2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit) | 978-1-6654-9871-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/EuCNC/6GSummit54941.2022.9815597

Reducing Storage and Communication Latencies in Vehicular Edge Cloud

Mostafa Kishani[1], Zdenek Becvar[1], Mohammadsaleh Nikooroo[1], Hossein Asadi[2]
[1]*Faculty of Electrical Engineering, Czech Technical University in Prague,* Prague, Czech Republic
[2]*Dpt. of Computer Engineering, Sharif University of Technology*, Tehran, Iran
{kishamos, zdenek.becvar, nikoomoh}@fel.cvut.cz, asadi@sharif.edu

***Abstract*—Low-latency data access is crucial in edge clouds serving autonomous vehicles. Storage I/O caching is a promising solution to deliver the desired storage performance at a reasonable cost in vehicular edge platforms. Current storage I/O caching methods, however, are not specialized for workload characteristics and demands of autonomous vehicles and/or do not consider the communication latency between the vehicle and the base station hosting the edge cloud node. In this work, we propose a storage mechanism for vehicular edge cloud platforms taking communication, I/O cache, and storage latencies into account. We evaluate our proposed framework using realistic storage traces of vehicular services. Our framework reduces the average latency and the average latency of high-priority services by up to 1.56x and 2.43x, respectively, compared to the state-of-the-art works.**

***Keywords*—Vehicular Edge Cloud, I/O Cache, Communication, Storage.**

I. Introduction

Vehicle to infrastructure communication is a challenge on the way towards autonomous vehicles. At the same time, Mobile Edge Computing (MEC) is recognized as a promising solution to deliver low-latency vehicular services [1], [2]. Vehicular services are characterized by unstructured data types and a huge volume of data transmission, while some of these services, such as collision avoidance, are safety- and time-critical, requiring sub-millisecond latency [3], [4]. These strict requirement, apart from its communication and computation challenges, mandates a large investment on emerging data storage architectures for autonomous vehicles [4]–[7].

Each vehicular service has different priority, data access pattern, and/or data transmission volume [3], [4]. In vehicular edge cloud platforms, the latency of access to stored data is a function of communication and storage resources allocated to the vehicular service. Achieving an optimal solution via a joint adjustment of both communication and storage resource allocation is of a crucial importance. However, as this problem is NP-complete, finding the optimum solution under realistic assumptions is complicated if not impossible [8].

Previous efforts dealing with latency minimization in vehicular edge cloud platforms mainly focus on computation and/or communication latencies. In [9], the authors propose a computation offloading framework for vehicular edge cloud platforms. This work dynamically decides on either local computation or offloading to the edge cloud servers, regarding the cost and delay constraints. The authors in [10] consider content reusability among the vehicles, proposing a content-caching framework for vehicular edge cloud platforms. In [11], a framework targetting the minimum violation of latency constraints is proposed for a vehicular edge cloud, focusing on computation and communication latencies. The authors in [12] propose a computation offloading framework caring about privacy concerns. This work concentrates on allocating computing resources, taking both execution time and energy as its objective. In [13], TV white space bands is used for computation offloading of vehicular services. The sub-optimal solution manages to jointly allocate radio and power resources to reduce the cost objective which is a function of queue size. In [14], the maximum system utilization is pursued in allocating communication and computation resources in a vehicular edge computing platform, proposing a sub-optimal solution for the NP-complete problem. With a deep reinforcement learning approach, [15] jointly optimizes the latency of communication, computation, and network caching.

All works [9]–[15], however, do not address the storage latency, which is not negligible and neglecting it leads to sub-optimal solutions for data access. In our prior work [4], we investigate the storage I/O caching for the vehicular edge cloud platforms, nevertheless, only from the storage perspective not considering the communication latency between the vehicles and storage. Let us point out that the storage I/O caching addressed in our paper is completely different and independent of the content caching addressed, e.g., in [10], [15]. The storage I/O caching aims to reduce the latency of access to stored data in cache memory regardless of the content. To our best knowledge, no previous work explores data access in the vehicular edge cloud platforms considering jointly the latency of the I/O cache for storage and the communication latency.

In this paper, we focus on a joint allocation of communication and storage resources for access to data of vehicular services such as collision avoidance, map/SW download, sensing data upload, stored in the edge clouds. Our proposed framework dynamically determines the service data placement and the amount of communication and storage resources allocated to each service. Our objective function is a factor of service priority and communication and storage latencies. As the targeted problem is NP-complete, we propose a feasible sub-optimal solution using dual relaxation, jointly considering both communication and storage latencies. We validate an efficiency of the proposed framework using realistic storage traces of the services related to autonomous vehicles. The results show that

This project is supported by international mobility of research, technical and administrative staff of research organizations, with the project number *CZ*.02.2.69/0.0/0.0/18_053/0016980 and by the grant of Czech Technical University in Prague No. SGS20/169/OHK3/3T/13.

978-1-6654-9871-5/22/$31.00 ©2022 IEEE Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.

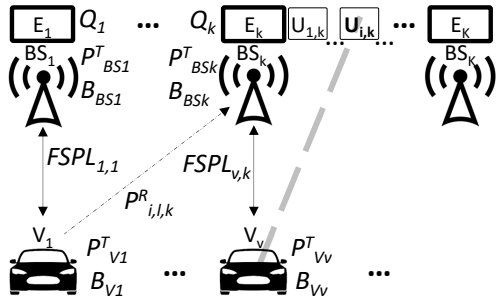


Fig. 1: Overview of the system model with unikernel i of edge cloud node k, $U_{i,k}$ associated to vehicle v (V_v).

our proposed framework reduces the average latency of the investigated services compared to the state-of-the-art works.

II. System Model

This section outlines model of the vehicular edge computing including edge cloud, mobile communication, and storage.

A. General System Overview

We assume the vehicular edge cloud platform consists of K edge cloud nodes connected via mobile networks to the vehicles. Each edge cloud node is collocated with one base station of mobile network, i.e., there are also K base stations. Offloading the services of vehicles into the edge cloud takes place in the form of unikernels [16]. A vehicle has multiple unikernels for different services, such as collision avoidance and map update, while the unikernels can be hosted in different edge cloud nodes. In each edge cloud node, a hypervisor is responsible for managing the storage resources. The hypervisor allocates the resources to each unikernel independently.

We assume each edge cloud node (server) is collocated with a base station which is responsible for allocating the communication resources. The storage resources managed by the hypervisor also include main storage resources and I/O cache resources. We assume that the main storage allocation from each edge cloud node is always feasible, i.e., the edge cloud nodes are equipped with enough provisioned main storage resources. Hence, we concentrate on efficient I/O cache and communication resource management which is crucial in the low-latency vehicular edge platforms. As in [17], [18], we further expect each edge cloud node allocates I/O cache to the hosting unikernels from the node's local resources.

In the vehicular edge storage environment, we characterize each unikernel by the following parameters:

- Latency of storage request, i.e., the time it takes from sending a storage read request by the vehicle, to receiving data from the edge cloud node at the vehicle. The latency of the storage request for the unikernel i hosted by the edge cloud node k is defined as the aggregation of the communication latency $D^C_{i,k}$ and the storage latency $D^S_{i,k}$, while the communication latency includes the uplink latency, $D^C_{UL_{i,k}}$ and the downlink latency, $D^C_{DL_{k,i}}$:

$$D_{i,k} = D^C_{UL_{i,k}} + D^C_{DL_{k,i}} + D^S_{i,k} \tag{1}$$

- Frequency (number) of storage accesses F, i.e., the number of storage accesses per time unit. As the accesses may have different size, we consider l as the unit storage access size and translate any arbitrary storage access with a size l' to multiple accesses with the unit size l, hence F represents the number of unit accesses with the size l.
- Priority ρ as a fixed, predefined value ($0 \leq \rho \leq 1$); the greater ρ represents the higher priority and criticality of the unikernel.

B. Communication Latency

The communication latency is composed of downlink and uplink communication.

1) Uplink Latency: The uplink communication latency $D^C_{UL_{i,k}}$ between the vehicle v to which the unikernel i is associated to, and the base station with the edge node k hosting the unikernel i is defined as:

$$D^C_{UL_{i,k}} = \frac{l \times 8}{R^{UL}_{i,k}} \tag{2}$$

where l is the size of the unit storage access (in Bytes) and $R_{i,k}$ is the uplink communication data rate over the channel between the vehicle v owning the unikernel i and the base station k (in bits per second), defined as:

$$R^{UL}_{i,k} = B_{i,k} \times log_2(1 + \frac{P^R_{i,k}}{\sigma + I_{i,k}}) \tag{3}$$

where $B_{i,k}$ is the bandwidth allocated for the communication between the vehicle v to which the unikernel i is associated and the base station k, $P^R_{i,k}$ is the received signal power at the base station of the edge cloud node k hosting the unikernel i from the vehicle v, σ is the power of thermal noise experienced by the base station, and $I_{i,k}$ is the interference power imposed to the base station by other vehicles.

According to Friis' transmission equation, $P^R_{i,k}$ is:

$$P^R_{i,k} = P^T_{V_i}(\frac{\lambda}{4\pi d_{i,k}})^2 \tag{4}$$

where $P^T_{V_i}$ is the transmission power of the vehicle v to which the unikernel i is associated to, λ is the communication wavelength, and $d_{i,k}$ is the distance between the base station k and the vehicle v owning the unikernel i.

2) Downlink Latency: Downlink communication latency $D^C_{DL_{k,i}}$ between the base station k of the edge cloud node hosting the unikernel i and the vehicle v to which the unikernel i is associated to is similarly defined as (2) by substituting $R^{UL}_{i,k}$ with $R^{DL}_{k,i}$, where $R^{DL}_{k,i}$ is the downlink communication data rate over the channel between the base station k and the vehicle v to which the unikernel i is associated to (in bits per second). We define the downlink data rate as:

$$R^{DL}_{k,i} = B_{k,i} \times log_2(1 + \frac{P^R_{V_{k,i}}}{\sigma + I_{k,i}}) \tag{5}$$

Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.

where $B_{k,i}$ is the bandwidth allocated for the communication between the base station k and the vehicle v, $P^R_{V_{k,i}}$ is the received signal power at the vehicle from the base station k, σ is the power of thermal noise experienced by the vehicle v, and $I_{k,i}$ is the interference imposed to the vehicle v from all base stations but the base station k. Analogously to (4), $P^R_{V_{k,i}}$ is defined as:

$$P^R_{V_{k,i}} = P^T_{k,i}(\frac{\lambda}{4\pi d_{k,i}})^2 \tag{6}$$

where $P^T_{k,i}$ is the transmission power of the base station k to the vehicle v to which the unikernel i is associated, and $d_{k,i}$ is the distance between the base station k and the vehicle v.

For both downlink and uplink, we assume the communication bandwidth and power are allocated by the transmitting device proportionally to the number of unikernels of the given hosted by the edge cloud node of the base station serving the vehicle, i.e., $B_{k,i} = B_{BS_k}/M_k$ and $P^T_{k,i} = P^T_{BS_k}/M_k$.

C. Storage Latency

We consider the main storage uses either *Hard Disk Drive* (HDD) or mid-range *Solid-State Drive* (SSD). We also consider a three layer I/O cache architecture of *Dynamic Random-Access Memory* (DRAM), *Non-Volatile Memory* (NVM), and SSD, while considering Read-Only policy for DRAM cache, Write-Back policy for NVM and SSD cache, *Least Recently Used* (LRU) replacement policy in all cache layers, exclusive cache management in all layers, and promotion to DRAM upon DRAM cache miss, as conventionally considered in state of the art I/O cache architectures [4], [17], [18]. The storage accesses is composed of read and write requests. All write requests have typically similar latency, as these are always written in the first cache layer and are not influenced by the I/O cache management policy [17], [18]. Moreover, the storage write requests are usually not in the critical path of edge cloud storage systems. Hence, we concentrate on optimizing the read requests, considering a constant latency for all write requests.

Average storage read latency for the unikernel i of the edge cloud node k is a function of the cache hit ratio, as well as the latency of cache memory media and main storage media [4]. Considering a single cache memory layer (extendable to more layers [4]), the average storage read latency is:

$$D^S_{i,k} = \eta_{i,k} \times L_{cache} + (1 - \eta_{i,k}) \times L_{main} \tag{7}$$

where $\eta_{i,k}$ is the cache memory hit ratio for the unikernel i of the edge cloud node k, L_{cache} is the latency of cache memory, and L_{main} is the latency of main storage. The cache memory hit ratio η is calculated by *Hit Ratio Curves* (HRC) [19] that plot the cache hit ratio as a function of the cache size by analyzing the stack distance. The stack distance of an access to the memory address a is the number of unique references to other memory addresses before a successive access to a. In a cache size of s with LRU replacement policy, all memory accesses with the stack distance of $s-1$ and below have a hit. The HRC is a histogram of stack distance for individual references. η of the cache size s is equal to the cumulative distribution function of HRC at the point $s-1$. Note that, in each monitoring period, the HRC is constructed once. We assume that η is a concave function of cache size (Q), which is conveniently considered as a mostly practical assumption in mainstream applications when the optimum cache replacement policies, such as LRU, are employed [20]. Finally, we assume that the main storage is slower than the cache memory, hence:

$$L_{cache} < L_{main} \tag{8}$$

III. Dynamic Decision-Making Problem

In this section, we present the dynamic decision-making problem, as well as our objective function and its constraints. The problem is to minimize the storage access latency of services provided to vehicles considering communication and storage aspects. Our objective function should take all three unikernel characteristics we concern about, i.e., priority ρ, frequency F, and latency D, into account. The objective function is defined as the multiplication of these three characteristic, $\rho \times F \times D$, hence, the latency of a unikernel with a higher priority and frequency should be of a higher impact on the objective function. Thus, we formulate the problem as:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{M_k} \rho_{i,k} \times F_{i,k} \times D_{i,k} \tag{9}$$

$$s.t. \quad \sum_{i=1}^{M_k} Q_{i,j,k} \leq Q_{j,k}, \quad \forall k \in \langle 1, K \rangle, \forall j \in \langle 1, N_l \rangle \quad (a)$$

$$\sum_{j=1}^{N_1} Q_{i,j,k} \leq \Delta_{i,k} + 1, \quad \forall k \in \langle 1, K \rangle, \forall i \in \langle 1, M_k \rangle \quad (b)$$

where $\Delta_{i,k}$ is maximum stack distance in the unikernel i of the edge cloud node k, M_k is the number of unikernels in the edge cloud node k, K is the number of edge cloud nodes, $\rho_{i,k}$ is the priority of the unikernel i of the edge cloud node k, $F_{i,k}$ is the storage access frequency of the unikernel i of the edge cloud node k, $Q_{i,j,k}$ is the cache memory allocated from the layer j to the unikernel i of the edge cloud node k, and $Q_{j,k}$ is the size of the cache memory layer j in the edge cloud node k.

The first constraint in (9) assures that the aggregation of allocated cache memory from the layer j of edge cloud node k is not greater than the size of cache memory installed in edge cloud node k. The second constraint guarantees the aggregation of cache space allocated to the unikernel i from different cache layers is not greater than the optimal cache space determined by the maximum stack distance, $\Delta_{i,k}$.

IV. Proposed Solution

The optimization problem presented in Section III is NP-complete. In this section, we propose a sub-optimal solution using dual relaxation. To this end, we first relax the cache size constraint on each node (constraint (a) in (9)) and replace it with a constraint on the overall cache size allocated from all edge cloud nodes. In the relaxed problem, the cache allocation is always possible from the edge cloud node whose base station has the highest channel quality, resulting to the optimum communication latency. Thus, we propose an optimal solution for the relaxed problem determining the I/O cache size allocated to each service. Afterwards, we propose a sub-optimal solution for the cache placement jointly considering both communication and storage latencies. For the service placement, we initially assume the I/O cache allocated to all services have the same size and we map the problem into a perfect matching problem in a bipartite graph. However, when the actual optimum cache

Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.

size of each service is allocated, the cache size constraint can be violated in some edge cloud nodes. To avoid the violation of cache size constraint, we propose a direct heuristic method to re-place some services from violated edge cloud nodes into the edge cloud nodes with free I/O cache space.

A. Constraints Relaxation and optimum cache allocation

To solve the problem, we first relax the cache size constraint on each edge cloud node, assuming the efficient cache size can be allocated from any edge cloud node, as far as the overall allocated cache does not exceed the aggregated cache sizes available on the edge cloud nodes. This relaxation allows us to replace the constraint (a) in (9) with $\sum_{k=1}^{K}\sum_{i=1}^{M_k} Q_{i,j,k} \leq \sum_{k=1}^{K} Q_{j,k}$. As such, the cache allocation is always possible from the edge cloud node whose base station has the highest channel quality, resulting in the minimum communication latency.

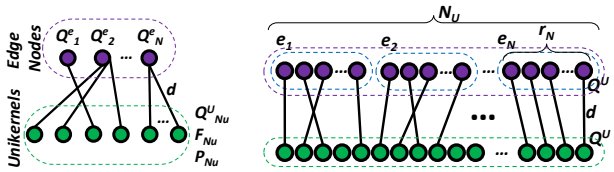
In Algorithm 1, we propose a definite method to find the optimum solution to (9) with relaxed constraint (a). We consider zero cache allocation for the initial state and we allocate the cache chunks one-by-one until all chunks are allocated. Per iteration, we find the unikernel i with the highest objective achievement, $\alpha = \rho_i \times F_i \times \Psi_i(Q_i)$, defined as the increase in objective function when allocating one extra cache chunk to the unikernel. Let Q_i be the current number of cache chunks allocated to the unikernel i and $\Psi_i(Q_i)$ be the stack distance histogram of i at the point Q_i. Regarding the definition of the stack distance, $\Psi_i(Q_i)$ is an increase in the hit ratio caused by the cache size increase from Q_i to Q_i+1. For example, when the cache size increases from 0 to 1, the hit ratio increases by $\Psi_i(0)$. Let $\eta_i(Q_i)$ be the current hit ratio of the unikernel i and $\eta_i(Q_i+1)$ be the hit ratio of i after allocating one extra cache chunk to it; then, we have:

$$\eta_i(Q_i+1) = \eta_i(Q_i) + \Psi_i(Q_i) \tag{10}$$

We sort the unikernels by their objective achievement and allocate one cache chunk from the edge cloud node k whose base station has the highest channel quality to the unikernel i with the maximum objective achievement. Afterward, we increase $Q_{i,k}$ and Q_i^U to track the number of cache chunks allocated to each unikernel, where $Q_{i,k}$ is the number of cache chunks allocated from the edge cloud node k to the unikernel i and Q_i^U is the number of chunks allocated to the unikernel i. As we assume allocating cache to the unikernel only from one edge cloud node, $Q_i^U = Q_{i,k}$.

Algorithm 1 I/O Cache Size Determination

```
for i ← 1 to N_U do
    Q_i^U = 0
end for
while Σ_{j=1}^{N_U} Q_j^U ≤ Σ_{k=1}^{K} Q_k^E do
    i = max(ρ_r × F_r × Ψ_r(Q_r^U)), r ∈ ⟨1, N_U⟩
    if Q_i^U < Δ_i + 1 then
        e = min(d_{i,k}), k ∈ ⟨1, K⟩
        Q_{i,e} += 1
        Q_i^U += 1
end while
```



(a) Cache placement problem as a weighted bipartite graph, matching in the unikernels part.

(b) Cache placement in dual placement problem represented as a perfect matching problem in a bipartite graph with the same number of vertices in both parts.

Fig. 2: Cache placement problem graphs

B. Cache Placement Problem

In previous subsection, the optimal solution for the cache size allocation is proposed for relaxed cache placement constraint. In such relaxed problem, we find the optimal cache size considering the constraint limiting the total cache size allocated to the unikernels to the aggregation of the cache space installed throughout the edge cloud nodes, i.e., assuming:

$$\sum_{i=1}^{N_U} Q_i^U = \sum_{k=1}^{K} Q_k^E \tag{11}$$

Our objective is to find a feasible assignment of the unikernel to the edge cloud nodes by considering the allocated cache sizes resulting into the minimum objective function value. Hence, we exploit a weighted bipartite graph matching in the unikernels part, as shown in Fig. 2a. Given our relaxed problem, by removing the relaxed constraints from (9) and substituting the communication latency from (2), the goal is to find a feasible solution to the following problem:

$$\min \sum_{k=1}^{K}\sum_{i=1}^{M_k} \rho_{i,k} \times F_{i,k} \times \frac{l \times 8}{B \times log_2(1 + \frac{P^T(\frac{\lambda}{4\rho d_{i,k}})^2}{N+I})} \tag{12}$$
$$s.t. \quad \sum_{i=1}^{M_k} Q_{i,k} = Q_k, \quad \forall k \in \langle 1, K \rangle$$

To solve this problem, we first define a dual problem by considering equal cache sizes assigned to all unikernels and map it to a Balanced Assignment problem that can be optimally solved in polynomial time. In the dual problem we consider each unikernel has the cache size Q^U ($Q_i^U = Q^U, \forall i \in \langle 1, N_U \rangle$), while the cache size of the edge cloud node, Q^E is divisible to Q^U ($Q_k^E | Q^U, \forall k \in \langle 1, K \rangle$). We replace each edge cloud node of the cache size Q_k^E with r_k edge cloud nodes of the cache size $C_k'^E = Q^U$, where $r_k = \frac{Q_k^E}{Q^U}$. Then, from (11), we can show that the number of edge cloud nodes is equal to the number of unikernels, while each unikernel is assigned to one edge cloud node (Fig. 2b). Assigning the unikernels to the edge cloud nodes is the balanced assignment problem, optimally solvable in polynomial time using Hungarian algorithm. However, due to the previous relaxation, the cache size constraint can be violated in some edge cloud nodes (due to cache space over-allocation), while some edge cloud nodes have free unallocated cache space. As the previous cache placement problem assumes the overall cache allocated to the unikernels is equal to the overall cache installed on the edge cloud nodes, asserted in (11), the aggregation of over-allocated cache space (the cache

Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.

TABLE I: Combination of services in examined workloads.

Workload	S1: Collision Avoidance	S2: Sensing Data Upload	S3: Map/Update Download	S4: Other
A	1x	1x	1x	-
B	1x	2x	2x	-
C	1x	3x	3x	-
D	1x	1x	1x	1x

space allocated in an edge node over its installed cache size) is equal to the aggregation of unallocated cache space in the rest of edge cloud nodes.

To derive a feasible cache placement with the constraint (a) in (9), we define the Replacement Cost Coefficient of unikernel i, $\theta_i = F_i \times P_i$, as the product of access frequency and priority of the unikernel i and we start re-placing the unikernels with smallest θ to the edge cloud node with free cache space whose base station has the highest channel quality. In the optimal solution proposed in Section IV-A, the evaluated optimum cache space is equal to the aggregation of cache resources installed on the edge cloud nodes. Hence, we make sure this direct algorithm can always place all unikernels with their evaluated optimum cache space. However, in practice there is a possibility of *internal fragmentation* after running the re-placement algorithm. We define internal fragmentation as the free cache space in an edge node that is too small be allocated to any unikernel in the edge nodes with over-allocated cache space. As an example, suppose we have the edge cloud node e_1 with 10 over-allocated cache chunks, while the edge cloud nodes e_2 and e_3 have 8 and 2 free cache chunks, respectively. Suppose that the unikernel u_1 of e_1 with the optimum cache size of 9, has the smallest θ. The algorithm starts re-placing the unikernels from the one which have the smallest θ, in this example u_1. In this example, however, the algorithm fails to finish the re-placement of remaining unikernels, as u_1 cannot be allocated to either of e_2 and e_3. In this case, we take either of two following decisions that results in the smaller objective function value; a) re-place u_1 to the edge node with the largest free cache space, e_2, and allocate all free cache space of e_2 to u_1, and b) place u_1 into the edge node it is already placed, e_1, with existing available cache resources.

V. Performance Analysis

In this section, we present simulation setup and the performance of the proposed framework in terms of average latency.

A. Simulation Setup

To evaluate the proposed framework, we post-processes the real block-layer traces of the edge cloud node serving the autonomous vehicles. To model vehicle's movement, we capture highway tracks data from highD dataset [21]. For each vehicle, we run four vehicular services including collision avoidance, sensing data upload, map/update download, and other services, while each individual service runs as a single unikernel. The characteristics of each service are detailed in Table II. We conduct the simulations with four representative workloads constructed by the combination of mentioned services with different intensities, as detailed in Table I.

As no previous work considers both communication and storage latency together, we compare the proposed framework

TABLE II: Characteristics of autonomous vehicle service types

	S1: Collision Avoidance	S2: Sensing Data Upload	S3: Map/Update Download	S4: Other
Read/Write %	70/30	0/100	100/0	70/30
Access Pattern	Zipf 1.2	Sequential	Sequential	Zipf 1.2
Priority	Real-time	Normal	Normal	Normal

TABLE III: Simulation Parameter Values

Notation	Parameter	Value
K	Number of edge cloud nodes	10
N_U	Number of unikernels (4 unikernels per vehicle)	1000
L_{cache} (DRAM)	I/O cache memory latency	7.9 ns
L_{cache} (NVM)	I/O cache memory latency	70 ns
L_{cache} (SSD)	I/O cache memory latency	25 us
L_{main}	Main storage latency	5 ms
P^T	Transmit power of edge cloud node base station	46 dBm
P_i^T	Transmit power of vehicle	40 dBm
l	Size of unit storage access	4 KB
σ	Noise power	-90 dBm
B	Bandwidth of edge cloud node base station	100 MHz

with two state-of-the-art I/O cache architectures for edge and virtualized platforms, PADSA [4] and ETICA [18]. Both works consider a single edge node and do not take the communication latency into account. To be able to compare communication latency, we consider both PADSA and ETICA allocate unikernels to the edge cloud node with free cache space whose base station has the highest channel quality, assuming that the allocation is performed by order, started from the unikernel 1 and finished with the unikernel N_U. Table III shows the simulation parameter's values. We consider each edge cloud node is collocated with one base station deployed along the road with equal inter-site distance. We also consider a bandwidth allocation with orthogonal frequency-division multiplexing (OFDM) and frequency of 2.6GHz. We use Friis transmission formula to calculate the signal path loss in free space, considering isotropic antennas for both vehicle and base station, respectively shown in (4) and (6). We consider a high priority for collision avoidance services by setting $\rho = 1$ and normal priority for the rest of vehicular services with $\rho = 0.5$.

Constructing the stack distance histogram Ψ and hit ratio curve η for each unikernel is of time complexity $O(NlogM)$ and space complexity $O(M)$, where N is the total number of storage requests (length of workload) and M is the number of unique references (addresses) in the workload [19]. Note that, in each monitoring period, Ψ and η are constructed once.

B. Results

Fig. 3 compares the average latency of the proposed framework with the state of the art works. The proposal improves the average latency compared to the state of the art works in all examined workloads, thanks to an efficient service placement reducing the communication latency. Note that PADSA and ETICA perform the same in terms of communication latency and their performance difference is due to the storage latency. We observe the most notable latency improvement in the workload A. This observation is justified by the fact that the workload A has the highest intensity of the high-priority services and benefits more from the proposed framework, which considers the priority of services in the objective function.

Fig. 4 shows the a share of the communication and storage

Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.

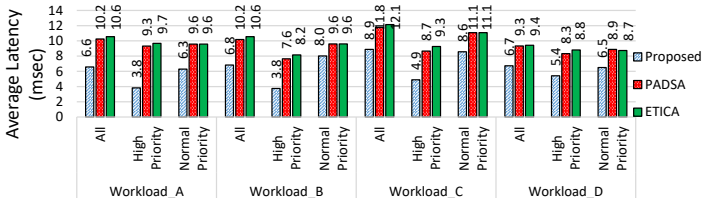


Fig. 3: Aggregation of communication and storage latency

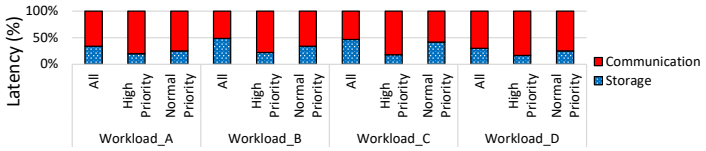


Fig. 4: Share of communication and storage in overall latency

latencies in the overall latency for the proposed framework. The figure shows the dominance of the communication latency contributing to the overall latency with 51% to 83%. We observe a higher share of the communication latency for the high-priority services. This observation shows that the proposed framework results in a lower latency for the high-priority services. This latency reduction is mostly due to an improved storage latency.

Fig. 5 shows the percentage of unikernels placed in the edge cloud node whose base station has the highest channel quality. The results show that more than 69% of unikernels (up to 79% in the case of workload A) are placed in the edge cloud node whose base station has the highest channel quality. In the workload A, the higher percentage of unikernel placement to the base station with the greatest base station to vehicle channel quality is justified by the lower density of the normal-priority services in the workload , making a room for a more efficient placement of the high-priority services.

VI. Conclusion

In this paper, we have proposed a dynamic decision-making system for communication and I/O cache resource allocation and placement in the vehicular edge cloud platforms. A sub-optimal solution is proposed using dual relaxation and an algorithm finding the optimum I/O cache size in the relaxed problem is presented. For the service placement with the minimum communication latency, we first assume the I/O cache allocated to all services have the same size and map the problem into a perfect matching problem in a bipartite graph, solvable in polynomial time. Finally, we propose a direct heuristic to remove the cache size constraint violation in some edge cloud nodes. The proposed framework reduces the average latency and the average latency of high-priority services respectively by up to 1.56x and 2.43x compared to the state-of-the-art. In the future works, we explore more realistic vehicular edge platforms. We also investigate how vehicle-to-vehicle communication can be leveraged in reducing communication and storage latency of vehicular services in practical scenarios.

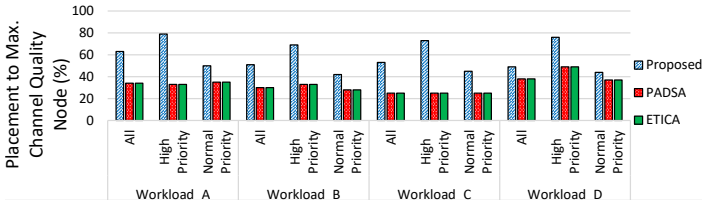


Fig. 5: Percentage of unikernels placed in the edge cloud node which base station has the highest channel quality

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communication Surveys & Tutorials*, vol. 19, no. 3, 2017.

[2] J. Plachy *et al.*, "Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users," *IEEE Trans. on Network and Service Management*, vol. 18, no. 2, 2021.

[3] P. Porambage *et al.*, "Survey on multi-access edge computing for internet of things realization," *IEEE Communication Surveys & Tutorials*, vol. 20, no. 4, 2018.

[4] M. Kishani *et al.*, "Padsa: Priority-aware block data storage architecture for edge cloud serving autonomous vehicles," in *VNC*, 2021.

[5] S. Liu *et al.*, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, 2019.

[6] M. Kishani *et al.*, "A modeling framework for reliability of erasure codes in ssd arrays," *IEEE Trans. Comput.*, vol. 69, no. 5, 2019.

[7] ——, "Dependability analysis of data storage systems in presence of soft errors," *IEEE Trans. on Reliability*, vol. 68, no. 1, 2019.

[8] L. Liu *et al.*, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, no. 3, 2021.

[9] K. Zhang *et al.*, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, 2017.

[10] X. Xu *et al.*, "Edge content caching with deep spatiotemporal residual network for iov in smart city," *ACM TOSN*, vol. 17, no. 3, 2021.

[11] M. Tareq *et al.*, "Ultra reliable, low latency vehicle-to-infrastructure wireless communications with edge computing," in *IEEE GLOBECOM*, 2018.

[12] X. Xu *et al.*, "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, 2019.

[13] J. Du *et al.*, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. on Vehicular Tech.*, vol. 68, no. 2, 2018.

[14] Y. Dai *et al.*, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE IoT Journal*, vol. 6, no. 3, 2018.

[15] Y. He *et al.*, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. on Vehicular Tech.*, vol. 67, no. 1, 2017.

[16] D. Grewe *et al.*, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Workshop on Mobile Edge Communications*, 2017.

[17] R. Koller *et al.*, "Centaur: Host-side ssd caching for storage performance control," in *ICAC*. IEEE, 2015.

[18] S. Ahmadian *et al.*, "Etica: efficient two-level i/o caching architecture for virtualized platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, 2021.

[19] C. A. Waldspurger *et al.*, "Efficient MRC construction with SHARDS," in *FAST*, 2015.

[20] N. Beckmann *et al.*, "{LHD}: Improving cache hit rate by maximizing hit density," in *NSDI*, 2018.

[21] R. Krajewski *et al.*, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *ITSC*, 2018.

Authorized licensed use limited to: CZECH TECHNICAL UNIVERSITY. Downloaded on March 11,2024 at 08:08:49 UTC from IEEE Xplore. Restrictions apply.