UniDRM: Unified Data and Resource Management for Federated Vehicular Cloud Computing

WISEBORN M. DANQUAH1,2 AND D. TURGAY ALTLAR1
1Faculty of Computer Engineering and Informatics, Istanbul Technical University, 34469 Istanbul, Turkey
2University of Cape Coast, Cape Coast, Ghana
Corresponding author: Wiseborn M. Danquah (wdanquah@itu.edu.tr)
The work of Wiseborn M. Danquah was supported in part by the Turkcell-Istanbul Technical University Researcher Funding Program.

ABSTRACT The demand for computational resources in vehicular environments has increased due to the deployment of numerous intelligent transportation systems in the last decade. The federated vehicular cloud, a variant of vehicular cloud computing where resources embedded in individual vehicles are organized as a single unit to provide cloud services, is considered as an emerging alternative to the conventional cloud platforms for the execution of computationally intensive and delay-sensitive applications. However, the federated vehicular cloud is beset with a capacity-constrained communication channel and limited resource capacity in individual vehicles, leading to challenges in data and resource management. To address these challenges, we propose UniDRM, a unified data and resource management framework for the federated vehicular cloud. The UniDRM organizes vehicles on the road into clusters based on their mobility and resource characteristics, such as resource cost, resource credibility level, resource type, and available resource capacity. The data of computationally intensive tasks are then partitioned using our proposed analytical model and assigned to individual vehicles in the cluster for parallel execution. Three data partitioning and scheduling schemes: time-aware, cost-aware, and reliability-aware, are proposed in this study to execute time-critical tasks, low-cost tasks, and high-security tasks, respectively. Through realistic simulations, a comparative analysis of the proposed partitioning and scheduling schemes is presented.

INDEX TERMS Vehicular cloud computing, federated vehicular cloud, resource management, resource-based clustering, resource ranking, divisible load partitioning, divisible load scheduling.

I. INTRODUCTION In the last two decades, Intelligent Transport System (ITS), which is the introduction of technology to make transportation safer and more efficient [1], has experienced many advancements. Notable developments in ITS include the design and deployment of smart vehicles that communicate with other devices.

Communication among vehicles, personal devices, and infrastructure along roads is enabled by Vehicular Ad-hoc Network (VANET), an infrastructure-less wireless network of vehicles. Through the 802.11p [2] and IEEE604 [3] VANET protocols, vehicles can connect to and communicate with devices in their transmission range, resulting in the Vehicle-to-Everything (V2X) communication network.

Currently, V2X is receiving much attention and is expected to be deployed in cities and urban areas as an essential component of smart city systems. The full deployment of V2X will lead to a wireless network with hundreds to thousands of connected vehicles and devices. This envisioned large network in vehicular environments has led to the emergence of new computing paradigms such as the Internet of Vehicles (IoV).

IoV is defined as the intelligent integration of vehicles, people, and devices as a network to provide services [4]. It is expected to support ITS and smart city applications such as collision avoidance systems and citywide intelligent traffic control systems. With advances in communication technologies such as 5G [5], we believe that in the next decade, IoV will also serve as a dynamic backbone network framework for sharing Big Data and vehicular resources.

Through the IoV, Owners of vehicles with underutilized onboard computing resources can donate them via Vehicular Volunteer Computing (VVC) [6] or lease them in the form of cloud services. Similar to utility computing [7], clients can access the cloud services on a pay-as-you-go basis.
This novel paradigm of vehicular resource sharing is referred to as vehicular cloud computing. It was first proposed by [8]–[10] to provide cloud services such as storage as a service (StaaS), computation as a service (CaaS), network as a service (Naas), and cooperation as a service (CaaS) [10], [11].

In order to efficiently provide the different cloud services, two main classes of vehicular cloud provisioning models have been proposed in the literature: the peer-to-peer provisioning model and the federated provisioning model [12]. This classification is based on the type of services provided, the number of vehicles involved in the service provision, and the resource management scheme used.

The peer-to-peer provisioning model is a decentralized model in which vehicle owners individually rent their resources to customers in the form of cloud services. Considering that the services are provided independently of the resources of other vehicles, all resource management tasks, including resource monitoring and task scheduling, are performed by the individual resource owners.

Since the peer-to-peer vehicular cloud provisioning model consists of resources from a single vehicle, it may not be suitable for running applications that require high resource capacity beyond what a single vehicular resource node can provide.

In the federated provisioning model, a centralized resource manager in a Region of Interest (RoI) of a road acquires resources from different vehicle owners and organizes them in the form of a pool to provide cloud services to clients as a single logical entity. A typical example of a federated vehicular cloud proposed in [13] is a citywide vehicular cloud platform consisting of resources from different vehicles under the management of a city government to provide security services to vehicles.

Unlike the peer-to-peer provisioning model, the federated provisioning model can serve applications that require resources with high computing and storage capacities by pooling many locally available resources together to form a single but larger resource image.

The federated vehicular cloud model is promising due to its potential commercial benefits and ability to perform computationally-intensive and data-intensive tasks. However, the main challenges of the federated vehicle cloud, namely, the capacity-constrained communication channel and limited resource capacity in individual vehicles, need to be addressed.

To address these challenges, we propose the UniDRM, a Unified Data and Resource Management framework for managing data and resources in a federated vehicular cloud by forming resource-based clusters, partitioning and scheduling data to the vehicular resources in the clusters.

The main contributions of the study are summarized as follows:

- We modeled a mathematical scheme inspired by divisible load scheduling schemes for data partitioning and scheduling in dynamic federated vehicular cloud. A probabilistic approach to determine network delay values is modeled and also considered in the data partitioning and scheduling scheme.
- Using the formulated mathematical scheme and the multi-criteria decision tool TOPSIS, three different data partitioning and scheduling schemes for dynamic federated vehicular cloud, namely, Cost-Aware, Time-Aware, and Reliability-Aware partitioning and scheduling, are proposed in this study.

The rest of the paper is organized as follows. Section II discusses the motivation and related work of the study. Section III provides details of the architecture of the proposed UniDRM. In section IV, the resource-based clustering algorithm for forming a federated vehicular cloud is presented. Section V presents the proposed data partitioning and scheduling schemes, and Section VI shows the performance analysis of the proposed schemes. Finally, section VII concludes the paper with a summary of the results and future work.

II. MOTIVATION AND RELATED WORK

The process of forming a federated vehicular cloud in an RoI on the road consists of organizing vehicles with sufficient resources into clusters and selecting cluster heads to serve as the cloud controller to manage the resources in the cluster.

Clustering is the grouping of vehicles in a region of interest based on predefined criteria such as speed, direction, and density of vehicles [14]. Many clustering techniques have been proposed in the literature to overcome node instability and provide an efficient message dissemination strategy in VANET.

Although clustering in VANET and vehicular cloud are all carried out in a vehicular environment, some key differences exist. In VANET clustering, the focus is to make network protocols such as routing and medium access control more scalable [15] for communication among all the vehicles in an RoI.

On the other hand, vehicle cloud clustering also attempts to optimize the selection of vehicles with sufficient resources and similar mobility characteristics as part of clusters to provide cloud services. Therefore, only nodes with sufficient resources willing to contribute to the vehicular cloud are considered part of the clusters.

Considering the differences between VANET and vehicular cloud clustering, existing VANET clustering techniques need to be improved to optimize their performance in the vehicular cloud.

In this regard, Arkain et al. [16] considered the formation of clusters in an RoI of a road to provide vehicular cloud services. A cluster head selection scheme based on a fit factor computed for each node was modeled using fuzzy logic. The study was focused on cluster head selection for vehicular
cloud and did not provide any specific algorithm or criteria for clustering vehicles.

Studies in [17] proposed a framework that organizes vehicles into non-overlapping clusters with varying numbers of communication hops to offer Data as a Service (DaaS). They also proposed a scheduling algorithm that determines when transmission links are activated and deactivated within a cluster to ensure contention-free medium access among cluster members within a cluster. The scheduling scheme maximized the throughput and minimized the delays in providing vehicular cloud services.

In [18], virtual partitions serving as logical cluster zones were marked on roads in an RoI. The vehicles in a partition were considered as cluster members to provide vehicular cloud services. A leading vehicle is then selected as the cluster head based on its expected duration in the virtual partition. The lead vehicle is assigned the responsibility of managing the resources in its virtual partition and the communication between its cluster members and that of other virtual partitions.

For stationary vehicular cloud, linear clusters of parked vehicles were formed along roads in an RoI to provide storage services in [19]. The most suitable vehicles at the front and rear ends of a linear cluster were selected as cluster heads to perform resource and data management tasks.

Although some clustering algorithms have been proposed for vehicular cloud computing, none of them except for one considers the characteristics of the resources embedded in the vehicles, such as the type of resource, the cost of the resource, and the reliability level of the resource as criteria for the formation of vehicular cloud clusters.

In the only exception, Ridhawi et al. [20] proposed a service provisioning framework that organizes vehicles with similar mobility and service types into clusters. A cluster head, which serves as a directory for vehicles and the services they offer, is selected based on link duration, connection distance, neighbors, and service availability. The clustering scheme mainly considers the scenario where individual vehicles provide cloud services independently of other vehicles (peer-to-peer provisioning model). The scenario where a group of vehicles works together as a logical unit to provide only one type of cloud service (federated vehicular cloud) was not considered.

Distinctively different from the approach considered in Ridhawi et al. [20], we propose a clustering scheme where vehicles with the same resource types (i.e., storage, computation) that meet the requirements of the vehicular cloud to be provisioned are grouped as a logical unit to provide only a particular cloud service under centralized management, i.e., a cluster head (federated vehicle cloud). This study also addresses the challenge of data partitioning and scheduling for federated vehicular cloud.

Data partitioning is very significant in vehicular cloud as it ensures that parallelism is achieved through the Single Program Multiple Data (SPMD) paradigm. In the SPMD, data for a program is partitioned into multiple independent points and executed simultaneously by coordinated computing units [21]. Examples of SPMD based applications include feature extraction and edge detection in image processing.

Several techniques have been used to partition data in distributed computing environments. Key among them is the MapReduce technique. In MapReduce, data is analyzed in parallel by decomposing tasks and specifying a map function for processing and a reduced function for aggregating tasks [22], [23].

In [24] and [25], the MapReduce model was considered for partitioning tasks among compute nodes in conventional cloud computing and vehicular cloud computing, respectively. However, the MapReduce method does not explicitly consider vehicular nodes’ frequent departure and joining of the vehicular cloud. This leads to higher waiting times when transmitting the results of partitioned tasks [26].

An optimal partitioning and allocation scheme based on the dynamic characteristics of vehicular cloud resources was proposed by [26]. In the study, the arrival and departure rates of vehicles in a parking lot were considered to model a distribution to determine the available resource capacity of vehicles. Based on the available resource capacities of vehicles in a parking lot, the optimal number of partitions for a given job and the chunk for each vehicle were determined. Although the proposed algorithm is promising, the capacity of the communication link between vehicles and the delay in data transmission due to channel access delay were not considered in the algorithm design. Moreover, the static vehicular cloud environment (parked vehicles) used in the algorithm design may lead to performance degradation if the proposed scheme is deployed in a dynamic vehicular cloud environment (moving vehicles on the road) without any enhancement.

Considering that scheduling schemes that consider both resource capacity (computation and storage) and network bandwidth capacity requirements perform better than those that consider only either of them [27], we proposed and implemented an optimal load partitioning and scheduling algorithm for dynamic vehicular cloud that takes into account the resource capacity and network data transmission capacity under different network congestion scenarios in vehicular environment.

Our proposed partitioning and scheduling scheme was inspired by divisible load partitioning and scheduling schemes such as the Periodic Write-Read-Compute (PWRC) [28] and the Divisible Load Theory (DLT) [29], [30]. Divisible load partitioning and scheduling schemes provide a tractable mathematical framework that considers both the data transfer cost and computational costs for partitioning and scheduling data to multi-connected computing systems to minimize the execution time.

Application in data mining [31], multimedia, and video processing systems [32], [33] have been developed and deployed on different distributed computing platforms such as sensor networks [34], [35] and conventional cloud computing [36] using divisible data partitioning and
scheduling schemes. However, to the best of our knowledge, no divisible load theory-based partitioning and scheduling schemes have been implemented or deployed in vehicular cloud computing.

In this study, we, therefore, propose and implement a divisible data partitioning scheduling scheme that considers the mobility and characteristics of the resources, such as the availability of the resource, the cost of the resource, and the credibility of the resources. Considering these characteristics of resources, three different task-based partitioning and scheduling schemes, namely, the expected completion time-aware, cost-aware, and reliability-aware, are proposed and implemented in this study.

III. THE UniDRM ARCHITECTURE

The UniDRM, as a unified framework, is designed to manage resources, application data, and communication network of the vehicular cloud. It can be deployed in the roadside cloud (layer-1) and the inter-vehicular cloud (layer-2) as shown in Figure 1.

The roadside cloud consists of roadside infrastructure such as traffic lights and billboards equipped with high-performance resources such as sensors, storage, and computing units organized to provide cloud services. The roadside cloud controller manages the roadside cloud resources and registers vehicles, assigns, and manages the credibility score of vehicles in the RoI on the road. In addition, the UniDRM performs tasks such as service level agreement negotiations with clients, task data partitioning, and scheduling in the roadside cloud.

The inter-vehicular cloud layer consists of mobile vehicles in an RoI on a road with sufficient resources. The UniDRM organizes resources in the inter-vehicular cloud in the form of a pool to provide cloud services to clients as a single logical entity. Figure 1 is an illustration of the UniDRM deployed in a section of the road. The vehicles in a cluster are grouped into task-specific sub-clusters to execute different tasks.

The architecture of the UniDRM shown in Figure 2, consists of a data management layer, resource management, and network management layers. In the data management layer, requests from clients are accepted after negotiating the terms of the service level agreement. The task class is then determined at this layer to select the appropriate partitioning and scheduling scheme. Finally, resource management activities such as resource clustering, resource ranking, and selection of appropriate resources for task execution are carried out at the resource management layer.

The network management layer determines the channel capacity and network access delay values that are considered in the mathematical model to determine the data fractions for vehicles (further details are provided in Section VI). The network management layer also ensures successful data transmission by the vehicular nodes.

For brevity, this study focuses on the operations performed by the UniDRM in the inter-vehicular cloud layer, specifically the formation of federated vehicular cloud clusters, the partitioning, and scheduling of application data to resources in a cluster.

IV. THE FORMATION OF FEDERATED VEHICULAR CLOUD

The process of forming a federated vehicular cloud consists of organizing vehicles into resource-based clusters and selecting the most suitable vehicle as the cluster head.

Considering that various types of resources are embedded in vehicles, different resource-based clusters can be formed in an RoI on the road. However, this study provides details on the formation of only computation as a service (CaaS) and storage as a service (StaaS) clusters.

In the formation of the CaaS and StaaS clusters, willingness and credibility values were assigned to each vehicle in the network. The willingness flag is a binary value that indicates whether a vehicle agrees to be part of the federated vehicular cloud cluster.

The credibility value is assigned and managed by the cloud controller in the roadside cloud. This ensures that only trusted nodes join the resource pool of the federated vehicular cloud. Vehicle owners who perform unethical and malicious
activities, such as abrupt termination of task execution, receive low credibility scores.

In this study, two different models, namely, the infrastructure-based model and the infrastructure-less model, are considered for the formation of vehicular cloud clusters. These models are based on the type of node that initiates the formation of vehicular cloud clusters.

In the infrastructure-based clustering model, a Road Side Unit (RSU) in an RoI initiates clustering by sending a join request message to all nodes in its transmission range. A node willing to become part of the cluster then replies with a response message containing its resource and mobility details. The response message consists of the direction of the vehicle, type of resource, the capacity of the resource, availability time, credibility value, and cost of the resource per unit time.

The RSU then organizes the vehicles that meet the resource, capacity and credibility requirements (more than a given threshold) into clusters based on the type of resource and the vehicle’s direction of travel. The node with the highest availability time is then selected as the cluster head.

In the infrastructure-less cluster formation model, the vehicle’s direction, speed, current location, resource availability time, type, credibility, and capacity of the resource node. Upon receiving the discovery message, unclustered nodes willing to become part of the federated vehicular cloud cluster then respond with an update message to establish a neighborhood association. If there are no cluster heads in the established neighborhood, nodes moving in the same direction and with credibility and capacity values more than a threshold are organized into StaaS and (or) CaaS clusters depending on the type of resources they declare. The node with the highest availability time is selected as the cluster head.

If the inquiry node has a cluster head moving in the same direction in its transmission range, the cluster head ascertains whether the inquiry node meets the cluster’s resource credibility and capacity requirements. The inquiry node that meets the requirement is sent a join-accept message to update its cluster member status. The inquiry node that does not meet the cluster requirements is sent a join-reject message from the cluster head. If the inquiry node receives only join-reject messages or does not respond to its discovery message after

FIGURE 3. Infrastructure-less federated vehicular cloud cluster formation flow diagram.
a given wait period, it rebroadcasts discovery messages until it receives a response or exists the RoI.

After the cluster formation, the cluster head is then selected, which computes the expiration time of the communication link between it and all the cluster member nodes. According to [37], the Link Expiration Time ($LET_{i,j}$) between the cluster head($i$) and any cluster member node ($j$) in the same transmission range ($r$), can be calculated considering the velocities ($v_i$, $v_j$), position coordinates ($x_i$, $y_i$) and ($x_j$, $y_j$), and the directions $\theta_i$ and $\theta_j$ ($0 \leq \theta_i, \theta_j < 2\pi$) of the vehicles $i$ and $j$ as shown in equation (1).

\[
LET_{i,j} = -\frac{(ab + cd)}{(a^2 + c^2)} + \sqrt{(a^2 + c^2)^2 - (ad-bc)^2}
\]  
(1)

where

\[
a = v_i \cos \theta_i - v_j \cos \theta_j \\
c = v_i \sin \theta_i - v_j \sin \theta_j \\
b = x_i - x_j \\
d = y_i - y_j
\]  
(2)

The computed ($LET$) and other parameters of the vehicular resource are used to rank the resources to determine the lead node for a given task, as explained in more detail in the subsequent sections of this paper.

V. DATA PARTITIONING AND SCHEDULING

The user task data partitioning and scheduling process considered in this study is shown in Figure 4. It consists of user task admission, application task classification, resource ranking, data partitions determination, and data partitions to resources mapping.

A. USER TASK ADMISSION AND TASK CLASSIFICATION

A client who wants to access the federated vehicular cloud negotiates the terms of the Service Level Agreement (SLA). If the terms are accepted, the task of the client are admitted by the cloud controller at the task admission phase. During the SLA negotiations, the cloud controller also determines the class of the task based on the user requirements.

The classification of tasks plays a vital role in determining the type of partitioning and scheduling scheme used by the cluster head to meet the user requirements. This study considers three main tasks types, time-critical tasks, low-cost tasks, and high-security tasks. Based on these task types, the partitioning and scheduling schemes, time-aware, cost-aware, and reliability-aware, are proposed.

1) TIME-AWARE PARTITIONING AND SCHEDULING (TAPS)

Applications such as disaster warning systems, disaster evacuation modeling, and simulations are considered time-critical tasks. They have a relatively short expected execution time and, therefore, must be partitioned and scheduled to minimize the total execution time of the task.

To meet this requirement, the TAPS prioritizes computational capacity over cost and the credibility of resources during data partitioning and scheduling to ensure that the minimum execution time is achieved. For example, nodes with high computational capacity that are expected to remain in the cluster until their assigned task are completed are allocated larger portions of the data for execution.

2) COST-AWARE PARTITIONING AND SCHEDULING (CAPS)

The cost of renting resources from vehicle owners to form a federated vehicular cloud varies and affects the overall cost of the service offered. Generally, vehicular resources with a high
level of credibility have higher costs. For example, resources from public transport may be considered more trustworthy and, therefore, have a higher cost than those from private third-party vehicles.

The CAPS scheme assigns larger data chunks to resources with low cost for a user task that requires a lower execution cost.

3) RELIABILITY-AWARE PARTITIONING AND SCHEDULING (RAPS)
The distributed ownership of resources in the vehicular cloud poses many security risks in executing applications such as personal health information systems.

Through the RAPS scheme, some security issues in the vehicular cloud can be addressed by ensuring that reliable vehicles are more involved in task execution. As a result, resources with higher credibility values are allocated larger percentages of data during data partitioning and scheduling.

Although the TAPS, CAPS, and RAPS schemes prioritize one criterion over the others in data partitioning and scheduling, it is desirable to consider other resource parameters to achieve optimal results for the other criteria. Therefore, the TOPSIS ranking and selection technique is used to rank resources so that high-ranked resources have optimal values for all the criteria under consideration.

B. TASK BASED RESOURCE RANKING
The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi-criteria decision analysis technique used to evaluate multiple conflicting criteria. It was proposed by [38] to select the option closest to the ideal best solution (best ideal alternative) and the farthest from the ideal negative solution (worst ideal alternative).

In the literature, TOPSIS has been considered to make decisions in cloud computing, such as the selection of cloud service providers [39] due to its simplicity and computational efficiency. It consists of the following steps.

Step 1: Establish a decision matrix for the criteria and alternatives.
Step 2: Normalize the decision matrix.
Step 3: Construct weighted normalized decision matrix.
Step 4: Identify ideal positive and negative solutions.
Step 5: Compute the separation measures for each alternative.
Step 6: Calculate the relative closeness to the ideal solution.
Step 7: Rank the alternatives according to their closeness coefficients.

In Step 1, the performance or decision matrix \( X_{ij} \) is constructed based on the number of resources in a cluster \( N \) and the attributes of the resources \( M \). The impact of the attributes is also determined and optimized: benefit criteria are maximized while the cost criteria are minimized.

The attributes of resources considered are the link expiration time between nodes and cluster head (LET), the unit cost of the resource, the credibility value of the resource and the reciprocal computational capacity of the resources.

\[
X_{ij} = \begin{bmatrix}
X_{11} & X_{12} & X_{13} & X_{14} \\
X_{21} & X_{22} & X_{23} & X_{24} \\
X_{31} & X_{32} & X_{33} & X_{34} \\
\cdots & \cdots & \cdots & \cdots \\
X_{n1} & X_{n2} & X_{n3} & X_{n4}
\end{bmatrix} \tag{3}
\]

The decision matrix is then normalized in step 2 using equation (4)

\[
\tilde{X}_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} X_{ij}^2}}, \quad i,j = 1, 2, \ldots, n \tag{4}
\]

In step 3, each attribute is assigned a weight value \( W_j = [w_1, w_2, w_2, w_4] \) based on the partitioning and scheduling scheme under consideration. The weights of the attributes for the TAPS, CAPS and RAPS schemes are listed in Table 1.

The sum of the weights of the attributes for each partitioning and scheduling scheme is one (\( \sum_{i=1}^{n} W_j = 1 \)).

| Scheme  | LET (min) | Cost (min) | CredVal (max) | Inv. comp cap (min) |
|---------|-----------|------------|---------------|----------------------|
| TAPS    | 0.05      | 0.05       | 0.25          | 0.65                 |
| CAPS    | 0.10      | 0.60       | 0.20          | 0.10                 |
| RAPS    | 0.10      | 0.20       | 0.60          | 0.10                 |

It is worth noting that the values of the assigned weights are empirical values that serve as case study input to the algorithm. A system designer or the cloud controller may define a new Data Partitioning and Scheduling Scheme (DPSS) by specifying different weight values for the parameters.

The weighted normalized matrix is then computed according to equation (5).

\[
\tilde{X}_{ij} = X_{ij} \times W_j
\]

\[
\tilde{X}_{ij} = \begin{bmatrix}
\tilde{X}_{11w_1} & \tilde{X}_{12w_2} & \tilde{X}_{13w_3} & \tilde{X}_{14w_4} \\
\tilde{X}_{21w_1} & \tilde{X}_{22w_2} & \tilde{X}_{23w_3} & \tilde{X}_{24w_4} \\
\tilde{X}_{31w_1} & \tilde{X}_{32w_2} & \tilde{X}_{33w_3} & \tilde{X}_{34w_4} \\
\cdots & \cdots & \cdots & \cdots \\
\tilde{X}_{nw_1} & \tilde{X}_{nw_2} & \tilde{X}_{nw_3} & \tilde{X}_{nw_4}
\end{bmatrix} \tag{6}
\]

In determining the ideal best solution in step 4, the largest weighted attribute value is selected if the relevant attribute values are maximized, while the smallest weighted attribute value is selected if the relevant attribute is minimized as depicted in equation 7.

\[
\tilde{X}_j^+ = \max_{i=1}^{M} \tilde{X}_{ij} \quad \text{or} \quad \tilde{X}_j^- = \min_{i=1}^{M} \tilde{X}_{ij} \tag{7}
\]

For the ideal worst solution, the smallest weighted attribute value is selected if the relevant attributes are maximized and...
the largest value is selected if the relevant evaluation attribute is minimized according to the equations 8.

\[
\hat{X}_j^- = \min_{i=1}^N \hat{X}_{ij} \quad \text{or} \quad \hat{X}_j^- = \max_{i=1}^M \hat{X}_{ij} \quad (8)
\]

The Euclidean distance between each alternative resources and the ideal best solution is computed according to in equation (9) and the distance between resources and the ideal worst solution is also computed according to equation (10).

\[
S_i^+ = \left[ \sum_{j=1}^n (\hat{X}_{ij} - X_j^+)^2 \right]^{0.5} \quad (9)
\]

\[
S_i^- = \left[ \sum_{j=1}^n (\hat{X}_{ij} - X_j^-)^2 \right]^{0.5} \quad (10)
\]

The TOPSIS performance score (closeness coefficient) for each resource was then computed considering equation (11).

\[
P_i = \frac{S_i^-}{S_i^+ + S_i^-} \quad (11)
\]

Based on the performance score, the resources are ranked in descending order and the highest-ranked node is then selected as the lead node.

**C. DATA PARTITION DETERMINATION AND DATA MAPPING**

The data partitioning and scheduling scheme was modeled as a one-round scheme in this study. The data of an application is partitioned and scheduled for resources in a time frame.

The data to be executed by the federated vehicular cloud are sent from the cluster head to the lead node. The lead node, acting as the load originator for the cluster, partitions the data into \( \alpha_0, \alpha_1, \ldots, \alpha_n \) units for vehicles \( V_0, V_1 \ldots, V_n \).

For each data chunk \( \alpha_i \), it takes \( Z\alpha_i \) time units to be sent from the lead node to the cluster members. The value of \( Z \) is the inverse data transmission rate \( (T) \) of the communication channel \( (Z_i = \frac{1}{T}) \).

Similarly, for a vehicle \( V_i \) with reciprocal computation capacity \( \omega_i \), the time required to execute a data chunk \( \alpha_i \) is \( \omega_i\alpha_i \). The reciprocal computation capacity is the time it takes a vehicle node to execute a unit of data of a given task.

The following conditions and assumptions were considered in modeling the data partition and scheduling scheme.

1) The processing units of the vehicles are not equipped with front-end processors; therefore, sending and computing the data partitions do not occur simultaneously. The vehicles can either process, send, or receive data partitions at a time.

2) Since the communication among nodes is carried out in a single channel, the transmission of data chunks from the lead node is carried out sequentially. Similarly, the other nodes also transfer their processed data chunks sequentially to the lead node.

3) The order of reception of data partition from the lead node is based on the TOPSIS rank of the node.

Higher-ranked nodes receive their data partitions before the low-ranked nodes. However, the transmission of processed data chunks is a reversed order of data reception (FILO).

4) The size of each task data chunk before processing is the same as after processing, and the sum of all data chunks assigned to vehicular nodes is one (normalized) that is, \( \sum_{i=0}^{n} \alpha_i = 1 \).

5) The lead node vehicle starts processing its partition after distributing chunks of task data to all the other nodes while the cluster members start processing their partitions after receiving their data chunks.

6) For all vehicles other than the lead node, the reception, processing, and submission time of the data is within the execution time of the preceding vehicle.

\[
EX(V_1) > EX(V_2) > \ldots > EX(V_{n-1}) > EX(V_n) \quad (12)
\]

Given these assumptions, the timing diagram in Figure 5 was produced for the partitioning and scheduling scheme.

From the timing diagram, the processing time is defined as the time taken to perform operations on data \( (\alpha\omega) \) while the execution time is the sum of the processing time and communication time \( (Z\alpha_i + \alpha\omega_i + Z\alpha_i) \). Based on the assumptions and features of the timing diagram, we determined the partitions for each node using an analytical model consisting of the following system equations.

The times required by the lead node \( EX(V_0) \) and cluster members \( EX(V_i) \) to complete the execution of their assigned task are given by equation (13) and equation (14).

\[
EX(V_0) = \sum_{i=0}^{n} Z\alpha_i + \alpha_0\omega_0 + \sum_{i=0}^{n} Z\alpha_i \quad (13)
\]

\[
EX(V_i) = Z\alpha_i + \alpha_i\omega_i + Z\alpha_i \quad \text{for } i = 1, 2 \ldots n \quad (14)
\]

From the timing diagram, the processing time of the lead vehicle \( V_0 \) and the last vehicle \( V_n \) are the same and can be expressed as follows:

\[
\alpha_n\omega_n = \alpha_0\omega_0 \quad \frac{\omega_n}{\omega_0} = \alpha_n = \alpha_0 \quad (15)
\]

Thus, the data chunk \( \alpha_n \) can be expressed in terms of the data chunk \( \alpha_0 \) in equation (16) as.

\[
\alpha_n = m\alpha_0 \quad (16)
\]

The execution time for all nodes except the lead node can be expressed in terms of the processing time and communication times of their successor nodes as follows.

\[
\alpha_i\omega_i = \alpha_{i+1}(Z\alpha_{i+1} + \omega_{i+1}) \quad \text{for } i = 1, 2 \ldots n - 1 \quad (17)
\]

For each node except the lead node and the last node, its partition can be expressed in terms of that of its successor
\[ \alpha_i = \alpha_{i+1} + 1 \left( \frac{2Z_{i+1} + \omega_{i+1}}{\omega_i} \right) \quad (18) \]

let \[ k_{i+1} = 2Z_{i+1} + \omega_{i+1} \quad (19) \]

Equation (18) can now be written as:
\[ \alpha_i = \alpha_{i+1} + \frac{k_{i+1}}{\omega_i} \quad (20) \]

From equation (20) the partition for the \((n-1)\)th node can be written as
\[ \alpha_{n-1} = \alpha_n \cdot \frac{k_n}{\omega_{n-1}} \quad (21) \]

Based on equations (16) and (20) the partitions for all vehicles can be expressed in terms of \(\alpha_0\) through the recursive equations:
\[
\begin{align*}
\alpha_n &= m\alpha_0 \\
\alpha_{n-1} &= m\alpha_0 \cdot \frac{k_n}{\omega_{n-1}} \\
\alpha_{n-2} &= m\alpha_0 \cdot \frac{k_n}{\omega_{n-1}} \cdot \frac{k_n}{\omega_{n-2}} \\
\alpha_{n-3} &= m\alpha_0 \cdot \frac{k_n}{\omega_{n-1}} \cdot \frac{k_{n-1}}{\omega_{n-2}} \cdot \frac{k_n}{\omega_{n-3}} \\
& \vdots \\
\alpha_1 &= m\alpha_0 \cdot \frac{k_n}{\omega_{n-1}} \cdot \frac{k_{n-1}}{\omega_{n-2}} \cdot \frac{k_{n-2}}{\omega_{n-3}} \cdot \cdots \cdot \frac{k_2}{\omega_1} 
\end{align*}
\quad (22) \]

Given that all data chunks sum up to one (equation (23)) and each data chunk can be expressed in terms of \(\alpha_0\), a closed-form equation can be derived to compute the data partition \(\alpha_0\) of the lead node as shown in equation (25).
\[
\sum_{s=0}^{n} \alpha_s = 1 \quad (23)
\]

\[ \alpha_0 + m\alpha_0 \cdot \sum_{j=1}^{n-1} \prod_{i=1}^{j} \frac{k_{n-i+1}}{\omega_{n-i}} + m\alpha_0 = 1 \quad (24) \]

\[ \alpha_0 = \frac{1}{m \left( \sum_{j=1}^{n-1} \prod_{i=1}^{j} \frac{k_{n-i+1}}{\omega_{n-i}} \right) + 1} \quad (25) \]

The partitions of all other vehicles are then obtained by substituting \(\alpha_0\) into the set of equations in (22).

After the partitions of the vehicles are determined, they are mapped to the ordered resources by distributing them sequentially by the lead node.

**VI. SYSTEM SIMULATION**

The simulation of the UniDRM was performed using the Omnet++ [40] simulator and the VANET simulation framework Veins [41]. The parameters of vehicular nodes in the Veins simulator were modified to include the willingness flag, credibility value, cost, type, and resource capacity. Realistic mobility and traffic scenarios were also modeled using the mobility simulator SUMO [42] and map data from OpenStreet [43].

To analyze the performance of the proposed UniDRM framework, we conducted simulations for all phases of the UniDRM framework: resource management, data management, and communication channel management phases.

In the resource management phase, resource-based clustering and task-based resource ranking were implemented. Although the cluster formation schemes proposed for the UniDRM cover the formation of StaaS and CaaS clusters, as described in Section IV, for simplicity, we considered the infrastructure-less CaaS clusters for implementation of the data management phase of the UniDRM framework. In the data management phase, the data partitioning and scheduling schemes are implemented. For the data partitioning and scheduling simulation, we considered a merge sort application as it reflects one of the key assumptions of the data partitioning and scheduling scheme; the size of data

---

**FIGURE 5.** Task partitioning and scheduling timing diagram.
fraction before execution and after execution is the same. The application consists of a file of a large dataset partitioned and distributed among the nodes of the cluster. The partitioned data chunks were then sorted by the cluster members and the results were sent to the lead node, which merged all the results.

The communication channel management phase of the simulation focused on modeling the channel access delay and incorporating it into the data partitioning and scheduling schemes. Three different network congestion scenarios were considered in the simulation, namely, low network traffic congestion (LCS), medium network traffic congestion (MCS), and high network traffic congestion (HCS).

Several simulation runs were carried out according to the system parameters presented in Table 2, and the average results of each scenario were analyzed and presented.

### TABLE 2. Simulation parameters.

| Parameter                        | Value                  |
|----------------------------------|------------------------|
| Simulation time                  | 1200s                  |
| Simulation area                  | 870m x 650m            |
| PHY protocol                     | IEEE-802.11p           |
| MAC protocol                     | MAC 1609.4             |
| Transmission range               | 500m                   |
| Data transfer rate               | 27Mbps                 |
| Max. Number of vehicles          | 100                    |
| Average cluster size             | 20                     |
| Vehicle velocity range           | 20 m/s - 50 m/s        |
| Network congestion scenario      | LCS, MCS & HCS         |

### A. ANALYSIS OF SIMULATION RESULTS

From the preliminary simulation results, it was observed that the execution time of the partitioning and scheduling scheme varied under network traffic congestion when the same resources (with unchanged parameters) were used for the task execution.

In the ideal network traffic scenario where only data partitions were transmitted during the scheduling phase of the simulation, the expected execution time was achieved as there were no channel congestion and channel access delays during data chunks transmission.

However, in the instances where different types of communication from both cluster and non-cluster members took place during the data scheduling phase of the simulation (in LCS, MCS, and HCS), the execution times were higher than that of the ideal network traffic scenario. This was due to the high channel access delay caused by channel congestion during the transmission of the data chunks.

As shown in Figure 6a and Figure 6b, the execution times for the ideal network traffic scenario and the HCS are 0.587s and 8.883s, respectively (the execution times in this study are reported as a fraction of the total data size). It was also observed that the channel access delay yielded unforeseen internal idle times in the scheduling scheme.

Although an ideal network traffic scenario is desirable, it is not realistic in a vehicular environment because of the large number of vehicles and devices that communicate in a vehicular environment at any given time. The transmission of data chunks in a vehicular environment experiences channel access delays, leading to an increase in the expected execution time of tasks.

To address this challenge and reduce the internal idle times, we reformulated the communication parameter (Z) in the analytical model in Section V to account for the channel access delay. The expression for Z now becomes

\[
\lambda_i + \frac{1}{\tau}
\]

Where \( \lambda_i \) is the communication channel access delay.

Using the modified communication parameter Z in the partitioning and scheduling scheme produced a better execution time when subjected to the same high network traffic congestion scenario, the execution time decreased from 8.883s in Figure 6b to 0.754s in Figure 6d which corresponds to a percentage reduction in execution time of approximately 14.6%.

Because the value of the communication channel access delay is dependent on the network condition, such as the number of vehicles communicating in the same transmission range, the actual values cannot be determined before the computation of data chunks for vehicles. Therefore, a predictive approach based on the results from different simulation scenarios is considered in this study to determine the delay values at different time instances of the simulation.

### B. THE DISTRIBUTION OF ACCESS DELAY FACTOR (\( \lambda \))

Similar to the studies in [26], the results of our simulations show that non-deterministic features such as data transmission failure and channel access delay in a vehicular environment can be considered as monotonically increasing or monotonically decreasing functions.

The channel access delay values increased monotonically in the initial simulation stage as the vehicles entered the simulation. This is due to the exchange of messages to establish neighborhood associations.

After all the vehicles entered the simulation, the channel access delay values appeared to be constant and then decreased monotonically to the end of the simulation as the vehicles left the simulation network.

This behavior of the channel access delay during the simulation can be represented as a sigmoid function. Therefore, to predict the channel access delay values, we considered the Boltzmann sigmoid distribution shown in Figure 7 to model the characteristics of our simulation network scenario. The values of the Boltzmann sigmoidal function values were obtained using the following equations

\[
f(t) = d_2 + \frac{(d_2 - d_1)}{1 + e^{-\frac{t - \text{band}}{\frac{4}{\tau}}}} \quad (26)
\]

\[
f'(t) = (d_1 + d_2) - f(x) \quad (27)
\]

where Equation (26) and equation (27) are the sigmoidal monotonically increasing and sigmoidal monotonically...
The network traffic congestion scenarios, the HCS, MCS, and LCS considered in the simulation were generated with the following values for the Boltzmann sigmoid distribution. The minimum delay value \(d_1\), and slope, \(y\), for all the congestion scenarios were 0.002\(s\) and 6. The maximum delay value, \(d_2\), for LCS, MCS, and HCS are 0.035\(s\), 0.053\(s\), and 0.074\(s\) respectively. The distribution curves of the access delay for the three congestion scenarios obtained with the generated values are shown in Figure 8a.

Although the Boltzmann sigmoidal model is considered in this study, due to the modularity of our system, other models that reflect the characteristics of vehicular environment channel access can be considered for predicting the channel access delay values.

Subsequent simulations were carried out with the modified communication variable \(Z\), which included predicted channel access delay values for different network congestion scenarios.

C. PERFORMANCE EVALUATION

For the performance evaluation, we considered fully homogeneous and fully heterogeneous cluster scenarios. The fully homogeneous scenario was modeled on the assumption that all resources in a cluster have the same parameter values.
In the performance analysis of the fully homogeneous scenario, the metrics, execution time, speedup, and parallel execution efficiency were considered.

The execution time is the total time required by the lead node to distribute data chunks and aggregate the results from all cluster members after processing their data chunks. Figure 8b shows the execution time for a varying number of vehicular processors. In the figure, three performance curves with different network traffic congestion scenarios are presented. It can be deduced that the execution time decreases as the number of vehicle processors involved in task execution increases. Furthermore, the curves become steeper and converge as more processing units are involved in task execution. This implies that the execution times do not improve significantly beyond a certain number of processing nodes. For this simulation instance, after approximately 18 vehicular nodes in the LCS, 17 nodes in MCS, and 15 vehicular nodes in the HCS, there is not much improvement in the execution time.

The speedup and efficiency are two essential metrics used to determine the performance of all parallel processing systems. Speedup is the ratio of the execution time of a single vehicular processor to the number of vehicular processes involved in the partitioning and scheduling scheme. The parallel efficiency is the ratio between the speedup and the number of vehicular processors considered.

Figure 8c and Figure 8d show the speedup and efficiency values obtained with an increasing number of vehicular nodes considering the low congested network scenario (LCS), medium congested network scenarios (MCS), and the high congested network scenarios (HCS). It can be observed that the speedup values generally increase up to a certain point and then flatten out, while the parallel efficiency curves also decrease until they flatten out. This is expected because the addition of more vehicles does not contribute to the execution time. However, the speedup and efficiency values for the LCS were always better than those of the MCS and HCS. This is because LSC has the lowest delay in data transmission.
The speedup and parallel efficiency values can be used to determine the optimal number of vehicular nodes to be included in the data partitioning and scheduling for a given task. For example, to achieve a speedup value of 2.0 required 4 (four) vehicular nodes in the HCS and 2 (two) vehicular nodes in the LCS and MCS, respectively.

As a fully homogeneous scenario may not be realistic in a vehicular environment, we also considered a fully heterogeneous scenario where all the properties of the vehicular resources vary to implement the task-based data partitioning and scheduling schemes (TAPS, CAPS, and RAPS).

The parameters of the resources in the cluster considered for task execution are shown in Table 3. This includes the cost per unit time usage, credibility value, and inverse computational capacity per unit data. In addition, the ranks of the resources based on the type of partitioning scheme are also shown in the table.

TABLE 3. Vehicular nodes and their attribute values.

| Veh. | Cost ($) | Cred. | Inv.Compcap | TAPS Rank | CAPS Rank | RAPS Rank |
|------|----------|-------|-------------|-----------|-----------|-----------|
| V0   | 0.504    | 0.708 | 2.740       | 5         | 6         | 6         |
| V1   | 0.639    | 0.742 | 2.320       | 6         | 5         | 5         |
| V2   | 0.850    | 0.783 | 2.210       | 10        | 4         | 7         |
| V3   | 0.792    | 0.855 | 2.270       | 7         | 3         | 2         |
| V4   | 0.456    | 0.624 | 3.720       | 4         | 8         | 8         |
| V5   | 0.240    | 0.165 | 3.720       | 2         | 10        | 10        |
| V6   | 0.420    | 0.942 | 3.020       | 1         | 7         | 1         |
| V7   | 0.931    | 0.866 | 1.940       | 9         | 1         | 4         |
| V8   | 0.876    | 0.870 | 2.160       | 8         | 2         | 3         |
| V9   | 0.357    | 0.382 | 3.570       | 3         | 9         | 9         |

The performance metrics considered are the execution time, the total cost of all resource usage, average reliability score, and resource efficiency.

The cost of each resource usage is calculated as the product of the execution time and the cost per unit time usage of the resource. The total cost for all resources used in the schedule is calculated as follows,

$$\text{Totalcost} = \sum_{i=1}^{n} Z_i \alpha_i + w_0 \alpha_0 + \sum_{j=1}^{n} Z_j \alpha_j \ast \text{Cost}_0 + (w_1 \alpha_j + 2z_j \alpha_j) \ast \text{Cost}_j$$ \hspace{1cm} (28)

where $V_0$ denotes the lead node and $j = 1, 2, 3 \ldots n$ denote all nodes other than the lead node.

The reliability score for each resource usage is also computed as the product of the data partition assigned to a node ($\alpha_j$) and the credibility value of the node ($\text{Cred}_j$). The average reliability score (AvReScore) is then computed in (29) as the total reliability score of all resources divided by the number of resources used for task execution ($n$).

$$\text{AvReScore} = \frac{\sum_{j=1}^{n} \alpha_j \ast \text{Cred}_j}{n}$$ \hspace{1cm} (29)

The results of the TAPS, CAPS, and RAPS schemes showing the percentages of data partitions for each node are shown in Tables 4, 5 and 6. The vehicular nodes in the table are ordered based on their TOPSIS scores, where the first node serves as the lead node, and the other nodes are the cluster members. It can be observed that the percentage of partitions of cluster member nodes decreases as the ranks of resources decrease. This ensures that highly ranked resources are assigned larger percentages of data.

The execution time, total cost, average reliability score, and parallel efficiency for the different task-based partitioning and scheduling schemes shown in the tables are compared using the graphs in Figures 9, 10, 11, and 12.

From the execution time graph in Figure 9, the values for the curves decrease and converge as more nodes are considered for task execution. The rate of decrease depends on the processor-to-channel capacity ratio (i.e., the ratio between the reciprocal processor speed ($\omega$) and the inverse data transfer rate of the channel $Z$). Thus, when nodes involved in the task execution have a larger processor-channel capacity ratio, the execution time decreases steadily.
TABLE 4. Data partitions for the time-aware partitioning and scheduling (TAPS) scheme.

| Res./No. Veh. | V7 | V8 | V3 | V2 | V1 | V0 | V6 | V4 | V9 | V5 |
|---------------|----|----|----|----|----|----|----|----|----|----|
| Partitions    | 0.527 | 0.332 | 0.223 | 0.158 | 0.119 | 0.093 | 0.077 | 0.062 | 0.051 |
| Execution time | 1.326 | 0.976 | 0.937 | 0.857 | 0.810 | 0.778 | 0.759 | 0.742 | 0.730 |
| Total cost    | 2.296 | 2.601 | 2.743 | 2.759 | 2.754 | 2.741 | 2.752 | 2.745 | 2.727 |
| Average reliability score | 0.343 | 0.288 | 0.212 | 0.167 | 0.137 | 0.118 | 0.102 | 0.089 | 0.079 |
| Efficiency | 0.732 | 0.601 | 0.517 | 0.453 | 0.399 | 0.356 | 0.320 | 0.290 | 0.266 |

TABLE 5. Data partitions for the cost-aware partitioning and scheduling (CAPS) scheme.

| Res./No. Veh. | V6 | V5 | V9 | V4 | V0 | V1 | V3 | V8 | V7 | V2 |
|---------------|----|----|----|----|----|----|----|----|----|----|
| Partitions    | 0.581 | 0.356 | 0.254 | 0.174 | 0.118 | 0.083 | 0.059 | 0.041 | 0.030 |
| Execution time (s) | 1.954 | 1.492 | 1.252 | 1.066 | 0.939 | 0.859 | 0.805 | 0.766 | 0.743 |
| Total cost ($) | 1.290 | 1.440 | 1.620 | 1.708 | 1.800 | 1.906 | 1.991 | 2.051 | 2.089 |
| Average reliability score | 0.297 | 0.169 | 0.128 | 0.106 | 0.092 | 0.082 | 0.074 | 0.067 | 0.061 |
| Efficiency | 0.773 | 0.674 | 0.603 | 0.567 | 0.536 | 0.502 | 0.469 | 0.438 | 0.407 |

TABLE 6. Data partitions for the reliability-aware partitioning and scheduling (RAPS) scheme.

| Res./No. Veh. | V6 | V3 | V8 | V7 | V1 | V0 | V2 | V4 | V9 | V5 |
|---------------|----|----|----|----|----|----|----|----|----|----|
| Partitions    | 0.429 | 0.242 | 0.148 | 0.103 | 0.077 | 0.055 | 0.045 | 0.037 | 0.030 |
| Execution time | 1.662 | 1.221 | 1.001 | 0.897 | 0.838 | 0.790 | 0.767 | 0.749 | 0.735 |
| Total cost ($) | 2.015 | 2.314 | 2.456 | 2.456 | 2.441 | 2.485 | 2.490 | 2.478 | 2.458 |
| Average reliability score | 0.446 | 0.294 | 0.219 | 0.171 | 0.140 | 0.119 | 0.103 | 0.090 | 0.079 |
| Efficiency | 0.710 | 0.824 | 0.754 | 0.673 | 0.600 | 0.546 | 0.492 | 0.448 | 0.411 |

However, nodes with a lower processor-channel capacity ratio negatively affect the execution time. It may either decrease insignificantly, remain unchanged, or increase. Nodes with a processor-channel capacity ratio less than 2 (two) lead to an increase in the execution time as their data transfer times exceed their data processing times. Such nodes can be excluded from the task execution to achieve optimal execution time. It can be observed that the TAPS performs better than the RAPS and CAPS as expected.

For the total cost of resource utilization shown in Figure 10, the values generally increase when more nodes are included in the task execution. As expected, the CAPS scheme has relatively lower cost compared to TAPS and RAPS as resources that are assigned large data percentage are relatively cheaper. For the average reliability score shown in Figure 11, the performance curves decrease when more resources are involved in the task execution since the resources with higher credibility scores receive a larger share of data. Therefore, the average reliability scores decrease when more nodes with lower credibility values are considered. It was observed that the average reliability score of the RAPS was relatively higher than those of the TAPS and CAPS. However, the scores of
TAPS are closer to those of RAPS since the weight assigned to the credibility attribute of resources in the TOPSIS ranking for TAPS is higher than that of CAPS.

The parallel efficiency of the task-based partitioning and scheduling schemes are compared in Figure 12. In general, the efficiency values of TAPS are better than those of RAPS and CAPS as the execution time is considered in the computation of the efficiency values being the lowest in the TAPS.

In all task-based scheduling schemes, the appropriate number of vehicles for task execution can be selected based on the execution time, total resource cost, average reliability score, and efficiency values required by the client.

VII. CONCLUSION

In this study, we proposed the UniDRM, a unified data and resource management framework for partitioning and scheduling divisible loads in a federated vehicular cloud. The framework organizes vehicles in a region of interest with sufficient resources into clusters based on the type of resource available in the vehicle and their mobility parameters to collaborate as a single unit to execute a task.

To execute a computationally intensive task, the UniDRM framework partitions data into non-dependent fragments and distributes them to selected vehicles in the cluster for parallel execution.

Mathematical models were used to determine partitions for each vehicle. Three different partitioning and scheduling schemes: expected completion time-aware, cost-aware, and reliability-aware, were proposed to obtain optimal execution times, total costs, and average reliability values for the task execution. Realistic simulations of the modeled schemes were performed by considering the dynamic characteristics of the vehicle resources.

Although the UniDRM is proposed for federated vehicular cloud, some of the applied techniques, such as resource ranking for data partitioning, can be used for other distributed computing platforms such as mobile cloud computing.

Future works will examine the effect of clustering issues such as the cluster head change rate and cluster member exit rate on the proposed scheduling schemes. Data partitioning and scheduling in multi-clusters, where two or more clusters execute a single application with multiple data instances in a vehicular environment, will also be investigated.

REFERENCES

[1] D. Ni, “Traffic sensing technologies,” in Traffic Flow Theory, Amsterdam, The Netherlands: Elsevier, 2016, pp. 3–17, doi: 10.1016/b978-0-12-804134-5.00001-5.
[2] IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, Standard ETSI EN 302 663 V1.3.1, doi: 10.1109/ieeestd.2010.5514475.
[3] IEEE Standard for Wireless Access in Vehicular Environments (WAVE)—Multi-Channel Operation Corrigendum 1: Miscellaneous Corrections, Standard 1609.4-2016/Cor1-2019, 2019, doi: 10.1109/ieeestd.2014.6998918.
[4] Y. Fangchun, W. Shangguang, L. Jinglin, L. Zhihan, and S. Qibo, “An overview of internet of vehicles,” China Commun., vol. 11, no. 10, pp. 1–15, Oct. 2014.
[5] J. G. Andrews, S. Buzzi, V. Choi, S. V. Hanly, A. Lozano, and J. C. Zhang, “What will 5G be?” IEEE J. Sel. Areas Commun., vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
[6] W. M. Danquah and D. T. Altilar, “Vehicular volunteer computing (VVC): A novel paradigm for the volunteerism of vehicular resources; opportunities and challenges,” in Proc. Int. Conf. Comput. Electron. Commun. Eng. (iCCECE), Aug. 2019, pp. 208–213.
[7] C. S. Yeo, M. D. de Assunç ao, J. Yu, A. Sulistio, S. Venugopal, M. Placek, and R. Buyya, “Utility computing and global grids,” 2006, arXiv:cs/0605056.
[8] S. Olariu, T. Hristov, and G. Yan, The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds. Hoboken, NJ, USA: Wiley, 2013, ch. 19, pp. 645–700, doi: 10.1002/9781118511305.
[9] M. Eltoweissy, S. Olariu, and M. Younis, “Towards autonomous vehicular clouds,” in Ad Hoc Networks, Berlin, Germany: Springer, 2010, pp. 1–16, doi: 10.1007/978-3-642-17994-5_1.
[10] M. Abuelea and S. Olariu, “Taking VANET to the clouds,” in Proc. 8th Int. Conf. Adv. Mobile Comput. Multimedia (MoMM), 2010, pp. 6–13, doi: 10.1145/1971519.1971522.
[11] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, “A survey on vehicular cloud computing,” J. Netw. Comput. Appl., vol. 40, pp. 325–344, Apr. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804513001793
W. M. Danquah and D. T. Altılar, “Vehicular cloud resource management, issues and challenges: A survey,” *IEEE Access*, vol. 8, pp. 180587–180607, 2020, doi: 10.1109/access.2020.3027637.

W. M. Danquah and D. T. Altılar, “VCloud: A security framework for VANET,” in *Mobile and Wireless Technology* (Lecture Notes in Electrical Engineering), Berlin, Germany: Springer, 2015, pp. 1–13, doi: 10.1007/978-3-662-47697-1_1.

R. S. Bali, N. Kumar, and J. J. P. C. Rodrigues, “Clustering in vehicular ad hoc networks: Taxonomy, challenges and solutions,” *Veh. Commun.*, vol. 1, no. 3, pp. 134–152, Mar. 2014, doi: 10.1016/j.vcom.2014.05.004.

K. Abboud and W. Zhuang, “Stochastic modeling of single-hop cluster stability in vehicular ad hoc networks,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 226–240, Jan. 2016.

H. Nov-Arkian, R. E. Atiani, and A. Pourkhalili, “A cluster-based vehicular cloud architecture with learning-based resource management,” in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 162–167.

M. Azizian, S. Cherkauri, and A. Hafid, “An optimized flow allocation in vehicular cloud,” *IEEE Access*, vol. 4, pp. 6766–6779, 2016.

J. F. Bravo-Torres, E. F. Ordonez-Morales, M. Lopez-Nores, Y. Blanco-Fernandez, and J. J. Pazo-Arias, “Virtualization in VANETs to support the vehicular cloud—Experiments with the network as a service model,” in *Proc. 3rd Int. Conf. Future Gener. Commun. Technol. (FGCT)*, Aug. 2014, pp. 1–6.

F. Dressler, P. Handle, and C. Sommer, “Towards a vehicular cloud—using parked vehicles as a temporary network and storage infrastructure,” in *Proc. ACM Int. Workshop Wireless Mobile Technol. Smart Cities* (WoWMoC), 2018, pp. 11–18, doi: 10.1145/3166126.3166371.

I. Al Ridhawi, M. Aloqaily, B. Kantarci, Y. Jararweh, and H. T. Mouftah, “A continuous diversified vehicular cloud service availability framework for smart cities,” *Comput. Netw.*, vol. 145, pp. 207–218, Nov. 2018, doi: 10.1016/j.comnet.2018.08.023.

A. Votorovic, M. V. Tomasˇevic´, and V. M. Milutinovic, “Manual parallelization versus state-of-the-art parallelization techniques: The spec cpu2006 as a case study,” in *Advances in Computers*, vol. 92, A. Hurson, Ed. Amsterdam, The Netherlands: Elsevier, 2014, pp. 203–251. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124202320000052.

J. Dean and S. Ghemawat, “MapReduce,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

Y. Chen, S. Alsapgha, and R. Katz, “Interactive analytical processing in big data systems,” *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, Aug. 2012, doi: 10.14778/2267502.2267519.

Z. Xiao and Y. Xiao, “Achieving accountable MapReduce in cloud computing,” *Future Gener. Comput. Syst.*, vol. 30, pp. 1–13, Jan. 2014, doi: 10.1016/j.future.2013.07.001.

T. Adhikary, A. K. Das, M. A. Razzaque, A. Almogren, M. Alrubaiyen, and M. M. Hassan, “Quality of service aware reliable task scheduling in vehicular cloud computing,” *Mobile Netw. Appl.*, vol. 23, no. 3, pp. 482–493, 2018, doi: 10.1007/s11036-017-0924-0.

T. Kim, H. Min, E. Choi, and J. Jung, “Optimal job partitioning and allocation for vehicular cloud computing,” *Future Gener. Comput. Syst.*, vol. 108, pp. 82–96, Jul. 2020, doi: 10.1016/j.future.2020.02.007.

W. Lin, C. Liang, J. Z. Wang, and R. Buyya, “Bandwidth-aware divisible task scheduling for cloud computing,” *Softw., Pract. Exper.*, vol. 44, no. 2, pp. 168–174, Feb. 2010.

D. T. Altılar, “Data partitioning and scheduling for parallel digital video processing,” Ph.D. dissertation, Dept. Comput. Sci., Queen Mary, Univ. London, London, U.K., 2002.

Y. C. Cheng and T. G. Robertazzi, “Distributed computation for a tree network with communication delays,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 26, no. 3, pp. 511–516, May 1990, doi: 10.1109/6.1016129.

R. Agrawal and H. V. Jagadish, “Partitioning techniques for large-grained parallelism,” in *Proc. 7th Annu. Int. Phoenix Conf. Comput. Commun.*, Conf., Mar. 1998, pp. 31–38, doi: 10.1109/pcc.1998.100399.

M. Masud, J. Gao, L. Khan, H. Han, and B. M. Thuraisingham, “Classification and novel class detection in concept-drifting data streams under time constraints,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.

D. T. Altılar and Y. Paker, “Optimal scheduling algorithms for communication constrained parallel processing,” in *Proc. Eur. Conf. Parallel Process. Paderborn*, Germany: Springer, 2002, pp. 197–206.