j) > \lambda_j] + \sum_{k=1}^{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{E}|}[\xi_k > \phi(t_k, \mathcal{E}_i)] \cdot x_{i,k} + \sum_{i=1}^{|\mathcal{E}|}\partial(\mathcal{E}_i) \tag{2}$$

subject to:

$$\sum_{i=1}^{|\mathcal{E}|} x_{i,k} = 1, \forall k \in \{1, ..., |\mathcal{S}|\} \tag{3}$$

$$\left[c_i \leqslant \sum_{k=1}^{|\mathcal{S}|} h_k \cdot x_{i,k}\right] + \left[r_i \leqslant \sum_{k=1}^{|\mathcal{S}|} d_k \cdot x_{i,k}\right] = 0, \forall i \in \{1, ..., |\mathcal{E}|\} \tag{4}$$

IV. THEA DESIGN

This section presents Thea, a power-aware heuristic for provisioning composite applications on federated edges, which considers the applications' privacy and latency requirements.

Thea follows a depth approach that provisions applications sequentially. So, it provisions all services of one application before moving on to the others. That being said, as edge resources are limited, the order in which applications are provisioned may yield placements with different qualities. Accordingly, Thea employs a score function (Eq. 5) to define the application placement order, considering the applications' latency and privacy requirements (Alg. 1, lines 1–3)[1].

$$norm(\zeta_{lat}(\mathcal{A}_j)) + norm(\zeta_{priv}(\mathcal{A}_j)) \tag{5}$$

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Thea algorithm.

```
1  foreach application A_j ∈ A do
2  |  Get application scores ζ_lat(A_j) (Eq.6) and ζ_priv(A_j) (Eq.8)
3  A' = Applications sorted by Eq. 5 (desc.)
4  foreach A_j ∈ A' do
5  |  foreach S_k ∈ w_j do
6  |  |  foreach E_i ∈ E do
7  |  |  |  Get edge server costs using Eq. 12–14
8  |  |  E' = Edge servers sorted by θ_sla (Eq. 11 as tiebreaker)
9  |  |  foreach E_i ∈ E' do
10 |  |  |  if E_i has capacity to host S_k then
11 |  |  |  |  Provision S_k on E_i
12 |  |  |  |  break
```

As described in Section III, the application latency accounts for the transmission time between its user and all its services. Consequently, applications accessed by users located in regions without nearby edge servers are more likely to present longer latencies than those whose users are surrounded by several edge servers. As such, Thea's application latency score function ζ_{lat} (Eq. 6) considers both the application's latency SLA and the number of edge servers close enough to the application's user to not violate its latency SLA (Eq. 7).

$$\zeta_{lat}(\mathcal{A}_j) = \begin{cases} 0 & \text{if } \alpha(\mathcal{A}_j) = 0 \\ \frac{1}{\sqrt{\alpha(\mathcal{A}_j) \cdot \lambda_j}} & \text{otherwise.} \end{cases} \tag{6}$$

$$\alpha(\mathcal{A}_j) = \sum_{i=1}^{|\mathcal{E}|} [\psi(u_j, \mathcal{E}_i) \leq \lambda_j] \tag{7}$$

Thea assumes that servers cannot accommodate services whose aggregated demand exceeds the available capacity. As such, provisioning services while avoiding privacy SLA violations involves checking whether the servers' reliability is compatible with the service privacy requirements and whether edge servers have sufficient capacity remaining. When provisioning an application, its services are provisioned according to their position in the application's service list. Services in the last positions of their application chains are the most difficult to provision, especially in the case of applications with a large number of services containing high capacity demands. Considering such complexity, Thea prioritizes such applications in its privacy score function ζ_{priv} (Eq. 8).

$$\zeta_{priv}(\mathcal{A}_j) = \sum_{\mathcal{S}_k \in w_j} \sqrt{h_k \cdot d_k} \cdot (1 + \xi_k) \tag{8}$$

After sorting applications based on their latency and privacy requirements, Thea iterates over the list of applications, selecting edge servers to host each application service (Alg. 1, lines 4–12). At this point, Thea employs a cost function (Eq. 9) that sorts edge servers based on the number of SLA violations (latency and privacy) that would be incurred by provisioning decisions (Alg. 1, lines 6–8).

$$\theta_{sla}(\mathcal{E}_i, \mathcal{A}_j, \mathcal{S}_k) = [\delta(\mathcal{A}_j) + \psi(\rho(\mathcal{A}_j, \mathcal{S}_k), \mathcal{E}_i) > \lambda_j] + [\xi_k > \phi(t_k, \mathcal{E}_i)] \tag{9}$$

[1]In this work, $norm()$ refers to the Min-Max Normalization method [18].

$$\rho(\mathcal{A}_j, \mathcal{S}_k) = \begin{cases} t_k & \text{if } \mathcal{S}_k = w_{j,0} \\ w_{j,|k-1|} & \text{otherwise.} \end{cases} \tag{10}$$

Thea employs a cost function, depicted in Eq. 11, as a tiebreaker for the number of SLA violations, considering the edge server's power consumption, the additional latency incurred to the application, and the number of services waiting to be provisioned that would be potentially harmed (in terms of privacy) by the allocation decision.

$$norm(\theta_{pw}(\mathcal{E}_i)) + norm(\theta_{lat}(\mathcal{E}_i, \mathcal{A}_j, \mathcal{S}_k)) + norm(\theta_{int}(\mathcal{E}_i, \mathcal{S}_k)) \tag{11}$$

As a portion of the infrastructure's power consumption incurs from edge servers' static power consumption, Thea focuses on reducing the infrastructure's power consumption by consolidating services on edge servers with lower power consumption profiles (Eq. 12). Also, it increases the allocation cost of inactive edge servers to comprehend scenarios where the static power consumption displays a significant portion of the edge server's overall power consumption.

$$\theta_{pw}(\mathcal{E}_i) = \frac{m_i}{c_i} + q_i \cdot \left(1 - sgn\left(\sum_{e=1}^{|\mathcal{S}|} h_e \cdot x_{i,e}\right)\right) \tag{12}$$

The edge server latency cost (denoted in Eq. 13) is given by the latency between the edge server to the previous element of the application's communication chain (which starts at the application's user and iterates over the list of application services). As Thea follows a heuristic procedure that sequentially provisions the services of each application, it lacks information on whether current decisions will harm services to be provisioned later. This way, when provisioning the last service of an application, Thea tries to avoid unnecessary latency gains incurred by occupying resources from edge servers that other applications could better use. Thea takes a similar approach regarding service privacy, assigning a higher cost to edge servers trusted by a larger number of unprovisioned services, as those servers are more likely to be required in subsequent placement iterations (Eq. 14).

$$\theta_{lat}(\mathcal{E}_i, \mathcal{A}_j, \mathcal{S}_k) = \begin{cases} 0 & \text{if } \mathcal{S}_k = w_{j,|w_j|} \\ \psi(\rho(\mathcal{A}_j, \mathcal{S}_k), \mathcal{E}_i) & \text{otherwise.} \end{cases} \tag{13}$$

$$\theta_{int}(\mathcal{E}_i, \mathcal{S}_k) = \begin{cases} \sum_{e=1}^{|\mathcal{S}|} \aleph(\mathcal{E}_i, \mathcal{S}_k) \cdot [\beth(\mathcal{E}_i, \mathcal{S}_e)] & \text{if } \mathcal{S}_k = w_{j,|w_j|} \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

$$\aleph(\mathcal{E}_i, \mathcal{S}_k) = \frac{1}{max(1, \psi(t_k, \mathcal{E}_i))} \tag{15}$$

$$\beth(\mathcal{E}_i, \mathcal{S}_k) = \sum_{y=1}^{|\mathcal{E}|} x_{y,k} = 0 \wedge \phi(t_k, \mathcal{E}_i) \geq \xi_k \tag{16}$$

After calculating the placement costs for each server in the infrastructure, Thea iterates over the sorted list of edge servers

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.

to check if they have enough free resources to host each service (Alg. 1, lines 9–12). Once Thea finds a suitable server, it stops iterating over the list of servers, provisions the service in that host, and moves on to the next service to be provisioned.

V. Performance Evaluation

This section describes the experiments that evaluate Thea in provisioning composite applications on federated edges.

A. Experiments Description

Our experimentation uses EdgeSimPy[2], a simulation toolkit written in Python that incorporates functional abstractions for servers, network devices, and applications. In addition to modeling the application placement procedure, EdgeSimPy provides multiple built-in system models for simulating various features of edge environments, including application composition and power consumption modeling, which are suitable for representing the scenario addressed in our work.

We consider an infrastructure with 18 edge servers uniformly positioned on the map through the K-Means algorithm [19]. Edge servers ship real specifications extracted from Ismail et al. [20] (Table II) while their power consumption grows linearly according to their demand, as in Beloglazov et al. [21]. In this scenario, edge servers are connected by a Mesh network topology [15] with 208 links with latency = 1[3].

TABLE II
Edge server specifications [20].

Model	CPU	RAM	Power (Idle)	Power (Max.)
Model 1	32 cores	32 GB	265 W	1387 W
Model 2	48 cores	64 GB	127 W	559 W
Model 3	36 cores	64 GB	45 W	276 W

As one of our goals is assessing privacy-aware allocation decisions, edge servers are managed by three providers. One of them, with low reliability, owns 12 edge servers, while the other two, with higher reliability, have three edge servers each. In this setting, careless allocation decisions can overload the most trusted edge servers forcing services with high privacy requirements to be provisioned on untrusted resources.

Edge servers must host 16 composite applications with different sizes (i.e., 1, 2, 4, and 8 services) and heterogeneous latency SLAs (i.e., 3 and 6). The set of potential representative use cases for such a scenario includes Augmented Reality applications, which often handle sensitive data and can be decoupled into specialized services to assure scalability and reusability [22]. For simplicity, we assume that each application is accessed by a single user positioned at a random location on the map. The evaluation scenario comprises four applications per size, totaling 60 services with heterogeneous demands (Table III) and different privacy requirements to be provisioned in the infrastructure.

To the best of our knowledge, we are the first to provision composite applications on federated edges while simultaneously optimizing end-user QoS (regarding latency and privacy requirements) and the infrastructure's power consumption. Thus, we compare our approach (Thea) against two placement strategies from the literature, Argos [13] and Faticanti et al. [8]. We have chosen these strategies as, similar to Thea, they optimize applications' latency and privacy on federated edges. As none of them focus on reducing the infrastructure's power consumption, we also consider a metaheuristic called Non-Dominated Sorting Genetic Algorithm (NSGA-II) [23], configured to find Pareto-optimal solutions for our scenario. We use NSGA-II as it presents effective outcomes in many multi-objective problems [23] [15].

TABLE III
Service demand specifications.

Specification	CPU Demand	RAM Demand
Small	2 cores	2 GB
Medium	4 cores	4 GB
Large	8 cores	8 GB
Extra Large	16 cores	16 GB

Our research strives to follow the reproducible research and open science principles. As such, the companion material hosted in a public GitHub repository[4] contains the source code, dataset, and instructions to reproduce our results.

B. NSGA-II Sensitivity Analysis

We conducted a sensitivity analysis to determine the best parameters for the NSGA-II algorithm. Without loss of generality, we define the population size as 300. Table IV presents the evaluated parameters. Each parameter combination σ was assessed by a cost function Λ (Eq. 17), which calculates the geometric mean between the infrastructure's power consumption and the number of SLA violations (latency and privacy).

$$\Lambda(\sigma) = \sqrt[3]{\sigma^{power} \cdot \sigma^{latency} \cdot \sigma^{privacy}} \tag{17}$$

TABLE IV
NSGA-II Parameters.

Parameter	Value
Population Size	300
Number of Generations	{100, 200, 300, ..., 4000}
Crossover Probability	{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}
Mutation Probability	{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}

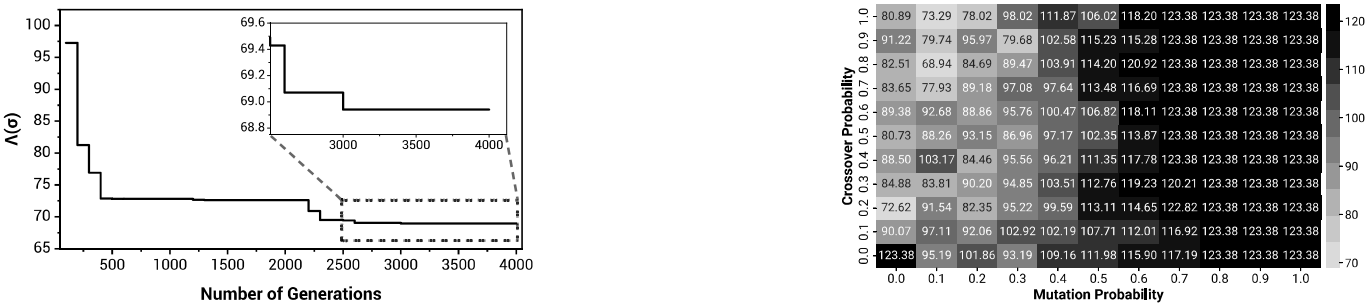
In total, our sensitivity analysis evaluated 4840 combinations of parameters. Figure 2(a) shows the impact of the number of generations on the results. As indicated by the zoomed-in region, there was no improvement after 3000 generations, which was the value used in the evaluation against the other strategies. Figure 2(b) shows the results for different combinations of crossover and mutation probability, where we can see that the best result was obtained using crossover probability = 80% and mutation probability = 10%.

[2]https://edgesimpy.github.io/

[3]Unless stated otherwise, specs are distributed uniformly in the dataset.

[4]https://github.com/paulosevero/thea/

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.



(a) Number of Generations

(b) Crossover and Mutation Probabilities

Fig. 2. NSGA-II sensitivity analysis results.

C. Comparison with Baseline Algorithms

1) Latency SLA Violations: Figure 3(a) presents the latency SLA violation results. Faticanti's strategy presented the worst outcomes (twice as many violations as Thea). Unlike the other strategies, which provision one application at a time, Faticanti's strategy simultaneously provisions the services of all applications based on their positions in their applications' chains. Also, Faticanti's strategy provisions services to servers with the highest trustworthiness available, regardless of the impact of such a decision over the applications' latency. That last decision was also implemented by Argos, which had the second-worst results (1.4x more violations than Thea).

In addition to prioritizing privacy over latency when choosing a host for a service, Argos and Faticanti's strategy prioritize edge servers nearby the application's user rather than provisioning the service in proximity to the last service provisioned from the same application. Although such a decision has no noticeable impact on smaller applications, it seriously penalizes the largest ones, as shown in Figure 3(b), as services are spread across different edge servers around users rather than being positioned close to each other, which incurs longer communication paths and, consequently, higher latencies.

Thea achieved the same number of latency SLA violations as the NSGA-II algorithm by adopting a few decisions. First, unlike Faticanti's strategy, Thea prioritizes services from applications with more strict latency SLAs. Second, rather than prioritizing privacy over latency (like Argos and Faticanti's strategy), Thea employs a score function that determines the application placement order based on both objectives rather than sacrificing one in favor of the other. Third, Thea provisions services on edge servers close to the last provisioned service of the same application instead of prioritizing edge servers close to the users, as such a decision does not necessarily lead to shorter and faster communication paths as services may end up scattered in different hosts.

2) Privacy SLA Violations: Figure 3(c) shows the number of privacy SLA violations for each evaluated strategy. Argos and Faticanti's strategy exhibited the worst results, as they indiscriminately provision services to edge servers with high trust degrees. Whereas such a decision has no impact at first, as the most trustworthy edge servers start being filled up with services containing low privacy requirements, those with higher privacy requirements need to be provisioned on untrusted edge servers, which incurs privacy SLA violations.

Unlike Argos and Faticanti's strategy, Thea takes some measures to avoid privacy SLA violations. When defining the placement order, Thea prioritizes larger applications composed of services with higher privacy requirements. Also, it avoids provisioning services to edge servers whose reliability exceeds the services' privacy requirement, making applications provisioned first less likely to harm the QoS of those provisioned later. This decision reflects in the results of Figure 3(d), which shows the number of services provisioned on edge servers with trust degrees greater than their privacy requirements.

3) Power Consumption: Figure 3(e) presents the power consumption results. Faticanti's strategy and Argos had the worst results, as they did not make decisions to reduce the power consumption. Consequently, as shown in Figure 3(f), they indiscriminately use more of the resources of edge servers with a higher power consumption profile (Models 1 and 2).

In contrast, Thea prefers the edge server models with the lowest power consumption profile (i.e., Model 3) to host the application services, which contributes to reducing the infrastructure's power consumption by 18.94% and 14.56% compared to the Faticanti's strategy and Argos, respectively, only 12.81% more than NSGA-II's Pareto-optimal outcome.

VI. Related Work

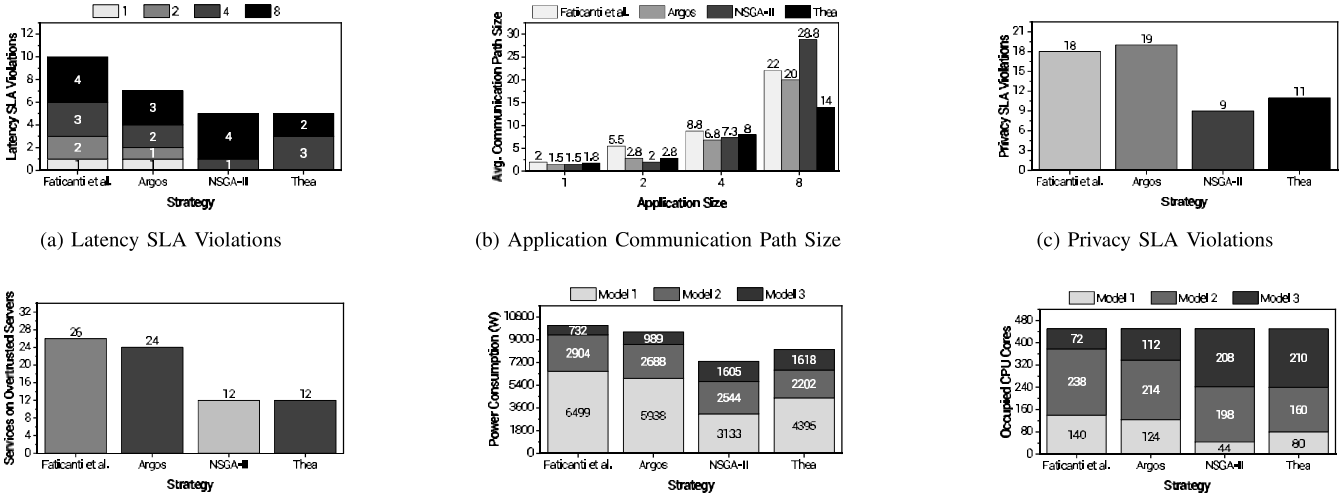
This section discusses related provisioning approaches for the edge with privacy-preserving and power efficiency goals.

A. Privacy Awareness

Qian et al. [24] present a novel service placement approach aiming to improve the infrastructure's resource usage efficiency and the applications' throughput at the edge. The authors employ a federated learning algorithm to train service placement preference models locally on end-user devices, preventing potential leaks of sensitive information on the network. Once the preference model output is obtained, the services' placement is updated accordingly.

Although holding back processing on end-user devices enhances data protection, handling resource-intensive tasks on such devices is complex, given the capacity and battery

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.



(a) Latency SLA Violations

(b) Application Communication Path Size

(c) Privacy SLA Violations

(d) Services on Overtrusted Servers

(e) Edge Servers Power Consumption

(f) Edge Servers Occupation

Fig. 3. Experimental results comparing the evaluated strategies.

constraints. In such cases, tasks can be offloaded to edge servers. Based on that, Zhu et al. [25] present a task-offloading technique that optimizes latency and resource usage while preserving user privacy. The authors model the task offloading process based on the Multi-Armed Bandit Problem, balancing user QoS (latency and privacy) and resource usage. Simulated experiments demonstrate that their proposed approach achieves near-optimal results within a few time slots.

While an edge federation allows providers to offer enhanced performance to end-users, it also introduces privacy concerns as users may hold different trust levels in certain providers. Faticanti et al. [8] introduce a placement approach for composite applications in federated edges that focus on maximizing providers' profit through optimized allocations. In their modeling, the resources managed by some providers are more expensive than others, which imposes restrictions on the applications' placement. A parallel can be established from a privacy-aware perspective, considering that some providers might be less trustworthy than others.

Souza et al. [13] tackle the privacy challenges during application provisioning in federated edges more directly. The authors argue that provisioning strategies should consider that services may exhibit different privacy requirements depending on the data they hold. In this sense, they propose Argos, an algorithm that provisions and migrates services according to user mobility considering the service privacy requirements and the trust between users and infrastructure providers.

B. Power Consumption Awareness

As edge servers are typically deployed in harsh spaces with limited power supply, designing resource management strategies that balance power efficiency with resilience is key to getting the most out of the edge. Based on this, Xu et al. [26] present an Ant Colony Optimization algorithm that provisions multiple replicas of edge servers across the infrastructure to achieve low latency and high resiliency while avoiding unnecessarily raising the infrastructure's power consumption.

Ahvar et al. [27] argue that adaptive resource management strategies are necessary to meet the needs of edge infrastructures, which are highly dynamic in terms of workload and energy costs. Accordingly, the authors present a heuristic algorithm that minimizes the edge infrastructure's power consumption and carbon emissions. In addition to defining the initial placement of applications, the proposed algorithm continuously migrates applications across the infrastructure to servers with lower energy costs.

Jeong et al. [9] optimize the provisioning of applications containing dynamic workloads on federated edges. The authors present an algorithm based on Reinforcement Learning that performs migrations to consolidate applications with low demand on a reduced set of servers. As workload peaks may lead to performance degradation of co-located applications, the algorithm migrates applications to less occupied servers as soon as it identifies that servers are about to get overloaded.

While efficiently provisioning applications on edge sites managed by a single entity is not trivial, such a problem becomes even more complex at federated edges, where providers might have conflicting goals. Considering that, Zakarya et al. [7] present a game-theoretical approach for provisioning composite applications in federated edges. The proposed optimizes conflicting goals between different providers, including end-users' QoS and infrastructure-related targets (power consumption, resource usage, and allocation cost).

C. Our Contributions

Significant research has focused on optimizing the application provisioning in Edge Computing infrastructures. Despite their contributions, the existing approaches were not designed to optimize the end-user QoS and infrastructure provider goals while provisioning composite applications in federated edges.

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.

Table V presents a high-level comparison of the related studies and our strategy (Thea). To the best of our knowledge, Thea is the first strategy that optimizes end-user QoS (i.e., latency and privacy) and infrastructure provider interests (i.e., edge server power consumption) simultaneously during the placement of composite applications in federated edges.

TABLE V
COMPARISON BETWEEN THEA AND RELATED APPROACHES.

Work	Environment	Target Metrics		
		Latency	Privacy	Power Consumption
Qian et al. [24] (PSP)	Private Edge	×	✓	×
Xu et al. [26]	Private Edge	✓	×	✓
Zhu et al. [25] (PAOTO)	Private Edge	✓	✓	✓
Ahvar et al. [27] (Deca)	Private Edge	×	×	✓
Zakarya et al. [7] (epcAware)	Federated Edge	✓	×	✓
Faticanti et al. [8]	Federated Edge	✓	✓	×
Jeong et al. [9] (ESFEC)	Federated Edge	✓	×	✓
Souza et al. [13] (Argos)	Federated Edge	✓	✓	×
This Work (Thea)	**Federated Edge**	✓	✓	✓

VII. CONCLUSIONS

Edge Computing has significant potential in coupling with the requirements of latency-sensitive applications with high bandwidth demand by bringing computational resources closer to data sources. As edge sites are usually resource-constrained, previous studies make a case for establishing federations between edge infrastructure providers to improve users' quality of service while optimizing resource usage [8] [7] [13]. Despite their significant contributions, existing approaches concentrate on fulfilling end-user goals (e.g., improving application performance and reducing privacy issues) or infrastructure provider interests (e.g., power consumption reduction) separately, but none focus on balancing both.

This paper presents Thea, a novel approach that leverages cost-based heuristic procedures to optimize the placement of composite applications on federated edges, increasing the applications' quality of service in terms of latency and privacy while reducing the power consumption of edge servers. Extensive simulated experiments demonstrate that Thea achieves near-optimal results, reducing latency and privacy issues on federated edges by up to 50% and 42.11%, respectively, and the infrastructure's power consumption by up to 18.95% compared to state-of-the-art approaches. In future work, we intend to reduce the application provisioning time by proactively provisioning part of the applications in the infrastructure.

REFERENCES

[1] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat, "Democratizing the network edge," *Computer Communication Review*, vol. 49, no. 2, pp. 31–36, 2019.

[2] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[3] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[5] J. Wang, Z. Feng, S. George, R. Iyengar, P. Pillai, and M. Satyanarayanan, "Towards scalable edge-native applications," in *Symposium on Edge Computing*, 2019, pp. 152–165.

[6] A. Aral and I. Brandić, "Learning spatiotemporal failure dependencies for resilient edge computing services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1578–1590, 2020.

[7] M. Zakarya, L. Gillam, H. Ali, I. Rahman, K. Salah, R. Khan, O. Rana, and R. Buyya, "Epcaware: a game-based, energy, performance and cost efficient resource management technique for multi-access edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1634–1648, 2020.

[8] F. Faticanti, M. Savi, F. D. Pellegrini, P. Kochovski, V. Stankovski, and D. Siracusa, "Deployment of application microservices in multi-domain federated fog environments," in *International Conference on Omni-layer Intelligent Systems*, 2020, pp. 1–6.

[9] Y. Jeong, E. Maria, and S. Park, "Towards energy-efficient service scheduling in federated edge clouds," *Cluster Computing*, pp. 1–13, 2021.

[10] M. Satyanarayanan, G. Klas, M. Silva, and S. Mangiante, "The seminal role of edge-native applications," in *International Conference on Edge Computing*, 2019, pp. 33–40.

[11] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of microservices for iot using docker and edge computing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 118–123, 2018.

[12] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2019.

[13] P. Souza, Ângelo Crestani, F. Rubin, T. Ferreto, and F. Rossi, "Latency-aware privacy-preserving service migration in federated edges," in *International Conference on Cloud Computing and Services Science*, 2022, pp. 288–295.

[14] G. Klas, "Edge computing and the role of cellular networks," *Computer*, vol. 50, no. 10, pp. 40–49, 2017.

[15] A. Aral, V. Demaio, and I. Brandic, "Ares: Reliable and sustainable edge provisioning for wireless sensor networks," *IEEE Transactions on Sustainable Computing*, pp. 1–12, 2021.

[16] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[17] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[19] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[20] L. Ismail and H. Materwala, "Escove: Energy-sla-aware edge–cloud computation offloading in vehicular networks," *Sensors*, vol. 21, no. 15, 2021.

[21] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[22] D. Vaquero-Melchor, A. M. Bernardos, and L. Bergesio, "Sara: A microservice-based architecture for cross-platform collaborative augmented reality," *Applied Sciences*, vol. 10, no. 6, p. 2074, 2020.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[24] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Information Sciences*, vol. 505, pp. 562–570, 2019.

[25] D. Zhu, T. Li, H. Liu, J. Sun, L. Geng, and Y. Liu, "Privacy-aware online task offloading for mobile-edge computing," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–16, 2021.

[26] F. Xu, Z. Yin, A. Gu, F. Zhang, and Y. Li, "A service redundancy strategy and ant colony optimization algorithm for multiservice fog nodes," in *International Conference on Computer and Communications*, 2020, pp. 1567–1572.

[27] E. Ahvar, S. Ahvar, Z. A. Mann, N. Crespi, R. Glitho, and J. Garcia-Alfaro, "Deca: A dynamic energy cost and carbon emission-efficient application placement method for edge clouds," *IEEE Access*, vol. 9, pp. 70 192–70 213, 2021.

Authorized licensed use limited to: Pontificia Universidade Catolica do Rio Grande do Sul (PUC/RS). Downloaded on December 06,2023 at 00:26:30 UTC from IEEE Xplore. Restrictions apply.