Open Access

Open Journal of Internet of Things (OJIOT)
Volume 7, Issue 1, 2021
http://www.ronpub.com/ojiot
ISSN 2364-7108

A Mobile and Web Platform for Crowdsourcing OBD-II Vehicle Data

Giuseppe Loseto, Filippo Gramegna, Agnese Pinto, Michele Ruta, Floriano Scioscia

Polytechnic University of Bari, Department of Electrical and Information Engineering, via E. Orabona 4, I-70125, Bari, Italy
{giuseppe.loseto, filippo.gramegna, agnese.pinto, michele.ruta, floriano.scioscia}@poliba.it

Abstract

On-Board Diagnostics 2 (OBD-II) protocol allows monitoring vehicle status parameters. Analyzing them is highly useful for Intelligent Transportation Systems (ITS) research, applications and services. Unfortunately, large-scale OBD datasets are not publicly available due to the effort of producing them as well as due to competitiveness in the automotive sector. This paper proposes a framework to enable a worldwide crowdsourcing approach to the generation of OBD-II data, similarly to OpenStreetMap (OSM) for cartography. The proposal comprises: (i) an extension of the GPX data format for route logging, augmented with OBD-II parameters; (ii) a fork of an open source Android OBD-II data logger to store and upload route traces, and (iii) a Web platform extending the OSM codebase to support storage, search and editing of traces with embedded OBD data. A full platform prototype has been developed and early scalability tests have been carried out in various workloads to assess the sustainability of the proposal.

Type of Paper and Keywords

Regular research paper: *Intelligent Transportation Systems (ITS), Crowdsourcing, OBD-II*

1 Introduction and Motivation

Advanced Driver Assistance Systems (ADAS) include automated emergency brake, lane keep assist, automatic parking, and adaptive cruise control. They implement levels 0-2 in the well-known taxonomy of driving automation defined by Society of Automotive Engineers (SAE) J3016 standard [23]. Levels 3-5 classify the realm of Autonomous Driving (AD). Between 2015 and 2020 the global market for ADAS has grown from 7.6 to 17.6 billion dollars, with an expected growth to 31.9 B$ in 2023 [24]. A large majority of cars is expected to equip autonomy levels 0-2 before levels 3 and higher enter the market [24]. This is due to the extremely complex technological challenges facing full AD.

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2021)* in conjunction with the VLDB 2021 conference in Copenhagen, Denmark. The proceedings of VLIoT@VLDB 2021 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

Current AD research and development combine the availability of (i) high-resolution maps, (ii) high-bandwidth, low-latency vehicle-to-everything (V2X) communications [3], (iii) one or more types of high-throughput environmental perception sensors, including cameras, radar, lidar and ultrasonic sensors, and (iv) an on-board real-time data processing platform to analyze data gathered and make autonomous decisions. State-of-the-art machine learning technologies are really effective in semantic-based image segmentation [9], but training models for such complex and variable settings as autonomous driving requires huge amounts of data. Furthermore, external perceptions collected through sensors must be fused with internal vehicle data to be concretely supportive in those cases. This implies the need for very large vehicle status datasets. On-Board Diagnostics, version 2 (OBD-II) [14] is the standard protocol for real-time access to vehicle status parameters and diagnostic trouble codes. While the physical interface connectors and signaling protocols are common across the automotive industry, each vehicle model grants access to a different subset of parameters, possibly including further non-standard types.

The consequence of the above state of affairs is that datasets collected and published by industry players suffer from three main limitations: (i) they mostly focus on cameras and lidar; (ii) even if they include vehicle status data, they refer to a very limited set of parameters and vehicle models; (iii) whilst their scale is adequate for testing and comparing semantic segmentation approaches on images or 3D scenes, their usefulness for real AD tasks has not been widely validated yet. The availability of larger and more diverse datasets, in particular including OBD-II parameters, is not limited purely by the effort required to produce them. The highly competitive nature of the automotive sector is also a problematic factor. Companies in competition with each other are unlikely to release critical information publicly or to reach broad sharing agreements. This curbs the data collection process and consequently limits the potential for open research, particularly by academia. Beyond ADAS/AD, large-scale collections of OBD-II vehicle data would support the creation and improvement of several other kinds of Intelligent Transportation Systems (ITS) services with positive societal impact. They include vehicle maintenance and remote assistance, fleet management (both for business entities and for public services such as police and fire departments, or ambulances), performance control for pollution reduction, traffic and road network management, and platooning.

Peer production, *i.e.*, the collaboration of volunteer communities, is one of the most significant Internet-mediated organizational innovations [6]. It leverages a common governance model and a set of technological tools to harness the diverse motivations of a large number of decentralized agents for contributing to a shared pool of information resources. Groundbreaking worldwide endeavors such as *Wikipedia* (`https://www.wikipedia.org/`) for encyclopedic knowledge and *OpenStreetMap* (OSM, `https://www.openstreetmap.org/`) for cartographic data have demonstrated the effectiveness of this paradigm –a.k.a. *crowdsourcing*– for building information repositories at a scale hardly attainable by a single centralized entity, without sacrificing average quality. The basic idea of this paper is to apply the peer production model to the collection of OBD-II data in real trips, without any limitation of route, vehicle model, or collected parameters. The goal is to allow any driver with an OBD-II scan tool and a smartphone to participate in data collection and sharing. For this reason, a crowdsourcing framework is proposed, based on the following main contributions:

- an open data format for embedding OBD-II parameter readings into GNSS (Global Navigation Satellite System) points composing routes;
- an Android mobile client for logging route traces augmented with OBD-II data and upload them;
- a Web platform for collecting, searching and editing route traces.

Each element of the framework has been produced by extending open source software and open standard technologies. In particular, the Web platform is based on OSM, exploiting its rich functionality and proven robustness.

Devised framework and tools allow any user driving a car to contribute to a worldwide repository of timestamped and georeferenced vehicle status data. This approach may enable the collection of an OBD-II data repository in fully realistic driving conditions with unprecedented scale and variety. Such an asset has the potential to foster a large number of innovative ITS research projects, applications and services, not unlike what has been happening to location-based services leveraging the OSM cartography. In particular, georeferencing OBD data can facilitate pattern analysis and machine learning model training, based not only on internal vehicle conditions, but also on correlations with road characteristics, as provided by the underlying OSM infrastructure.

Since embedding OBD-II parameters can make GNSS route traces significantly more data-intensive, scalability is a significant concern. To demonstrate the feasibility and sustainability of the proposal, an early experimental campaign has been carried out on a full prototype instance of the platform to appraise computational resource utilization under increasingly complex workloads.

The remainder of the paper is organized as follows. Section 2 recalls preliminaries on the OBD-II protocol, then Section 3 discusses related work. The proposed framework is described in detail in Section 4, followed by experiments in Section 5. Conclusion closes the manuscript.

HEADERS			DATA							CRC CHECKSUM
H1 TYPE	H2 TARGET	H3 SOURCE	D1 MODE	D2 PID	D3	D4	D5	D6	D7	

Figure 1: OBD-II generic frame structure

2 On-Board Diagnostics: Basics

OBD is a real time request-response diagnosis protocol, able to monitor the vehicle status parameters (endowed with unique PIDs, Parameter IDentification) of on-board subsystems and diagnose the presence of any errors/malfunctions (identified by DTCs, Diagnostic Trouble Codes). DTCs values are stored in car Electronic Control Units (ECUs) and can be later retrieved by maintenance technicians using proper hardware and software kits. Traditional processing and control platforms embedded in vehicles adopted *federated architectures*, where several ECUs monitor and coordinate internal components, communicating over one or more Controller Area Network (CAN) buses in a loosely coupled fashion. Due to requirements of increased functional complexity and flexibility, automotive designs are evolving toward *integrated architectures*, a concept borrowed from the Integrated Modular Avionics (IMA) paradigm for aircraft. Unlike federated architectures, where each subsystem performs a dedicated function, integrated architectures include generic computing platforms, which can be used for several kinds of functions and, in some cases, run multiple tasks concurrently [16]. With this approach, fewer subsystems are needed, but the complexity of hardware and software integration and coordination increases with strict safety guarantees.

International standards require new vehicles support the version 2 of the protocol (OBD-II) [7] and be equipped with an OBD-compliant interface to provide direct access to data in the vehicle network. The majority of current automobiles have the OBD-II port installed under the dashboard. The OBD-II specification only comprises the Physical Signal Layer (PSL) and the OBD-II Data Communication Layer (DCL) w.r.t. the ISO/OSI model. In particular, PSL outlines hardware characteristics, standard connector (SAE J1962 [21]) conformation and exploited protocols. DCL defines the structure of diagnosis messages exchanged with the ECU, as described in SAE J1979 specification [22].

Figure 1 shows the details of exchanged generic request/reply datagrams:

- header bytes H1, H2 and H3 denote priority or message type (request/response), destination and sender addresses, respectively;
- the first data byte D1, called *mode byte*, represents the modality to access OBD information. The standard supports 10 modes for diagnosis data. In particular, *mode 1* is used to obtain current diagnostic information;
- the second data byte includes the so-called PID: a value indicating what data is required. The PID also fills the second byte in the corresponding reply packet coming from the vehicle;
- the remaining data bytes, when used, are reserved for further specification about possible required information. In a reply message, they are the actual value returned from the vehicle ECU;
- the last byte is for message error control.

In latest years, the access to vehicle data has been granted also to the general public of car enthusiasts by means of OBD-II *scan tools*. They are cheap electronic devices bridging the OBD-II port with wired (RS-232, USB) or wireless (Bluetooth, IEEE 802.11) computer communication interfaces. A majority of scan tools is based on the ELM327 (`https://www.elmelectronics.com/ic/elm327/`) microcontroller, which acts as a bridge between low-level OBD transmission protocols and a serial interface. This interface allows access through a terminal, which can be used directly on PCs or mobile devices (smartphone, tablet) connected to the scan tool via USB, Bluetooth or Wi-Fi. The microcontroller provides a set of request/reply messages for both reading and writing. To read vehicle status parameters, a specific command must be sent via the terminal indicating the required PID. The car ECU will reply with raw data that requires appropriate decoding in order to obtain meaningful final information.

3 Related Work

Available AD datasets such as *ApolloScape* [12], *nuScenes* [8], the *Waymo Open Dataset* [26] and *SemanticKITTI* [5] provide camera frames and lidar point clouds for semantic-based image/3D segmentation and object recognition tasks. ApolloScape and SemanticKITTI also contain GPS (Global Positioning System) and IMU (Inertial Measurement Unit, *i.e.*, accelerometer and gyroscope) companion data. Unfortunately, none of these datasets incorporates OBD parameters. The recent *Audi Autonomous Driving Dataset* (A2D2) [11] includes OBD data (specifically, steering angle, brake, throttle, odometry, speed, pitch, and roll) in addition to camera and lidar, but the corpus has been collected on a single vehicle model, an *Audi Q7 e-tron*.

Table 1: Available OBD-II datasets

Source	Year	Parameters	Samples	Research goal
UTDrive [2]	2007	9	390537	Driver behavior modeling
Abut *et al.* [1]	2008	15	16608	Driver behavior modeling
Kwak *et al.* [13]	2016	51	94380	Driver identification
Stocker *et al.* [25]	2017	10	119638	Vehicle lifecycle services
Barreto [4]	2018	27	57519	Driving style classification
Ruta *et al.* [20]	2019	14	24961	Road surface, traffic and driving style classification

To the best of our knowledge, the only publicly available OBD-II datasets are summarized in Table 1. Earlier projects [2, 1] spent a huge effort gathering multi-modal sensor data including OBD-II, in-car audio and video, pedal pressure and others. The availability of commodity scan tools and apps facilitated OBD-II data collection significantly in later projects. Anyway, the size of each dataset is relatively small, since all but one –*i.e.*, [25]– were created for focused rather than open-ended research.

A possible solution to overcome the data sharing problem in the automotive industry has been proposed in *AutoMat* (`https://automat-project.eu/`), a European research project created with the goal of establishing a cross-border Vehicle Big Data Marketplace. The idea is based on a brand-independent Common Vehicle Information Model (CVIM) [17], granting access to aggregated vehicle data for cross-sector service providers. CVIM aims at standardizing the gathering process as well as the formats of both data and descriptive metadata. This makes information access uniform and independent of the producer, as well as datasets comparable across the industry.

Based on similar requirements for large-scale interoperability, crowdsourcing requires open data formats to enable collaborative generation and maintenance of repositories. The *GPS eXchange Format* (GPX) [10] defines an open XML Schema, which is a *de facto* standard for exchanging GPS traces among software applications. Ease of usage and expandability are its key benefits. The GPX data model is described in subsection 4.2; during GPX trace upload, OSM translates GPX *points* and *traces* respectively to *nodes* and *ways* in its own data model, described in subsection 4.3.

Several ITS applications exploit the extraction of OBD vehicle data through a smartphone. The survey in [27] has found main use cases include the estimation of fuel consumption, the analysis of driving behavior, the inference of information about the driver's state and the prediction of engine failures.

Critical services and applications in the automotive sector require strong guarantees of data integrity and traceability. For this reason pilot projects are experimenting with data exchange approaches based on blockchain technologies. BMW has proposed *VerifyCar*, a decentralized application (*dapp*) running on the *VeChainThor* (`https://www.vechain.org/`) blockchain. Basically, VerifyCar works like a real-time vehicle registration document, storing data about the mileage, the history of accidents, maintenance, repairs and other useful information. In this way, each vehicle is always equipped with an updated history, which cannot be tampered with and can be consulted by any authorized party at any time. One of the major goals is to prevent frauds towards insurance companies or buyers in the car market, which currently have significant associated costs. Analogously, Rahman *et al.* [18] have proposed a blockchain platform for OBD data sharing and transactions among user groups in Internet of Vehicles (IoV) infrastructures. The blockchain stores references to the actual data, which are saved off-chain on InterPlanetary File System (IPFS, `https://ipfs.io/`) distributed storage and processed by dapps exploiting an edge-cloud computing architecture. A preliminary analysis of transaction throughput and latency has been carried out in a simulated setting, but the first proof-of-concept dapps have not been completed yet.

4 Crowdsourcing Framework

This section outlines the proposal for a mobile and Web platform aiming to collect and share vehicle data acquired through the OBD-II protocol. Starting from research in [19], the system monitors vehicles by using electronic devices (smartphones, smart OBD-II scan tools) collecting heterogeneous driving information (*e.g.*, speed, acceleration, fuel consumption, driving style, status of roads and traffic). According to the high-level architecture depicted in Figure 2, cars are equipped with an OBD-compliant scan tool to expose raw data through a Wi-Fi or Bluetooth connection. Particularly, *PLX Kiwi 2+* (`http://www.plxkiwi.com/kiwiwifi/hardware.html`) wireless adapter has been exploited in our tests. When turned on, it

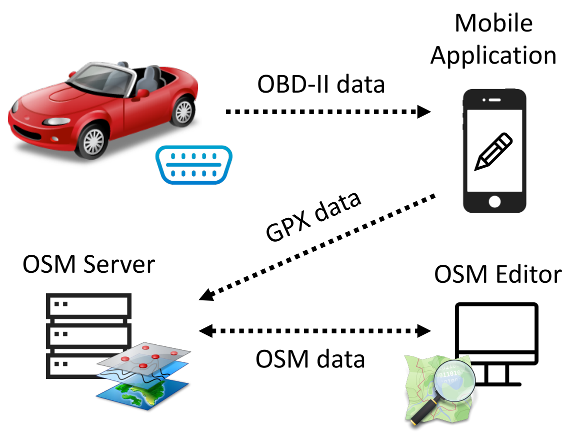


Figure 2: Framework architecture

exposes a Bluetooth interface allowing an application to communicate with the OBD adapter. The application itself provides additional information acquired from embedded smartphone micro-devices, such as GPS and accelerometer. Data are gathered within short observation intervals and annotated according to the GPX [10] lightweight data format: as detailed later on Section 4.2, a specific extension has been proposed to model OBD-II and environmental parameters. Retrieved information are stored in an open cartographic database, based on the OpenStreetMap project to guarantee robust data integrity. OSM also provides functionalities aiming to create crowdsourced maps by importing GPX data.

Users can set new types of map tags without restriction (albeit following OSM community guidelines is highly recommended) to accommodate previously unforeseen cartography usage. Users can also query or modify stored information exploiting the OpenStreetMap editor: this makes any authorized contributor capable of enriching maps with additional elements.

Technologies and software modules grounding the framework are discussed in detail in the following subsections.

4.1 A Mobile Application for OBD Data Collection

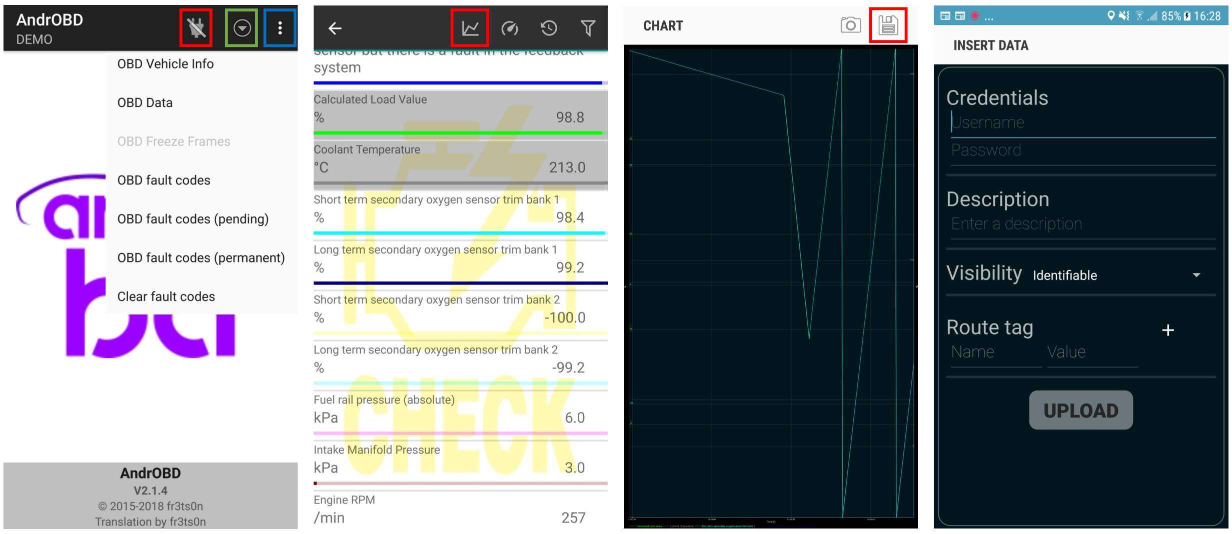
In order to record GPS data and collect associated OBD parameters, we propose to extend the *AndrOBD* (`https://github.com/fr3ts0n/AndrOBD/wiki`) open source Android application. It is designed to interact with the car diagnostic system via an ELM237 scan tool supporting Bluetooth, Wi-Fi or USB communication interfaces. Its main features concern the visualization of DTCs stored in the vehicle ECU and the real-time monitoring of OBD-II parameters, with the possibility to export the recorded data in Comma-Separated Values (CSV) files. AndrOBD provides a simple approach to extend those functionalities by developing specific plug-ins. Each plug-in is a standalone application that communicates with the main one through the Android *intent exchange* mechanism.

To achieve the objectives discussed in Section 4, the original application has been extended with the following features:

- possibility to choose a variable logging interval from 1 to 5 seconds;
- tracking of GPS coordinates, in order to draw a route on a map;
- compliance with the GPX file format (see Section 4.2), to export logged data. To manipulate the XML-based structure of GPX files it has been used the *dom4j* (`https://dom4j.github.io/`) Java library;
- ability to automatically upload recorded routes on the Web platform after a data gathering session.

Figure 3 reports on main activities of the application. By clicking on the *plug-in* button icon (in the red box in Figure 3a) the user can select the reference scan tool from a list of nearby Bluetooth devices. The scan tool must be physically connected to the OBD-II port and the vehicle instrument panel must be switched on[1]. Once a connection is established, the user gets current diagnosis by clicking on the *menu* button icon (green box in Figure 3a) and selecting the *OBD Data* item from the drop-down menu. As shown in Figure 3b, all OBD parameters supported by the vehicle are listed. Each of them has a short description along with interpreted value (repeatedly updated) and measurement unit (if any). Hence, by selecting a parameter with a long click, one can press on the *display graph* button (red box in Figure 3b) and data acquisition starts (Figure 3c). The application begins recording data –sampled as specified in the settings section– until the user clicks on *save* button (red box in Figure 3c); and then the recorded information is exported as GPX file, see Figure 3d. Finally, to upload the GPX trace the user is required to enter his/her credentials to the Web platform, also inserting a description for registered route, visibility criteria and any associated tag. Once the form has been filled in, the *Upload* button starts the file transfer. A HTTP POST request is adopted for that, in compliance with the *OSM API*. A confirmation message is returned

[1] AndrOBD also supports a demo mode, which does not require a scan tool connected. To start a demo session the user should disable the Bluetooth interface before running the application: this operational mode allows simulating a vehicle that supports all recognized OBD PIDs, generating random values for each of them.



AndrOBD
DEMO

OBD Vehicle Info
OBD Data
OBD Freeze Frames
OBD fault codes
OBD fault codes (pending)
OBD fault codes (permanent)
Clear fault codes

AndrOBD
V2.1.4
© 2015-2018 fr3ts0n
Translation by fr3ts0n

sensor but there is a fault in the feedback system
Calculated Load Value % 98.8
Coolant Temperature °C 213.0
Short term secondary oxygen sensor trim bank 1 % 98.4
Long term secondary oxygen sensor trim bank 1 % 99.2
Short term secondary oxygen sensor trim bank 2 % -100.0
Long term secondary oxygen sensor trim bank 2 % -99.2
Fuel rail pressure (absolute) kPa 6.0
Intake Manifold Pressure kPa 3.0
Engine RPM /min 257

CHART

INSERT DATA
Credentials
Username
Password
Description
Enter a description
Visibility Identifiable
Route tag
Name Value
UPLOAD

(a) App homepage **(b) OBD data selection** **(c) OBD data chart** **(d) Upload to server**

Figure 3: AndrOBD main activities

in case of success, otherwise the server reply contains a status code for the given error.

User preferences can be changed through the *ellipsis* menu (blue box in Figure 3a) by selecting the *Options* item. Particularly, the settings affecting the generation of GPX files are:

- *OBD options* → *Data to display*: enable/disable the selectable OBD parameters in the list of Figure 3b;
- *GPX options*: enable the automatic upload of GPX traces and set data recording interval.

4.2 GPX Format Extension

The need for uploading georeferenced data combined with OBD parameters (collected step-by-step during a trip) and the capability to associate given tags for information annotation lead to an extension of standard GPX format. OSM currently accepts loading .gpx files only, but the GPX XML Schema [10] supports defining extensions through additional schemas. The version considered here as reference is 1.1, the most recently available one.

A GPX file includes a `<trk>` element for each trace it contains. A trace is a sequence of segments, each represented by a `<trkseg>` tag. A segment, in turn, is a sequence of points marked by `<trkpt>` components. Given a point, GPS data loggers typically include coordinates (latitude and longitude of `<trkpt>`) and children elements for elevation and timestamp. The following extension has been proposed for the GPX XML Schema:

- a path (`<trk>`) possibly includes tags (`<tag>`), each defined by a [key,value] pair, all enclosed within an XML container (`<tags>`);
- a GPS point (`<trkpt>`) possibly includes tags (`<tag>`) as before;
- a GPS point (`<trkpt>`) possibly contains OBD parameters (`<obd_param>`), each defined by [ID,description,value] and enclosed in a container XML (`<obd_params>`);
- each container possibly has other extensions.

4.3 Web Platform

The Web application proposed here grounds on OpenStreetMap, a project and a platform aiming to create and share open and freely editable cartographic data on a worldwide basis. OSM relies on the paradigm of crowdsourcing: a collective effort scheme where users record routes via GPS devices to update and enrich a map. Manual editing is also possible, through several available editors. Basically, an OSM map is a data model containing the following main elements:

- *nodes*: *i.e.*, cartographic points with geographic coordinates;
- *ways*: *i.e.*, sequences of nodes, forming a polyline or polygon;
- *relations*: *i.e.*, groups of nodes, ways and other relations to which specific properties can be assigned;

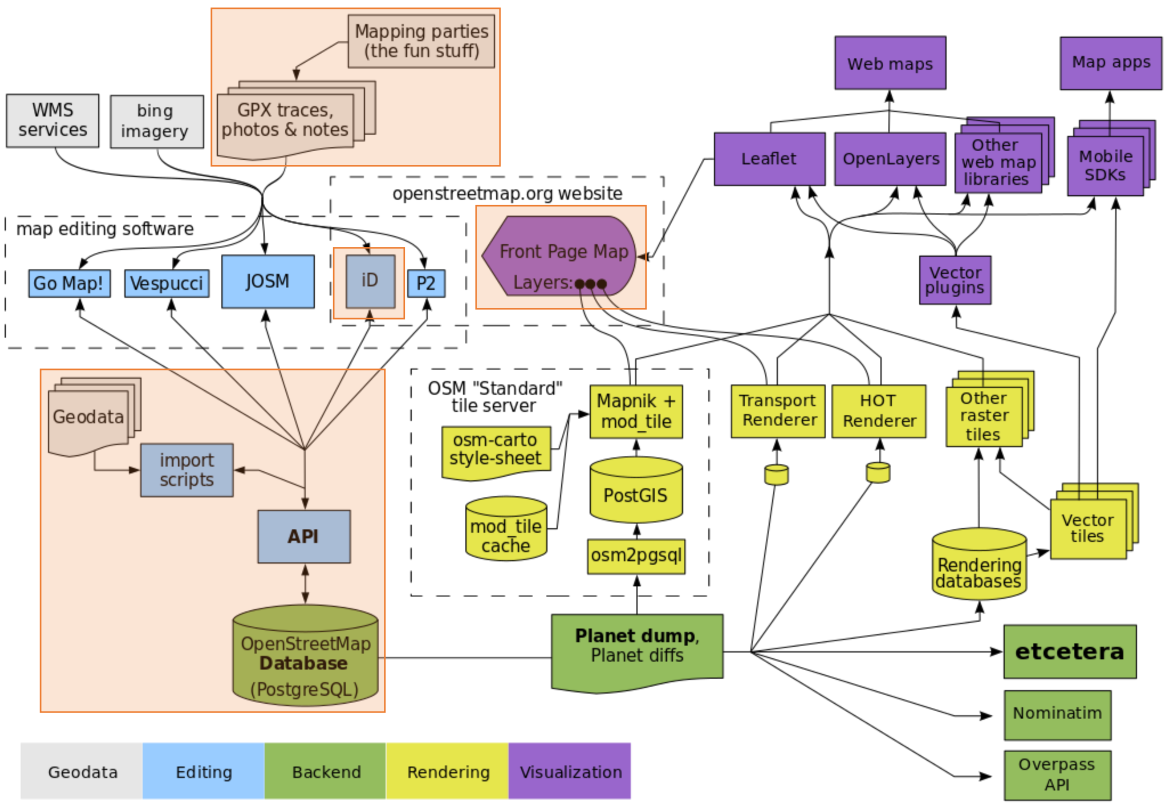


Figure 4: OSM components architecture

- *tags*: *i.e.*, [key,value] labels applicable to nodes, ways or relations.

Any change to OSM data must happen through a *changeset* (group of modifications). Specifically, to edit a map it is needed to: (i) open a new changeset and define its metadata, (ii) make the desired changes, (iii) close the changeset. Once closed it is immutable, except for metadata.

Figure 4 shows the overall OSM technological stack[2]: boxes with a light orange background enclose the functional components modified for our framework. The architecture consists of a large number of services, distributed on different logical layers, related to operations as rendering, editing, storage and backup. Main elements are:

- *Leaflet*. A JavaScript library for managing the *slippy map i.e.*, the sliding map component embedded in Web pages. Based on the displayed area, it is responsible to select the tiles (raster images of limited size) to be placed in the grid for map rendering.

- *Overpass API*. Used to implement query functionalities on the slippy map, such as viewing the list of nearby points of interest.

- *Mapnik*. A toolkit exploited to render individual tiles and store them in the tile server in case of map updates.

- *Nominatim*. A tool to implement geocoding and reverse geocoding operations, used to search or display an address on the slippy map.

- *Database*. It contains all the information related to maps and users. OSM is based on the PostgreSQL (`https://www.postgresql.org/`) open source relational DBMS.

- *OSM API*. It provides a RESTful interface for read and write operations on the *Database*, using simple URLs to access objects and standard HTTP response codes.

- *iD*. A modern Web-based editor for OSM with an attractive, easy to use and practical user interface. It is developed in JavaScript and interacts with the OSM *Database* via the *OSM API*.

[2] `https://wiki.openstreetmap.org/wiki/File:OSM_Components.svg`

The main focus of the solution proposed here is on managing GPX traces. A user should be able to load a route, show it and make changes without modifying the entire map, which is simply used as cartographic reference. By default, OSM loads the points belonging to a GPX trace in an immutable data structure called *tracepoint*, in order to generate a path displayed as overlay on the map. To enable the editing of a GPX trace, the upload procedure has been modified so that the route points are no longer saved in a fixed data structure, but stored as *nodes* belonging to a *way*; where each *node* also contains information of recorded OBD parameters. For this purpose the OSM *Database* schema has been extended with: (i) a new `gpx_id` attribute added to a `way` entry for connecting each uploaded GPX file to the reference `way` object, (ii) a new `aux_info` table containing additional immutable data related to each node, such as the values for OBD parameters. Furthermore, since it is now possible to modify a trace, the platform allows to dynamically recreate the GPX file starting from the last changes applied to a route.

Figure 5a shows the homepage of the Web portal in a smartphone browser viewport; the platform inherits support of laptop and desktop computers from OSM as well. The *History* button refers to a standard OSM feature listing all the changesets within a given area currently displayed on the map. Changeset details, such as *tags* and modified objects (*ways* and/or *nodes*), can be shown by selecting a specific item. The *Paths* button allows the access to the new paths section that, as shown in Figure 5a, resembles the changesets list view. In this case, uploaded routes related to the selected geographic area are listed. For each item, description, *way* identifier and a bounding box surrounding the modified map zone are shown. On the top of the sidebar there are two other commands:

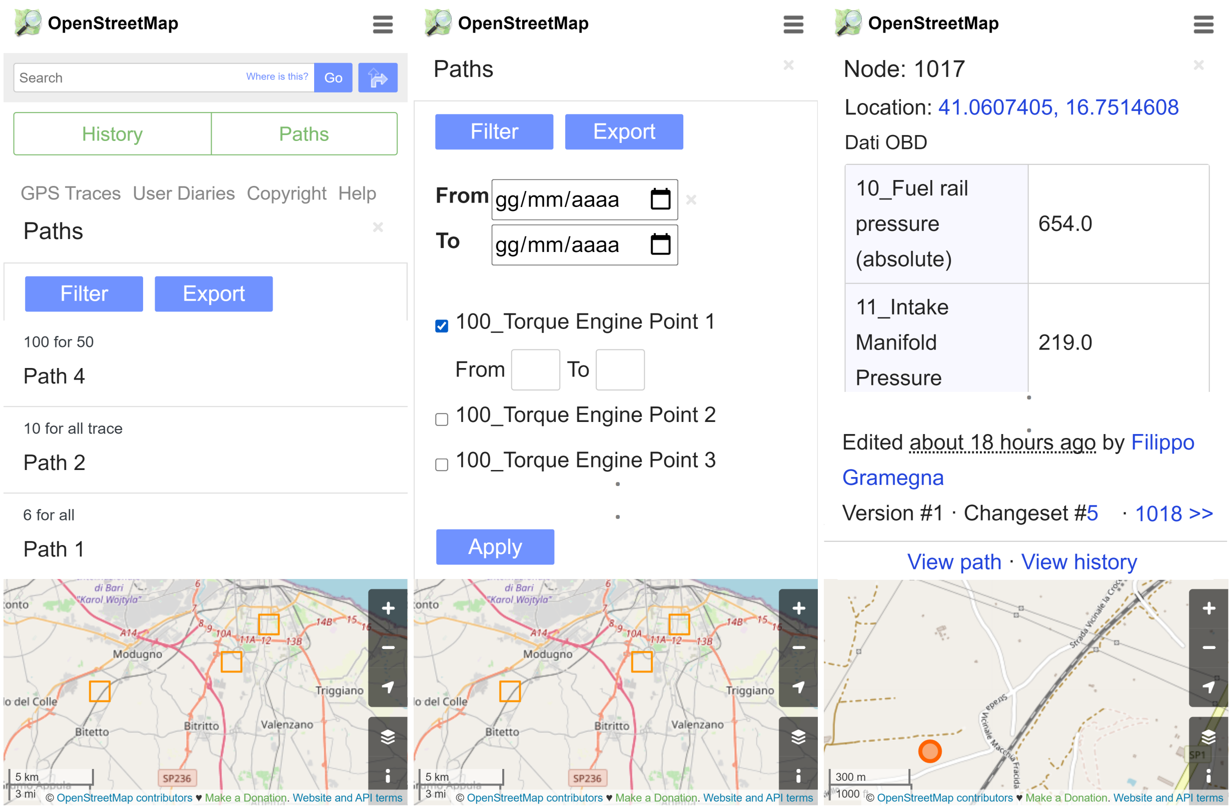
1. *Filter*, used to select the returned paths in the area of interest, based on upload date of GPX and given OBD parameters. In particular, as depicted in Figure 5b, the user indicates the OBD parameters (among all those available in the displayed paths) and sets constraints (minimum/maximum) on the assumed value for each of them. To get the filtered data, the user must press the *Apply* button.

2. *Export*, allowing to extract all the paths belonging to a zone or returned by a search (if filters have been applied). The output of the export operation consists of a single GPX file including OBD data (see Section 4.2).

By selecting a path it is possible to view related main information: identifier, description, number and list of *nodes*, *tags*, recorded OBD parameters, date and author of changes, version and identifier of the associated changeset. Furthermore, for each path it could be visualized the history of modifications. A node in a path is represented as hyperlink: by clicking on it, the information reported in Figure 5c is shown: identifier, GPS coordinates, OBD parameters, date and author of changes, version (#1 means that the node has never been modified) and the identifier of the associated changeset. From the node detail view the user can also navigate to the *way* the node belongs to (*View path*) or to the *node* history data (*View history*).

From the home page the user can also be redirected to the GPX traces page by clicking on the *GPS Traces* link. As shown in Figure 6a, there are all routes loaded on the server. For each of them, the following information is reported along with a thumbnail: GPX file name, number of points, upload date, route visibility (identifiable, public, traceable or private), description and owner. By clicking on a specific path, the user can access detailed data (Figure 6b): a segment-wise track animation, a time series plot of an OBD parameter (a dropdown list allows choosing the parameter to be displayed), a button for editing the route metadata (visibility and description) and a button to delete the trace. Finally, the associated GPX file can be downloaded via the *download* link.

Logged users can modify a path through the *edit* link opening the *iD* editor, see Figure 7. As per Figure 4, the *iD* editor does not have direct access to the OSM *Database*, but it gets all needed information via *OSM API*. Usually the editor allows modifying a map, whilst we aim to edit a single route. To this end, it has been necessary to modify the OSM API by adding new endpoints in order to: (i) load all data related to a *way* bounded with a GPX file, (ii) update involved tables after route modifications have been confirmed. As shown in Figure 7, the route to be edited is displayed on the satellite map as directed line made up of a series of points. Each of them is a node to be moved freely on the map or deleted. By clicking on the path with the right mouse button, the available transformations are shown within a context menu; notice that some operations provided by the *iD* editor have been hidden in case of currently unsupported features. The sidebar includes functions to view, add or modify the *tags* associated with a way or its nodes. By pressing the *Save* command, one could add a short comment summarizing changes and finally the *Upload* button starts transfer. The GPS trace section (Figure 6a) also allows loading new routes. The user is redirected to the page in Figure 6c, where s/he selects the GPX to upload and enters metadata, a short description, a list of tags ([key,value] pairs) and visibility information. The *Create Trace* button saves entered data and starts the import of trace points in the OSM *Database*. In more detail, the system creates a new



Paths

100 for 50
Path 4

10 for all trace
Path 2

6 for all
Path 1

From gg/mm/aaaa
To gg/mm/aaaa

[✓] 100_Torque Engine Point 1
From To
[ ] 100_Torque Engine Point 2
[ ] 100_Torque Engine Point 3

Node: 1017
Location: 41.0607405, 16.7514608
Dati OBD

10_Fuel rail pressure (absolute)	654.0
11_Intake Manifold Pressure	219.0

Edited about 18 hours ago by Filippo Gramegna
Version #1 · Changeset #5 · 1018 >>

(a) Web platform homepage (b) List of paths within the selected area (c) Details of a single node

Figure 5: Web platform Paths section

entry in the *gpx_files* table containing the route metadata. The upload procedure is assigned to a *job worker*, which manages the queue of background tasks and proceeds the input of data points defined in the GPX file. The following steps succeed:

1. create a new changeset;
2. parse the GPX and, for each point, create a new *node* element, with the related data stored in the *aux_info* table;
3. link extracted *nodes* to a new *way* object, whose *gpx_id* focuses path data in *gpx_files* table;
4. update route information (number of points, first point coordinates, route bounding box) stored in *gpx_files*;
5. close the changeset.

Finally, the system generates an e-mail containing an operational summary. In case of errors, this resume contains a detailed description of the issue.

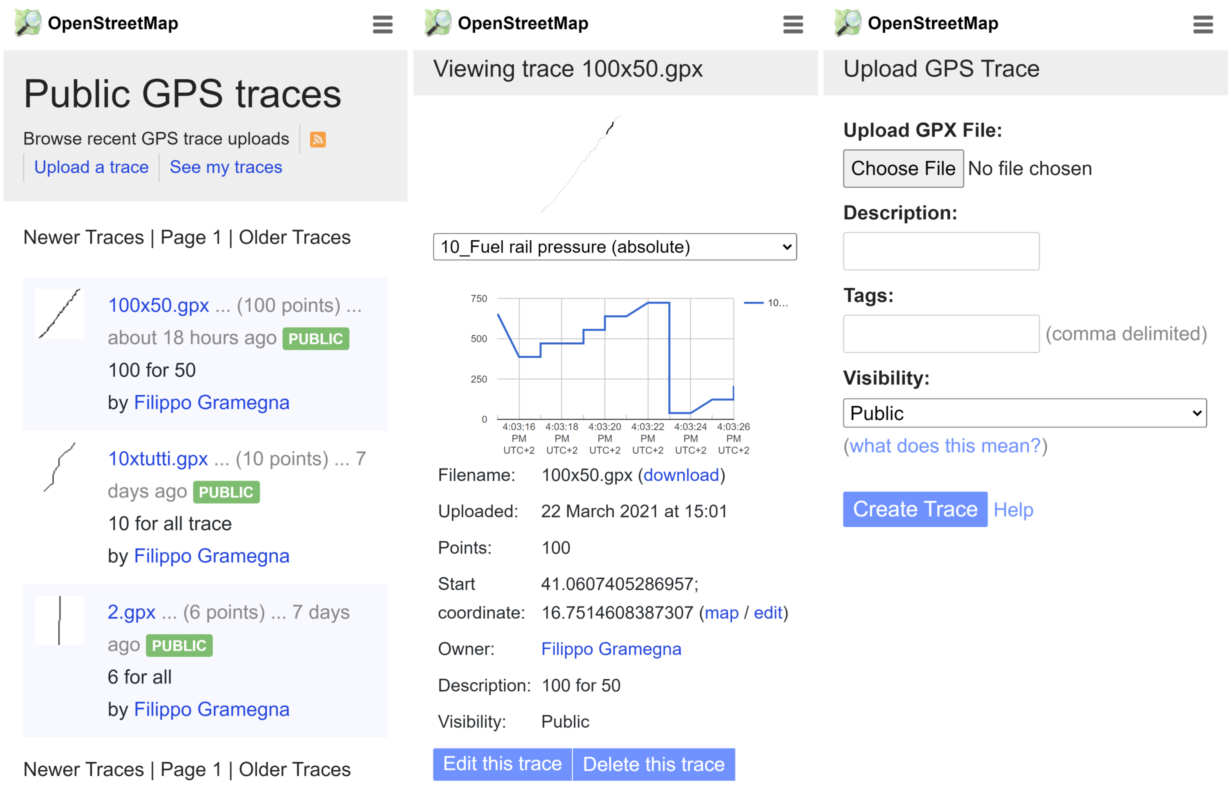
5 Experiments

Performance assessment of the proposed framework and implementation are outlined hereafter followed by a comprehensive discussion.

5.1 Testbed and Method

Apache *JMeter* (`https://jmeter.apache.org`) has been used for load tests on the Web application. It allows simulating server load and analyzing the overall performance under different workload configurations. A test plan has been devised to send several HTTP POST requests to the server API as in case of GPX uploads coming from mobiles requiring to store collected data. The prototype has been implemented by extending a containerized version of the OSM platform, based on *Docker* (`https://www.docker.com`), running on a blade server equipped with an Intel Xeon 16-core CPU, 48 GB RAM, 1 TB storage memory and Ubuntu Server 18.04 LTS 64-bit.

Several scenarios have been defined to analyze the communication between mobile applications and Web system. The following elements have been taken into



Public GPS traces

Browse recent GPS trace uploads

Upload a trace | See my traces

Newer Traces | Page 1 | Older Traces

100x50.gpx ... (100 points) ... about 18 hours ago PUBLIC
100 for 50
by Filippo Gramegna

10xtutti.gpx ... (10 points) ... 7 days ago PUBLIC
10 for all trace
by Filippo Gramegna

2.gpx ... (6 points) ... 7 days ago PUBLIC
6 for all
by Filippo Gramegna

Newer Traces | Page 1 | Older Traces

Viewing trace 100x50.gpx

10_Fuel rail pressure (absolute)

Filename:	100x50.gpx (download)
Uploaded:	22 March 2021 at 15:01
Points:	100
Start coordinate:	41.0607405286957; 16.7514608387307 (map / edit)
Owner:	Filippo Gramegna
Description:	100 for 50
Visibility:	Public

Upload GPS Trace

Upload GPX File: No file chosen

Description:

Tags: (comma delimited)

Visibility: Public

(what does this mean?)

(a) List of uploaded GPX traces **(b) Details of the selected GPX trace** **(c) Upload of a GPX file**

Figure 6: Web platform GPX Traces section

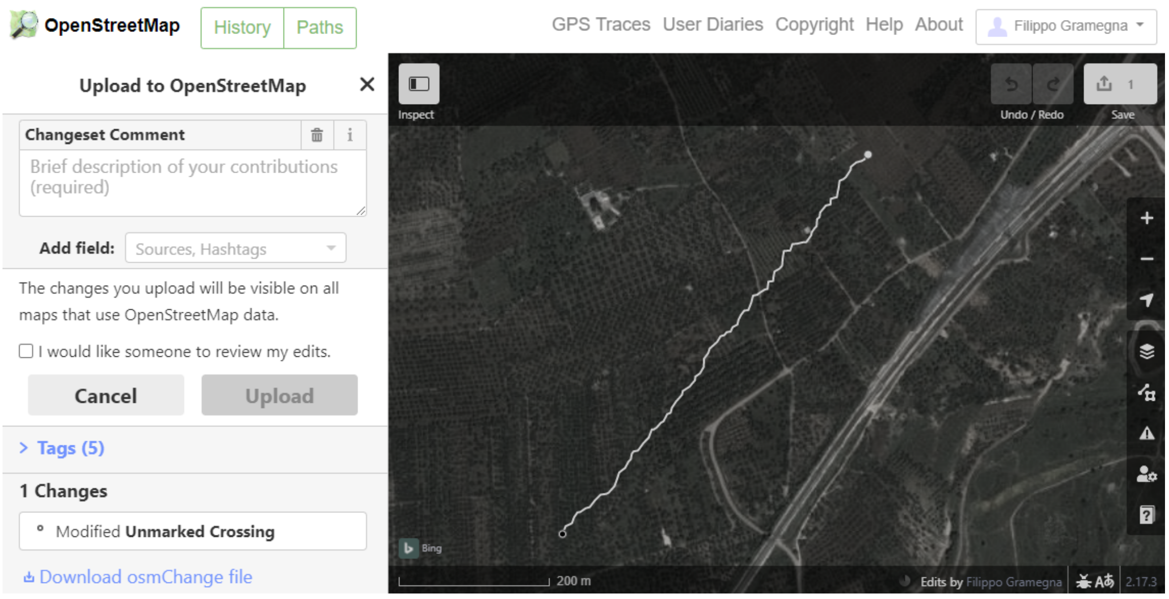
account:

- nodes: different georeferenced locations included in the GPX file;
- vehicle parameters: number of OBD-II features collected by the mobile application;
- threads: concurrent mobile applications interacting with the Web platform;
- ramp-up period: time interval, in seconds, where the threads are distributed to simulate task concurrency;
- routes: number of random paths generated for each scenario.

Experiments have been carried out exploiting 7 configurations, reported in Table 2 and generated by combining the above elements, with growing complexity in terms of number of nodes and collected information. In particular, 10, 100 and then 1000 nodes have been progressively adopted in the GPX trace to model applications collecting data with low, medium and high frequency, respectively. Also a minimum, medium or complete set of vehicle parameters (*i.e.*, 5, 50, 310) defined by OBD-II can be simulated. Table 2 also includes the size of the gathered data and the length of the route used as reference in each scenario. Paths have been generated by setting a reference location and an area through a bounding box. Starting from the initial coordinate, the selected numbers of nodes have been defined by increasing progressively both latitude and longitude within the test area. For each node, random values of the OBD parameters have been annotated employing the demo mode of the mobile application.

A performance assessment for the system considers testing all configurations w.r.t. four workloads (see Table 3). Also in this case, different profiles refer to the Web platform behavior (CPU and RAM usage) to serve requests with increasing computational load. Additional experiments analyze the performance of Web tasks involving end-users. *BlazeMeter* (`https://www.blazemeter.com`) has been exploited to define a set of scripts automating the following interactions with the OSM user interface:

- load the initial map including routes within the visibility area, Figure 5a;



Upload to OpenStreetMap

Changeset Comment

Brief description of your contributions (required)

Add field: Sources, Hashtags

The changes you upload will be visible on all maps that use OpenStreetMap data.

☐ I would like someone to review my edits.

Cancel Upload

Tags (5)

1 Changes

Modified Unmarked Crossing

Download osmChange file

Figure 7: OSM iD editor user interface

Table 2: Configuration for the reference scenarios

ID	Nodes	OBD-II Parameters	Data (MB)	Route Lenght (km)
C1	10	310	0.24	6.16
C2	100	5	0.06	7.01
C3	100	50	0.41	6.75
C4	100	310	2.38	6.87
C5	1000	5	0.60	13.20
C6	1000	50	4.09	16.86
C7	1000	310	23.77	16.73

Table 3: Configuration for load tests

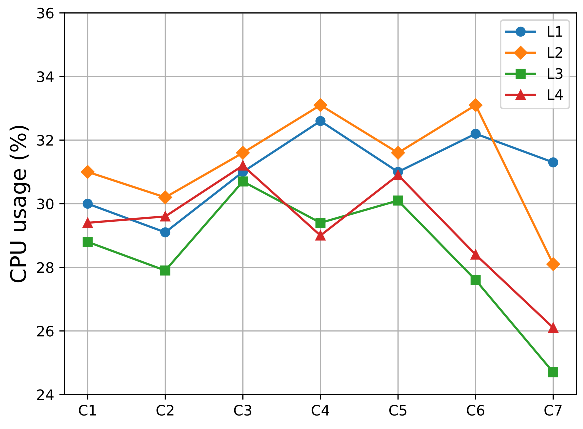
ID	Threads	Ramp-up (s)	Routes
L1	50	10	14
L2	100	60	14
L3	50	10	143
L4	100	60	143

- list details of all paths in the initial map;
- query the server to retrieve all routes within a given zone;
- filter all routes whose annotations include a specific OBD parameter (*e.g.*, EngineRPM, *i.e.*, engine revolutions per minute), Figure 5b.

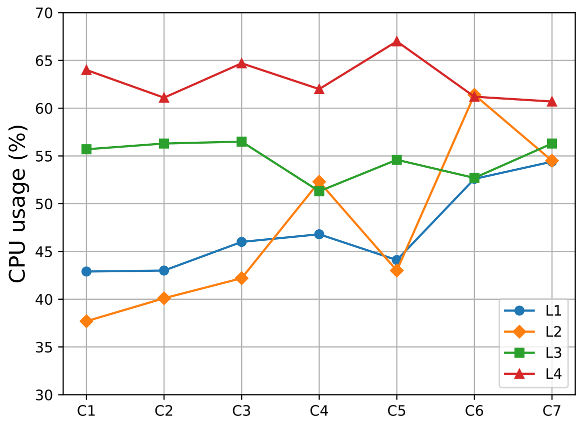
5.2 Results and Discussion

Scalability and usability of the system have been evaluated uploading vehicle data on the Web platform in 28 different reference scenarios, generated by combining the configurations in Table 2 and Table 3. For each scenario, tests have been repeated three times and average values have been reported to filter out the bias deriving from conditions of single runs.

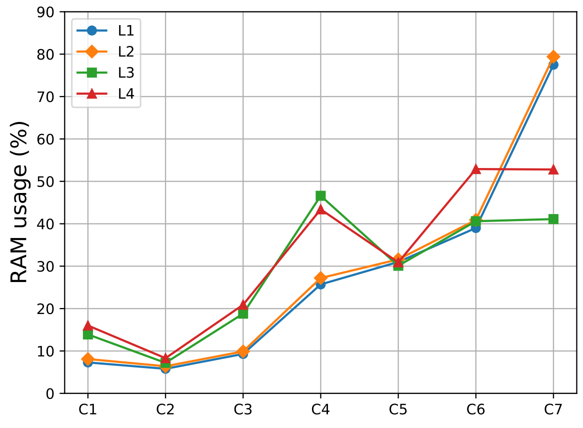
Figure 8a and Figure 8b plot results on the average and maximum CPU load, respectively. As expected, the configurations with 100 threads (L2, L4) require on average a higher CPU usage w.r.t. the corresponding scenarios with 50 threads (L1, L3). Moreover, the configurations with 143 routes (L3, L4) exhibit higher CPU peaks, as reported in Figure 8b, due to the intensive load to process GPX data, but lower values of average CPU load. This is motivated because receiving a great number of routes by multiple clients easily saturates the available memory, causing a decrease in threads concurrency and therefore CPU utilization. In particular, Figure 8e highlights this relation showing both RAM and CPU usage during a whole observation related to the most resource-intensive experiment, *i.e.*, configuration C7 and load test L4. After about 500 seconds, the CPU load tends to decrease, exactly when the memory usage reaches its peak. On the contrary, increasing the number of nodes and parameters affects results slightly in terms of average and maximum CPU usage. Only L1 and L2 show a significant growth of the CPU peak for intensive scenarios (C6, C7), whilst the average CPU utilization decreases for all load profiles only in correspondence



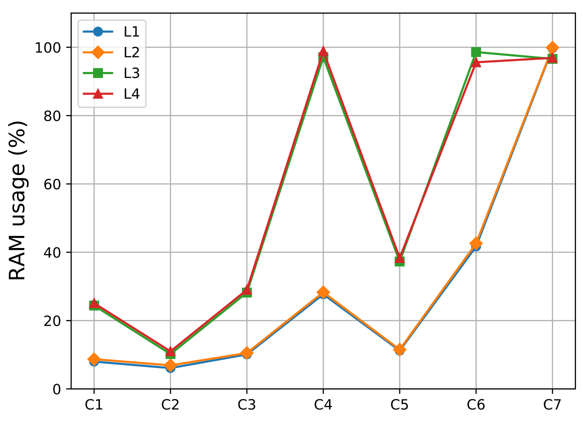
(a) Average CPU usage



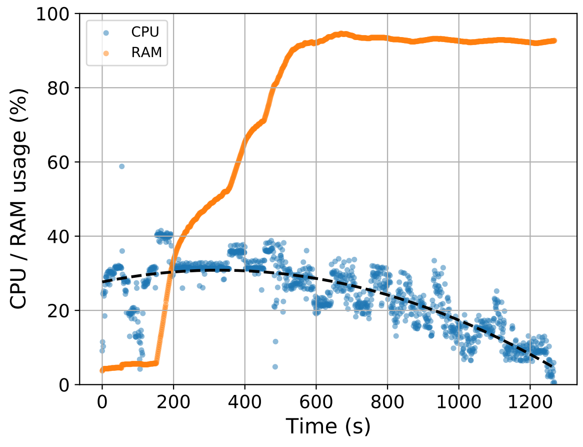
(b) Maximum CPU usage



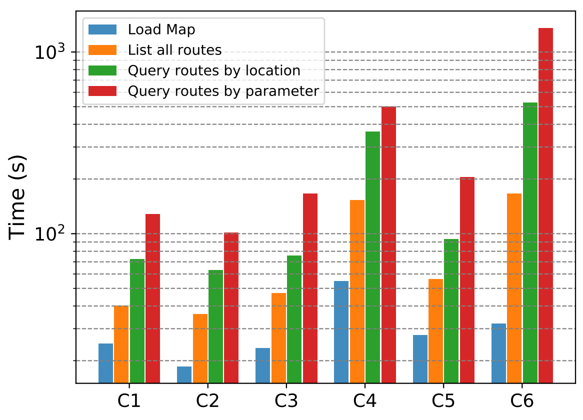
(c) Average RAM usage



(d) Maximum RAM usage



(e) CPU and RAM usage over time



(f) GUI processing time

Figure 8: Performance evaluation of the Web platform

with C7, due to the aforementioned memory saturation.

Similarly, as depicted in Figure 8c, average RAM usage grows according to the number of nodes and OBD-II parameters in the scenarios. Workloads with the same number of routes ($\langle$L1, L2$\rangle$ and $\langle$L3, L4$\rangle$) show similar trends, with higher values in case of more traces. The amount of collected data mainly affects the RAM peak required by the server, as evidenced by Figure 8d. Analyzing the scenarios with the same number of nodes (*i.e.*, sequences $\langle$C2, C3, C4$\rangle$ and $\langle$C5, C6, C7$\rangle$), the reported values rise considerably with the increasing number of OBD features for all load tests.

Finally, Figure 8f presents the processing times related to the main functionalities of the OSM platform in order to evaluate the system usability. Collected data refer to the load configuration L2, as it presents the most stressful conditions, where all tasks were completed with a maximum memory usage lower than 90%. In fact, previous results indicate memory saturation affects the processing times, making evaluations inconsistent. Due to these reasons, the most intensive scenario (C7) is not reported, since it saturates the available memory in all load tests, as shown in Figure 8d. Predictably, processing times increase according to the complexity of scenarios, due to the amount of data to be managed. Loading the initial map is the fastest phase, only retrieving a subset of the stored data. Basically, the Web GUI is configured to center the map on the user's current position with a high zoom level, in order to query and list only routes close to the reference location. Then, the user can ask the datastore to retrieve and show all collected paths. Also in this case, both the number of nodes and OBD-II parameters in paths affect the execution times. As an example, C2, C3 and C4 all have 100 nodes but an increasing amount of gathered data, causing a perceivably slower task in the third configuration. Analogously, C1 and C4 share the same set of OBD-II features but differ in the number of nodes for each path: results show a clear increase of the processing time, due to the different size of stored information. Finally, also the last two tasks result slower for more complex configurations. As expected, time for retrieving routes of a given geographical area (defined through a bounding box) or a reference OBD-II parameter is strongly correlated to the number of available paths to be queried and the consequent data to be reported to the user.

The above experiments support cautious optimism about the scalability of the proposed framework. Scenarios with all 310 OBD-II parameters have been deliberately designed as extreme cases for stress testing; the number of parameters of datasets in Table 1 provide more realistic indications. Nevertheless, the goal of our proposal is to support a worldwide community, therefore the Web platform should ideally support a usage level comparable with OpenStreetMap, for which statistics updated daily [15] report overall numbers of millions of users and hundreds of millions of ways as of 2021. Towards this goal efficiency optimizations on the proposed prototype are surely needed, but load balancing in server clusters will be the main strategy to improve performance by avoiding as much as possible memory saturation and the consequent request stalling.

6 CONCLUSION

In this paper a framework has been introduced for crowdsourcing vehicle data extracted via the OBD-II protocol. Following the OpenStreetMap model, the proposal aims to provide a decentralized platform for large-scale collaborative collection of vehicle information. A public and open repository of OBD data in real driving conditions can be a valuable asset for ADAS/AD research as well as other innovative ITS applications and services.

The approach comprises an extension of the GPX XML Schema to embed OBD-II parameters in GPS logger traces, a mobile Android client extending the open source AndrOBD application with trace logging and upload and a Web platform expanding OSM with storage, search and editing of OBD-augmented routes. A full prototype of the framework has been developed and evaluated in a preliminary experimental campaign to assess usability in terms of turnaround time and scalability of the data-intensive route upload task under various workload profiles. Results show performance optimization of the Web application is needed to sustain the highest workloads, but in typical usage conditions the platform follows the expected behavior.

Future work aims at optimizing and tightening the codebase of both the client and server components before publishing the platform online, as well as dealing with the licensing and privacy policies. From a research viewpoint, an investigation has started on the integration of the framework with a blockchain platform for secure and verifiable storage of route digests, in order to support ITS use cases such as fleet management and monitoring.

ACKNOWLEDGEMENTS

This work was supported by Italian Ministry of Economic Development R&D project *BARIUM5G* (Blockchain and ARtificial Intelligence for Ubiquitous coMputing via 5G).

REFERENCES

[1] H. Abut, H. Erdoğan, A. Erçil *et al.*, "Data collection with UYANIK: too much pain; but gains are coming," in *Corpus and Signal Processing for Driver Behavior*, K. Takeda, J. Hansen, H. Erdoğan, and H. Abut, Eds. Springer Business-Science, 2008.

[2] P. Angkititrakul, D. Kwak, S. Choi, J. Kim, A. PhucPhan, A. Sathyanarayana, and J. H. Hansen, "Getting start with UTDrive: Driver behavior modeling and assessment of distraction for in-vehicle speech systems," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[3] W. Anwar, N. Franchi, and G. Fettweis, "Physical layer evaluation of V2X communications technologies: 5G NR-V2X, LTE-V2X, IEEE 802.11 bd, and IEEE 802.11 p," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–7.

[4] C. A. d. S. Barreto, "Uso de técnicas de aprendizado de máquina para identificação de perfis de uso de automóveis baseado em dados automotivos," Master's thesis, Brasil, 2018.

[5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understandingof LiDAR Sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[6] Y. Benkler, "Peer production and cooperation," in *Handbook on the Economics of the Internet*. Edward Elgar Publishing, 2016.

[7] C. A. R. Board, "On-Board Diagnostic II (OBD II) Systems Fact Sheet," https://ww2.arb.ca.gov/resources/fact-sheets/board-diagnostic-ii-obd-ii-systems-fact-sheet, 2019, accessed: 2021-05-04.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[9] G. Dong, Y. Yan, C. Shen, and H. Wang, "Real-time high-performance semantic image segmentation of urban street scenes," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[10] D. Foster, "GPX: the GPS Exchange Format," https://www.topografix.com/gpx.asp, 2004, accessed: 2021-05-04.

[11] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. Hoang Pham, M. Mühlegg, S. Dorn *et al.*, "A2D2: Audi Autonomous Driving Dataset," *arXiv e-prints*, pp. arXiv–2004, 2020.

[12] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape Open Dataset for AutonomousDriving and Its Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2702–2719, 2020.

[13] B. I. Kwak, J. Woo, and H. K. Kim, "Know your master: Driver Profiling-based Anti-theft method," in *14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2016, pp. 211–218.

[14] K. McCord, *Automotive Diagnostic Systems: Understanding OBD I and OBD II*. CarTech Inc, 2011.

[15] OpenStreetMap, "OpenStreetMap Statistics," https://www.openstreetmap.org/stats/data_stats.html, 2021, accessed: 2021-05-04.

[16] P. Parkinson and L. Kinnan, "Safety-critical Software Development for Integrated Modular Avionics," *Embedded System Engineering*, vol. 11, no. 7, pp. 40–41, 2003.

[17] J. Pillmann, C. Wietfeld, A. Zarcula, T. Raugust, and D. C. Alonso, "Novel Common Vehicle Information Model (CVIM) for future automotive vehicle big data marketplaces," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1910–1915.

[18] M. A. Rahman, M. M. Rashid, S. J. Barnes, and S. M. Abdullah, "A Blockchain-based Secure Internet of Vehicles Management Framework," in *2019 UK/China Emerging Technologies (UCET)*. IEEE, 2019, pp. 1–4.

[19] M. Ruta, F. Scioscia, F. Gramegna, S. Ieva, E. Di Sciascio, and R. Perez De Vera, "A knowledge fusion approach for context awareness in vehicular networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2407–2419, 2018.

[20] M. Ruta, F. Scioscia, G. Loseto, A. Pinto, and E. Di Sciascio, "Machine learning in the Internet of Things: A semantic-enhanced approach," *Semantic Web*, vol. 10, no. 1, pp. 183–204, 2019.

[21] SAE Standard J1962, "Diagnostic Connector Equivalent to ISO/DIS 15031-3," Society of Automotive Engineers, Standard, 2002.

[22] SAE Standard J1979, "E/E Diagnostic Test Modes - Equivalent to ISO/DIS 15031-5," Society of Automotive Engineers, Standard, 2002.

[23] SAE Standard J3016_201806, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Society of Automotive Engineers, Standard, June 2018.

[24] Statista, "Connected Cars Worldwide," https://www.statista.com/study/21443/connected-cars-statista-dossier/, Statista GmbH, Dossier, 2020, accessed: 2021-05-04.

[25] A. Stocker, C. Kaiser, and A. Festl, "Automotive Sensor Data. An Example Dataset from the AEGIS Big Data Project," http://doi.org/10.5281/zenodo.820576, 2017, accessed: 2021-05-04.

[26] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

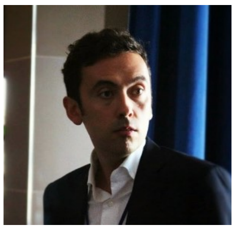
[27] G. F. Türker and A. Kutlu, "Survey of smartphone applications based on OBD-II for intelligent transportation systems," *International Journal of Engineering Research and Applications*, vol. 6, no. 1, pp. 69–73, 2016.

AUTHOR BIOGRAPHIES

Giuseppe Loseto received the master's degree in Information Technology Engineering from Polytechnic University of Bari in 2009, and the Ph.D. in Information Technology Engineering in 2013 from the same institution. He is currently a post-doc research fellow at the Information Systems Laboratory (SisInfLab) in the same University. His research interests include pervasive computing and the Internet of Things, knowledge representation systems and applications for ubiquitous smart environments. On these topics, he has co-authored over 50 papers in international journals and conferences.

Filippo Gramegna received the master's degree in Information Technology Engineering from Polytechnic University of Bari, in 2009. After several years of collaboration, he is currently a research assistant at the Information Systems Laboratory (SisInfLab) in the same university, working on mobile knowledge representation and reasoning systems for ubiquitous and pervasive contexts, semantic-enhanced real-time vehicle monitoring and driving assistance. He has co-authored several papers in international workshops and conferences on his research interests.

Agnese Pinto received the master's degree in Management Engineering and the Ph.D. in Electrical and Information Engineering from Polytechnic University of Bari in 2004 and 2015, respectively. She is currently a Postdoctoral Research Fellow with the same institution. Research activity of Agnese Pinto started in 2004 soon after graduation. Her research interests include Description Logics, semantic matchmaking, ubiquitous knowledge management and storage, machine learning for data mining in pervasive environments. She has co-authored papers about her research interests in international journals and conferences.



Michele Ruta received the master's degree in Electronics Engineering from Polytechnic University of Bari in 2002 and the Ph.D. in Computer Science in 2007. He is currently full professor. His research interests include pervasive computing and ubiquitous web, knowledge representation systems and applications for wireless ad-hoc contexts. On these topics, he has co-authored more than 140 papers in international journals, edited books and conferences. He is involved in international and national research projects and is Program Committee member of several international conferences and workshops. He is also editorial board member of journals in areas related to his research interests.

Floriano Scioscia received the master's degree in Information Technology Engineering with honours from Polytechnic University of Bari in 2006, and the Ph.D. in Information Engineering in 2010 from the same institution. He is currently an assistant professor at the Information Systems Laboratory (SisInfLab) in the same university. His research interests include knowledge representation systems and applications for pervasive computing, cyber-physical systems and the Internet of Things. He co-authored over 80 papers in international journals, edited books and conferences.