OpenVDAP: An Open Vehicular Data Analytics Platform for CAVs

Qingyang Zhang*†, Yifan Wang*‡, Xingzhou Zhang*‡, Liangkai Liu*, Xiaopei Wu*, Weisong Shi* and Hong Zhong†
*Department of Computer Science, Wayne State University, Detroit, MI, USA., 48202
†School of Computer Science and Technology, Anhui University, Hefei, China, 230601
‡Institute of Computing Technology, University of Chinese Academy of Sciences, Beijing, China, 100190
{qyzhang, liangkai, xiaopei.wu, weisong}@wayne.edu, {wangyifan2014, zhangxingzhou}@ict.ac.cn, zhongh@ahu.edu.cn

***Abstract*—In this paper, we envision the future connected and autonomous vehicles (CAVs) as a sophisticated computer on wheels, with substantial on-board sensors as data sources and a variety of services [1] running on top to support autonomous driving or other functions. In general, these services are computationally expensive, especially for the machine learning based applications (*e.g.,* CNN-based object detection). Nevertheless, the on-board computation unit possess limited compute resources, raising a huge challenge to deploy these computation-intensive services on the vehicle. On the contrary, the cloud-based architecture *conceptually* with unconstrained resources suffers from unexpected extended latency that attributes to the large-scale Internet data transmission; thus, adversely affecting the services' real-time performance, quality of services and user experiences. To address this dilemma, inspired by the promising edge computing paradigm, we propose to build an Open Vehicular Data Analytics Platform (*OpenVDAP*) for CAVs, which is a full-stack edge based platform including an on-board computing/communication unit, an isolation-supported and security & privacy-preserved vehicle operation system, an edge-aware application library, as well as an optimal workload offloading and scheduling strategy, allowing CAVs to dynamically detect each service's status, computation overhead and the optimal offloading destination so that each service could be finished within an acceptable latency and limited bandwidth consumption. Most importantly, contrast to the proprietary platform, *OpenVDAP* is an open-source platform that offers free APIs and real-field vehicle data to the researchers and developers in the community, allowing them to deploy and evaluate applications on the real environment.**

***Index Terms*—edge computing; vehicular data analysis; vehicular operating system; security & privacy; connected and autonomous vehicles.**

I. Introduction

As the rapid growth of technologies in communication, chip processing, sensing and machine learning, vehicles are becoming increasingly connected and automated [1]–[4]. A typical connected and autonomous vehicle is often equipped with a High Definition (HD) map that provides CAVs with detailed road data, such as the road shoulders, and a plethora of diverse sensors, *e.g.,* light detection and ranging (LiDAR), radio detection and ranging (radar) and camera, to monitor the vehicle itself and its surroundings. According to Intel, a CAV will generate 4TB of data per day in the near future [5]. Although the sensor data can be transmitted to and processed in a remote cloud, the bandwidth and latency challenges make it impossible to ensure the real-time service (*i.e.,* autonomous driving) provides timely decision all the time. Thus, the computing solution adopted by the autonomous driving platform is to process data on the vehicles themselves.

Keeping data processing on the vehicle is effective in offering real-time services. However, doing this will impose a huge challenge on the on-board computing unit, especially for the future CAVs that include more sensors and third-party applications with varying degree of computation demands. One naive solution is to add extra computing chips (*e.g.,* GPU or DSP) once the on-board compute resource is not sufficient to support upper services. Since the powerful processors are often power-hungry, usually on the order of hundreds of watts for GPU, this method will cause processor overheating and a concern for energy consumption.

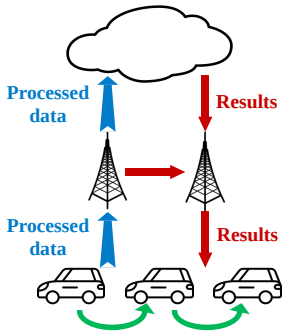

Fig. 1. Data processing model in an edge-based solution.

In this paper, we envision the future CAVs as a sophisticated computer on wheels with a variety of services, such as advanced driver-assistant systems (ADAS), remote real-time diagnostics, in-vehicle infotainment, other third-party applications running on top. These applications could be the embedded services on the vehicle or newly added third-party services. Note that they are usually based on large-scale data analytic, thus quite computation expensive. For the in-vehicle only computing model, these applications need to contend with each other for the limited on-board computing resources. For instance, assume two latency-sensitive applications require

[1]In this paper, we will use services and applications interchangeably to represent the functions that are available on CAVs.

execution on the GPU at the same time. If there is only one GPU on the vehicle, the second scheduled application might not produce a *timely* decision.

The emerging edge computing [6], [7] (also referred to as fog computing [8], cloudlet [9], [10]) is a novel networked computing architecture which will deploy the compute, storage, and communication resource at the network edge, enabling the latency-sensitive and confidentiality-aware applications to be performed in proximity of the data source, thus removing extra data transmission time and potential security and privacy risk. As shown in Figure 1, the vehicles are able to communicate with each other and send processed data to nearby (usually one-hop away) edge servers (*i.e.,* base stations) located between remote cloud and edge vehicles, with more powerful compute resources than the on-board computing unit, at the same time, producing much smaller latency and less chance in privacy leaking, compared with the cloud solution.

Research on CAV technologies is one of the hottest areas. In addition, machine learning based real-time applications dominate on CAVs. Unfortunately, there are no edge computing platforms that support vehicular data analytic and processing. Many companies, such as Ford [11], General Motors [12] and Baidu, are working on this. Except for Baidu's Apollo [2], they are all proprietary. Though Apollo is open source, it relies on the in-vehicle only computing solution that puts all the data processing on the vehicle, which is neither scalable nor suitable for the future CAVs with plenty of third-party services.

Motivated by the above observations, we propose to build an edge computing based Open Vehicular Data Analytics Platform (*OpenVDAP*) for future CAVs which is a full-stack edge supported platform that includes a series of heterogeneous hardware and software solutions. The key characteristics of *OpenVDAP* are summarized below:

- *OpenVDAP* supports on-board computing environment by providing multiple carefully selected heterogeneous computing processors and a security and privacy-preserved vehicle operating system to ensure a safe and trusted execution environment for upper applications all while maintaining effective and optimal resource management and utilization for lower diverse hardware.
- In addition to the on-board computing, *OpenVDAP* is two-tier computing architecture based. In particular, *OpenVDAP* provides systematic mechanisms on how to request, utilize, share and even collaborate with external computing entities located on neighboring vehicles, nearby roadside edge servers, or remote cloud servers. And a dynamic offloading and scheduling algorithm is included to allow *OpenVDAP* to detect each service's status, computation overhead, and the optimal offloading destination so that each service could be completed at the right time with limited bandwidth consumption.
- The *OpenVDAP* expected to run on an autonomous vehicle test-bed offers an open and free edge-aware application library named `libvdap` that contains how to access and deploy edge-computing based vehicle applications, various commonly used artificial intelligent (AI) models, as well as the interface of accessing open real field vehicle data, all of which will enable the researchers, developers in the community to freely deploy, test and validate their applications in the real environment.

The remainder of this paper is organized into six sections. In Section II, we introduce four types of in-vehicle services on the CAVs. In Section III, we discuss three potential computing architectures for CAVs as well as their corresponding challenges on bandwidth requirement, energy consumption, and security and privacy. We present our edge-based solution, *OpenVDAP*, an open source, full-stack vehicular data analytics platform in Section IV. Related works are revisited in Section V and Section VI concludes the paper.

II. In-Vehicle Services

In this section, we briefly discuss four types of services that will be available on CAVs. Conventionally, these services on current vehicles could be classified into three groups according to their functionality: *real-time diagnostics*, *advanced driver-assistant systems*, and *in-vehicle infotainment*. In addition, we envision a new type of services, third-party applications from various vendors, will be prevalent on CAVs as the vehicle data in the future will not be exclusive to the automakers.

A. Real-time Diagnostics

This type of service usually refers to the On-board diagnostics (OBD) system, which allows the vehicle to have the capability of self-diagnosis and reporting. The OBD system appeared on the vehicle in the 1980s and has evolved from the early simple "idiot light" to a modern version that can provide real-time vehicle data (*e.g.,* the engine's revolution from the engine control unit) and a standardized series of diagnostic trouble codes. Such code is useful for vehicle maintenance and repair. The device reading the real-time data is actually an additional device to the vehicle called the OBD reader. The maintainer can leverage the OBD reader to obtain information about the fault, *e.g.,* the diagnostic trouble code. Thus, this usually will not consume any resources of the vehicle. However, it is not an in-vehicle system. In future CAVs, this type of service should be built in the vehicle, which collects the related vehicle data, including real-time data and historical data, and quietly analyzes it to predict faults. Thus, it can remind the owner of keeping the vehicle in good condition.

B. Advanced Driver-Assistant Systems

Nowadays, more and more vehicles are equipped with the ADAS that can detect some objects, complete basic classification, and alert the driver of unsafe driving behaviors. It may also slow or stop the vehicle. For example, the steering wheel will vibrate when the vehicle is close to a traffic line without illuminating the turn light. The Society of Automotive Engineers (SAE) International grades autonomous driving at six levels [13], in which the level 0 means the vehicle is without any assistant, and the level 5 means the vehicle is in full control by an autonomous driving system. A vehicle with ADAS is usually rated the SAE level 1 or level 2 based on

provided functions. Usually, level 3 means the human driver can safely turn their attention away from driving tasks. As the level ascends, the autonomous driving system takes over more controls from human drivers and needs more computation resources to run computational intensive algorithms.

TABLE I
THE PERFORMANCE OF AUTONOMOUS DRIVING -RELATED ALGORITHMS.

Name	Latency (ms)
Lane Detection	13.57
Vehicle Detection (Haar)	269.46
Vehicle Detection (TensorFlow)	13971.98

The most important element of ADAS is the real-time object detection that is based on either computer vision or deep learning technology. To obtain intuitional perception on detection algorithm's computation complexity, we chose two of the most representative detection algorithms: Lane Detection and Vehicle Detection. The former relies on the computer vision technology. Regarding the latter, the underlying technologies are Haar-based image processing and TensorFlow-based deep learning algorithm. We conducted our experiments on the AWS EC2node with 2.4GHz vCPU. For the vehicle detection, our initial results, as shown in Table I, show that the latency of Haar-based algorithm significantly outperforms (around 51x faster) than the TensorFlow-based. But in the future CAVs, deep learning based algorithms will dominate since they can detect multiple types of objects at once. To reduce the system latency, more powerful hardware is highly required.

C. In-Vehicle Infotainment

In-Vehicle infotainment includes a wide range of services that provide audio or video entertainment. For example, the driver uses the radio to listen to music, including cloud services from the Internet such as Pandora, and the passengers sitting in the backseat can relax by watching online videos or news using the device embedded on the seat. This means these services might involve large-scale Internet data transmission. Most mainstream auto vendors have Internet-supported infotainment services, e.g., Uconnect for Chrysler, Blue Link for Honda, and iDrive for BMW. Another trend of on-board infotainment is the Android-based system that has been implemented by many auto vendors including Honda, Hyundai, Audi, and Volvo. For these services, video or audio data must be downloaded from the Internet and then decoded locally (*i.e.,* on the vehicle) to make the data smooth or at higher quality, and eventually delivered to the passengers.It means these applications not only require compute resources but also present a high requirement on the network bandwidth.

D. Third-Party Application

The in-vehicle third-party application provided by a vendor other than the car maker is used to enhance the user experiences or provide other add-on services (*e.g.,* finding a missing car for law enforcement). The trend of openness of vehicle data will make it easier for third-party vendors to develop and deploy different kinds of applications on the vehicle. In addition, with the rapid development of in-vehicle processor processing, vehicle to vehicle communications and autonomous driving technologies, future CAVs could be viewed as a sophisticated computer on wheels with a variety of third-party applications.

Several projects including these types of applications have been initiated. For example, Kar *et al.* [14] proposed an application to enhance the vehicle safety by detecting whether the driver is registered or not through analyzing his/her operation features (*e.g.,* the time duration of door open and close). Another example is to leverage the on-board camera to recognize and track a targeted vehicle, which is a mobile version for A3 [15] promising to enhance the AMBER alert system. Generally, the core of such types of applications is either vision-related or machine learning based data processing algorithm fed by the same on-board sensor data (*e.g.,* dash camera). Since the core algorithms are computationally intensive, this type of third-party applications needs powerful computing hardware as well.

III. PROBLEMS AND CHALLENGES

The latency and accuracy are equally paramount for many in-vehicle services, especially for the safety-critical ones supporting autonomous driving. Any incorrect or slow decisions could be disastrous. Due to the large-scale data processing, limited compute resources, and unreliable network bandwidth in the moving context,it is not trivial to build an efficient architecture that can simultaneously satisfy these requirements. Moreover, security & privacy is another big concern for future CAVs as the future CAVs are more connected. In this section, we will discuss three architectures: *Cloud-based Solution*, *In-vehicle Based Solution* and *Edge-based Solution*, and point out their corresponding problems and limitations. We end this section by separately discussing the problems of security & privacy, as well as isolation for CAVs.

A. Problems and Challenges of Cloud-based Solutions

Cloud computing is often characterized as a powerful server with conceptually unlimited computational resource. However, it needs to transmit large-scale data over the Internet. Unfortunately, the network quality (*e.g.,* connectivity, bandwidth) cannot always be guaranteed, most likely leading to extra unpredictable data transmission time, and eventually a considerable higher response latency. The biggest challenge of the cloud-based solution is to provide reliable and fast Internet transmission.

As previously mentioned, around 4TB of data will be generated daily on a CAV. It will take around 18 days to accomplish data uploading procedure via current LTE, while keeping an average upload rate of 20Mbps. Thus, current cellular network fails to serve as the basic data conduit to stream all vehicle data to the cloud at the right time. Alternatively, the 5G with a larger upload bandwidth might be a good candidate. Nevertheless, there are two big barriers. First is the network coverage and cost, making 5G inaccessible

to many vehicles. Secondly, the move of vehicle may cause substantial data packet loss, which was validated under current cellular network via a group of real-field experiments.

Regarding the experiments, we drove a car in Detroit at the speed of 35 miles per hour (MPH) and 70 MPH, and deployed a video uploading procedure on the car. The LTE network was chosen as the Internet access method. The network protocol that we used was UDP-based Real-time Transport Protocol, in which the packet re-transmission mechanism is not supported. Two sets of 5-minutes videos with different resolution qualities were used as the test data: 1280×720 (720P) and 1920×1080 (1080P), both encoded into H.264 with 30 frames per second and one key frame per two seconds. The measured metrics are packet loss rate on the network level and frame loss rate on the application level.

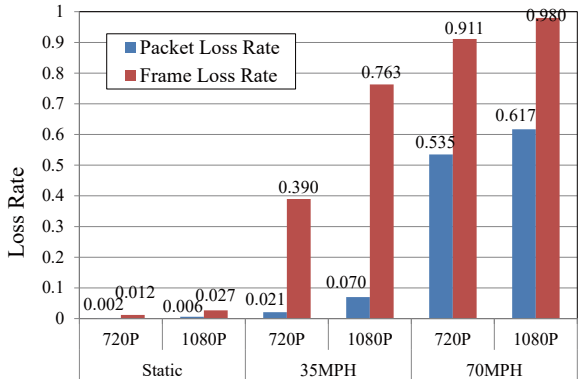

Fig. 2. The packet and frame loss rates in different scenarios.

The experimental results are depicted in Figure 2. For comparison purpose, we also measured the data loss rate in the static environment. As expected, both the packet loss rate and frame loss rate increase significantly compared with the static case. And in the moving context, the data loss rate increases exponentially with the increase of moving speed. At the speed of 70MPH, this becomes worse for the higher resolution data (1080P), with more than 80% data loss rate. The underlying reasons are straightforward. First, the higher speed may lead to the vehicle's stay time within the coverage of its closest base station pretty short, making the Internet connection between the vehicle and the cellular base station highly unreliable.In other words, the vehicle might disconnect from the Internet during the process of base station change. Secondly, the higher video resolution contains more pixels, accordingly requires higher network bandwidth for successful transmission. For example, the bandwidth of transmitting a live 1080P video is around 5.8Mbps, while the lower bound is 3.8Mbps for a 720P video.

Note that the frame loss rate is bigger than the packet loss rate for all the cases. The primary reason is the difference of counting policy. Given a frame, the rule of marking a frame as lost is based on whether its first key frame is lost or not, rather than on its own status. Therefore, if the first key frame is lost, all its successive frames will be viewed as lost even if they might be successfully delivered. While the packet loss doesn't consider such interdependent counting policy.

Our preliminary experiments demonstrate the biggest challenge of using Cloud based two tier computing architecture for CAVs is the unreliable, or even unpredictable network quality between the vehicle and Cloud in the moving context.

B. Problems and Challenges of In-vehicle-based Solutions

Due to the unreliable network quality, most of the autonomous vehicles use the in-vehicle-based solution, in which all the data are processed by a computing unit on the vehicle. As mentioned in section II-B, many autonomous driving algorithms are computationally intensive. Thus, a heterogeneous platform with varying degree of processing power is suitable for the real-time data processing dominated context. Intuitively, such a platform will lead to enormous energy consumption as the computation tasks increase.

To support our perception, we conducted a CNN based image recognition, *Inception v3*, on CPU (Intel i7-6700), 3 types of GPU with different power levels and a DSP-based processor (Intel Movidius Neural Compute Stick [16]). We measured the total processing time as shown in in Figure 3. Here, the GPUs are the Max-Q model and Max-P model of NVIDIA Jetson TX2, labeled as GPU#1 and GPU#2. The GPU#3 represents NVIDIA Tesla V100. Obviously, GPU#3 outperforms other kinds of processors inprocessing speed, while its corresponding max power consumption is considerably bigger than others.

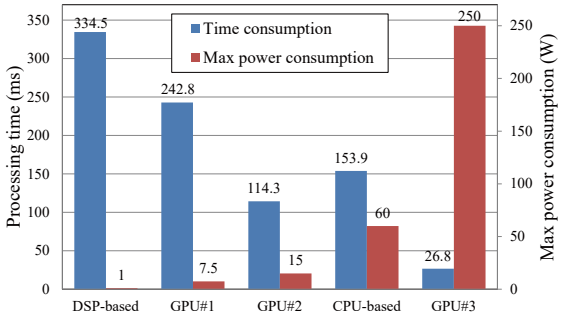

Fig. 3. Performance of running Inception v3 on various processors.

In regards to the real-time performance, the pure in-vehicle solution that adds different types of processors will result in high energy consumption, especially for electric vehicles. For example, the NVIDIA Drive PX2 [17], supporting fully autonomous driving (level 5), consist 2 CPUs and 2 GPUs, of which max power consumption is 500 watts. However, it might be hard to support various additional third-party applications at the same time, especially for computational intensive applications, such as A3. Thus, more powerful computing unit, with higher energy consumption on computing and heat dissipation, is needed, which will affect the mileage per discharge cycle. In this case, EV vendors, e.g., GM, BYD, do want to reduce the energy consumption on computing. which will leave more energy for the car driving purpose.

Sharing recognition results of ADAS between vehicles might be a possible approach, which avoids repeating computing. Another possible approach is to avoid directly executing computational intensive algorithm, which could be achieved by developing lightweight algorithms without sacrificing the final processed results. For example, the Inception v3 recognizing tens of thousands of objects at once could be tailored and customizedwith limited interested objects, such as pedestrians and road blocks, to reduce the computation overhead.

C. Problems and Challenges of Edge-based Solutions

The emerging edge computing refers to various technologies that allows computation to be performed at the edge of the network, which is promising for latency-sensitive applications. The edge computing based solution usually contains a remote Cloud as well. Therefore, such a solution will enable CAVs to offload workload to the edge servers and a remote Cloud server, and where the workload will go depends on its tolerance to latency. To avoid redundant computing, the edge based solution also considers the processed results sharing scheme through the dedicated short-range communications (DSRC), a key communication part on CAVs. Although edge computing is promising in dealing with latency-sensitive applications, there still exist several challenges as noted below:

- *Workload offloading*: Migrating the workload from the cloud to the edge will reduce transmission latency. As shown in [18], a license plate number recognition process is split into three parts, including motion detection, license plate detection and license plate number recognition. These three parts are able to be executed on different devices concurrently. However, how to dynamically schedule the sub-workloads to achieve the best end-to-end latency in terms of network quality and vehicle residual compute power is an open issue.
- *Collaboration*: Though the collaboration of vehicles can save computing power by avoiding executing unnecessary repeating operations, a collaboration mechanism does not exist in the literature to the best of our knowledge. To bridge this gap,we should identify what type of data can be shared safely, and then know how to share processed data. Moreover, the synchronization is also an open issue when sharing data across multiple entities.

D. Security, Isolation and Privacy

Besides the computing architecture, security, privacy and isolation are essential for future CAVs, due to the openness of wireless communication, data sharing, and vast third-party applications. In this section, we will discuss these problems.

- ***Trusted execution environment***: The availability of diverse on-board wireless communication interfaces (*e.g.,* DSRC, cellular network, Bluetooth) make the CAV be more vulnerable to be attacked, increasing the chance of being remotely controlled by external attackers, which creates new security risks. On one hand, to overcome such security concerns, the firewall as a basic can be used to protect some attacks. On the other hand, the key and safety-critical applications (*e.g.,* autonomous driving services) could rely on the trusted execution environment (TEE) [19], a promising technique to ensure the security of these key services through running encrypted instructions in the memory.
- ***Isolation***: As more and more third-party services are added to the vehicle, the services running on the same vehicle may form another type of security risk referred to as an internal attack that is from another other services, rather than a remote malicious party. Although the above mentioned TEE could be used as the service isolation solution, it only targets for a few key services. Therefore, it is highly desirable to have an effective isolation scheme for all potential services regardless of their significance in safety driving. This problem will become more serious in the context supporting collaboration between vehicles. For example, the service might be migrated from a neighbor vehicle which may not be trustworthy.
- ***Privacy preserved data sharing***: In the future the vehicle data, either raw sensor readings or processed results, will be shared with other external entities. For example, the personalized recommendation service needs to frequently send the location data to a remote server to provide better service, which has a risk of leaking the most sensitive personal data (*e.g.,* home address, medical disease) via simple GPS trace analysis. Therefore, how to protect the privacy in data sharing is an indispensable part as well.

IV. *OpenVDAP*: Overview and Design

In this section, we will introduce our work, Open Vehicular Data Analytics Platform (*OpneVDAP*), an edge-based solution for future CAVs. Considering the constraints of network (*i.e.,* latency, bandwidth, and connectivity) and onboard computing resources, *OpenVDAP* enables the vehicle to collaborate with surrounding vehicles, offloading workloads to edge servers, denoted as *XEdge*, which could be running on base stations, RSUs, and traffic signal systems, as well as on remote Cloud. Below is an overview of *OpenVDAP* followed by a detailed discussion of each component.

A. Overview of OpenVDAP

OpenVDAP is a full-stack vehicle computing platform for future CAVs with the capabilities of collaborating with other edge nodes (*i.e.,* nearby vehicles), XEdge, and a remote Cloud. It consists of four in-vehicle components: an on-board heterogeneous computing/communication unit (VCU), an isolation-supported and security/privacy-aware vehicle operating system (EdgeOS_v), a driving data integrator (DDI), and an edge-aware application library (`libvdap`),

Specifically, the VCU is a heterogeneous computing unit in terms of processors, storage, and communication modules. On top of VCU, it is an operating system, EdgeOS_v, which is responsible for providing a security running environment and energy-efficient and latency-aware resourcemanagement for upper applications. A library, `libvdap`, is provided allowing the third-party developers to build their own applications on

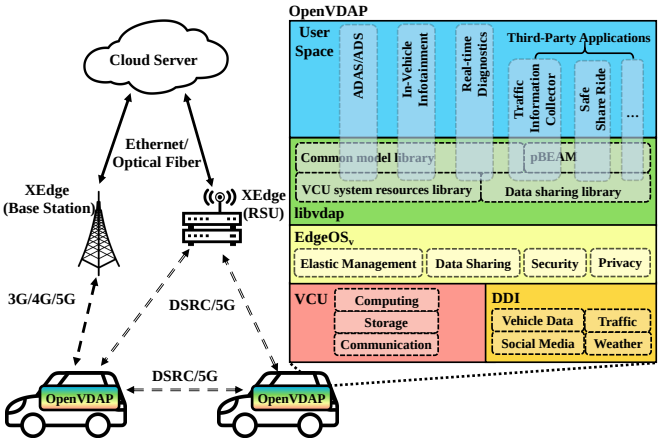

Fig. 4. The overview of *OpenVDAP*.

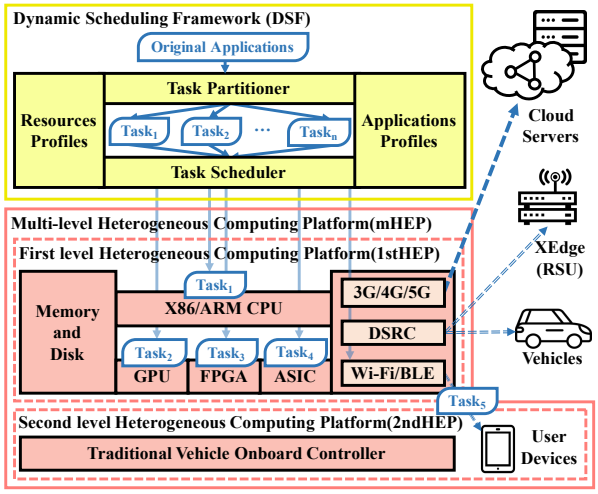

Fig. 5. The overview of VCU.

the *OpenVDAP*. Additionally, *OpenVDAP* also contains a service called DDI that can collect all driving data generated by the in-vehicle sensors (*e.g.,* OBD and dash camera) and other external information (*e.g.,* local weather from the Internet). All data collected by the DDI will be cached on the vehicle and eventually migrated to a cloud based data server. Note that these data will be open to the community.

To enable collaboration between vehicles, XEdge and a remote Cloud, *OpenVDAP* supports multiple wireless interfaces such as DSRC, 5G, 3G/4G/LTE, WiFi and Bluetooth, in which the DSRC and 5G will be used for vehicle-to-vehicle and vehicle to RSU-based XEdge communication due to their higher bandwidth. In addition, the vehicle can communicate with the base station via traditional cellular networks (*e.g.,* 3G/4G/LTE). As the communication between the RSU/base station and cloud, the wired Ethernet or Optical Fiber will be used.

B. VCU: Heterogeneous Vehicle Computing Unit

Traditionally, vehicles usually contain a computing unit on board, also known as on-board controller. However, this on-board controller does not provide any open interface to other users/developers. In general, it has very limited computing power, failing to support the state-of-the-art applications, such as autonomous driving. Hence, we propose a new computing platform for vehicles called *Heterogeneous Vehicle Computing Unit (VCU)*, mainly consisting of a *multi-level heterogeneous computing platform (mHEP)* and a *dynamic scheduling framework (DSF)*. The architecture of VCU is shown in Figure 5, in which mHEP relies on heterogeneous hardware devices to provide the functions of computing, communication and storage, and DSF to effectively utilize and manage the underlying hardware resources, especially the heterogeneous processors. The mHEP and DSF are described in detail below.

1) mHEP: Multi-Level Heterogeneous Computing Platform: The *first level heterogeneous computing platform (1stHEP)* is the computing platform on the vehicle that contains heterogeneous hardware. This level is the main computing resource of VCU. In addition to CPU, 1stHEP leverages GPU, FPGA, and ASIC to match and accelerate the services on the vehicle. Due to the excellence in paralleled computation that benefits from thousands of hard floating-point units, GPU is becoming widely used for accelerating complex machine learning or computer vision based applications [20]. FPGA can be dynamically reconfigured, so it is suitable for various algorithms, including but not limited to machine learning [21], [22]. FPGA will perform the tasks like feature extraction, and data compression and media coding and decoding, etc. The 1stHEP will use some ASICs to accelerate specific algorithms, because they have best performance and energy-efficiency.

The storage and communication modules are also deployed on the 1stHEP. In order to achieve better I/O performance, the parallelism-supported solid state drive (SSD) [23], [24] is chosen to store vehicle data and applications. Regarding the communication module, 1stHEP also contains several communication modules, such as 3G/4G/5G, Wi-Fi/BLE and DSRC. The 3G/4G/5G modules enable the vehicles to communicate with the cloud server directly. Vehicles are able to join the Internet of Vehicles through the 5G/DSRC module to cooperate or data share with other vehicles or XEdge. And the short-range Bluetooth interface will allow the vehicle to connect with the passengers' mobile devices.

Additionally, VCU also contains a *second level heterogeneous computing platform (2ndHEP)* that tries to exploit all other possible on-board computing resources, such as the passengers' mobile devices and vehicle on-board controller to alleviate the computation burden on the 1stHEP.

2) DSF: Dynamic Scheduling Framework: As mentioned in Section II, the upper applications on the vehicle are often computational expensive. Meanwhile, they may depend on different kinds of computation tasks, such as data prepossessing, model training/refining and decision prediction, each requiring different compute resources. We can assume the upper computation tasks are heterogeneous as well. Note that these tasks are not fixed over time, as do the underlying available computing resource. To optimally utilize the

hardware resource, the primary goal of DSF is to provide an effective scheduling framework to dynamically assign each task/sub-task to the best *fit* processor. In particular, DSF has the following two functions:

Computing resources collection: DSF provides dynamic management of computing resources of mHEP. First, DSF allows computing resources to join and exit dynamically, which is used to manage the 2ndHEP and some plug-and-play computing resources. Second, DSF acquires the real-time status of all computing resources periodically. For example, utilization rate and task type at processing. These dynamic status and static information (computing ability and matched task type) of computing resources are taken as their profiles. Third, the profiles are important information for DSF to dynamic allocate computing resources to applications. And resources accessed by applications are tightly controlled by DSF, which will achieve resources isolation and reduce the interference among the applications. DSF also provides the access interfaces of all computing resources, which we called *control knob*. The upper system or applications can access the computing resources via the control knob.

Task scheduling: DSF divides the original applications into some sub-tasks by fine-grained and tries to match the tasks with the computing resources according to their computing characteristics. DSF determines the resources type and amounts which will be allocated to each task according to the dynamic status of each resource, QoS requirement and processing priority of each task, and the cost of each scheduling plan. After the tasks are distributed to specified computing resources, DSF will reduce the results of each task and return it to the upper operating system or application.

Note that VCU could be viewed as complementary to the traditional vehicle on-board controller by offering more computing, storage and communication capabilities for modern vehicle applications. The resources on VCU can be accessed by users, auto vendors and third-party developers via the open interfaces. Moreover, the hardware resources could be easily augmented through the extensible interfaces (USB/PCIe).

The main design question for VCU is how to build an efficient computing platform for a vehicle that supports a wide array of applications on vehicle scenario. This is challenging for various reasons. First, VCU will integrate heterogeneous hardware (CPU, GPU, FPGA and ASIC) on a single board. And, the board will contain several communication modules to make the vehicle connected to cloud, XEdge, other vehicles, and user devices. Second, VCU needs to provide dynamic management to resources and collect the real-time status of resources which will ensure the proper resource allocation for tasks. Finally, because all resource allocations and task distributions depend on the scheduling algorithm in VCU, the algorithm should consider more possible factors to make the best scheduling plan. Meanwhile, the complexity and overhead of the scheduling algorithm should be considered.

C. $EdgeOS_v$: An Edge Operating System for Vehicles

$EdgeOS_v$ is an edge operating system for the CAVs, consisting of *Elastic Management*, *Data Sharing*, *Security*, and *Privacy*. Four fundamental features (DEIR) of service quality are proposed in [6], [25] for Edge Computing, which also should be inherited in the design of $EdgeOS_v$. The DEIR refers to *Differentiation*, *Extensibility*, *Isolation* and *Reliability*. *Differentiation* is achieved by the *Elastic Management* module and *Isolation* by the *Security* module. *Reliability* is supported together by two modules, *Elastic Management* and *Security*. It is worth noting that the *Extensibility* property is two-fold: hardware-level and software-level. The hardware level of extensibility is achieved by VCU, which can extend the computational devices using mobile devices, while the software level of extensibility is achieved by an application library called `libvdap` discussed in Section IV-E.

The *Elastic Management* module is used to manage all services on the vehicle. The *Data Sharing* module provides the data exchange and sharing between different services. And the *Security* module is used to provide a trusted execution environment and isolation scheme for security-sensitive services. The *Privacy* module provides some privacy protection schemes for data sharing between vehicles and XEdge, as well as the cloud. Before we introduce these four modules, we first introduce the conception of *Polymorphic service*, and then introduce how the *Elastic Management* module manages these services. Then, we will introduce *Data Sharing*, *Security*, and *Privacy* modules.

In $EdgeOS_v$, each service offers multiple execution pipelines in response to various network and computational constraints. Take the third-party application searching for a kidnapper using mobile version A3, as an example. The vehicle can automatically pinpoint the targeted vehicle by recognizing the license plate number. This service can be accomplished in three pipelines: 1) all workloads execute on board; 2) all workloads execute on the edge/cloud server, and 3) the detecting motion is put on board while recognizing license plate is deployed on the edge/cloud server. Here, the first one needs more local computational resources than the second, but with less bandwidth requirement and low network latency.

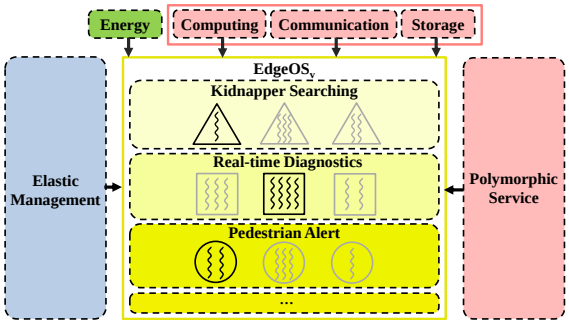

Fig. 6. The overview of *Elastic Management*.

The *Elastic Management* module is illustrated in Figure 6. The *Elastic Management* module can dynamically choose an optimal pipeline for each *Polymorphic Service*, considering

priority, required response time and polymorphic requirements for computational resource and network quality (*i.e.,* latency, bandwidth and connectivity). Let's take the kidnapper searching as an example, if the network quality (*i.e.,* remainder bandwidth by other applications) is enough for uploading video data the nearby XEdge, the *Elastic Management* module might offload all workloads to the XEdge, especially when there is a lack of on-board computational resources. Once all of bandwidth are occupied by other services with higher priority (*e.g.,* autonomous driving services), it can adjust the pipeline to make part of workload being processed on board, e.g., detecting motion. If the network quality and computation resources cannot support this service, the service will be hung up until meeting requirements again. As the dynamic adjustment, the service quality and user experience will be optimized.

To address the security issues of CAVs, EdgeOS$_v$also contains a *Security* module, which relies on the trusted execution environment (TEE) technique. The major benefits of using TEE can ensure all services running on top be securely isolated via encryption of their corresponding memory space. For other non-TEE supported services, the containerization, compared with the virtualization technology, is a good candidate for isolation and migration due to the light weight of a container [26]. Note that the containerization also can enhance the isolation of a TEE [27]. Moreover, the *Security* module monitors services and prevents them from compromising. Once the service is compromised , this module will remove the compromised one and re-install an initialized one without compromising, which implements the part of function of *Reliability*.

As mentioned before, the data will be shared between vehicles as well as services. For example, both of the pedestrian detection service for autonomous driving and the mobile A3 service need to access real-time camera data, and the mobile A3 service will share the result with a vehicle recorder service, which records all surrounding vehicle information for future vehicle searching task. Thus, in our TEE enhanced EdgeOS$_v$, the *Data Sharing* module provides a mechanism for data sharing between different services with a high security, which will authenticate the service and perform fine grain access control. To protect the privacy of data sharing between vehicles, some identity privacy protection schemes will be provided by the *Privacy* module. For example, the vehicle can use the pseudonym, generated and periodically updated by the *Privacy* module, for privacy protection in data sharing.

Open Problems: Here are several open problems that need to be addressed for EdgeOS$_v$. Zhang *et al.* [18] and Kang *et al.* [28] have demonstrated that dividing a workload into several parts and making them execute on different edge nodes along the path from the source to the cloud can get a better response latency and data transmission. However, how to dynamical divide workload on the edges is still a problem. And in our EdgeOS$_v$, it requires knowing the network quality to other edge nodes, which has not been well solved.

D. DDI: Driving Data Integrator

Besides the data collected by sensors embedded in different components of vehicles, such as engine status, tire pressure, battery status, vehicle speed, and so on, we think the context data of driving, such as road condition, weather, traffic information, also plays an important role in decision making, *e.g.,* assistant driving, battery cell management, remote diagnostics, abnormal driving behavior detection and so on. Therefore, we propose to include a driving data integrator (a.k.a DDI) in *OpenVDAP*. The main objective of DDI is to automatically collect and store relevant context information on the vehicle and to serve high level services via a set of APIs.

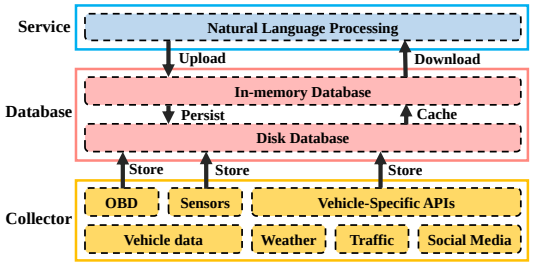

Fig. 7. The design of DDI.

As shown in Figure 7, the DDI consists of three layers. The bottom layer is the data collector layer. The middle layer is the database and the top layer is the service layer. The data of DDI consists of four aspects: vehicle driving data, weather information, traffic condition, as well as social web information like some emergencies. OBD reader and on-board sensors collect the driving data, which includes the location, speed, acceleration, angular velocity and so on. Noted that, we used an OBD reader since most of the normal vehicles only provide an OBD interface to obtain driving data, and in the future, we will adapter this to more types of vehicles by multifold devices, such as CAN card for electric vehicles. Weather, traffic and social data are collected from vehicle-specific APIs. Environment information gives the weather conditions and real-time traffic conditions, which can be a necessary part for detecting the abnormal driving behavior. Social web information is used to help keep the vehicle away from troubles. In this way, for a specific person driving through a period of time, DDI can provide the driving information, real time weather, real time traffic conditions as well as emergencies that have happened nearby.

The database layer is composed of two types of database, in-memory database, e.g., Redis [29], and disk database, e.g., MySQL [30]. As the data from the collector layer is time-space related, disk database is utilized to store it. Meanwhile, in-memory database caches the frequently used data from disk database to decrease the response latency of request. For all the data caches into the in-memory database, a survival time is set for it. Therefore, all the request for the data would search the in-memory database first, when it can't be found in in-memory database, it would go to the disk database. All the

related data includes location and timestamp. Collected data are permanently stored in the disk database.

The service layer takes charge of requests from the upper layer like `libvdap` via a set of APIs. The requests include two types: download requests and upload requests. An upload request is for users to upload their data onto the DDI while a download request is for the user to download data from DDI. For a download request, the service layer extracts keywords like location and timestamp, then it goes to in-memory or disk database to get data. For an upload request, firstly the data would be stored in in-memory database, when the survival time is up and the related charts have been created in disk database, the data in in-memory database would be written to disk database for data persistence.

Open Problems: Here are some open problems that need to addressed in the design and implementation of DDI. First, what are the right mechanisms to get real-time data, *e.g.,* traffic condition, social web events? It is hard to obtain real-time data with an unreliable, limited bandwidth communication channel available. Second, how to efficiently store and manage time-space related data on a vehicle is challenging. For example, how long will these data need to be stored is still unclear.

E. `libvdap`: Library for Open Vehicular Data Analytics

With the burgeoning of AI-based applications developed for CAVs, *OpenVDAP* should support a variety of artificial intelligence algorithms and models. However, the edge node is not a good fit for executing large scale models since they require large footprints on both the storage and computing power. To support edge intelligence, `libvdap` is provided which stores many AI common algorithms and models. These models are compressed based on the powerful models and can run smoothly on the edge node. Based on `libvdap`, developers can build AI applications more openly and easily.

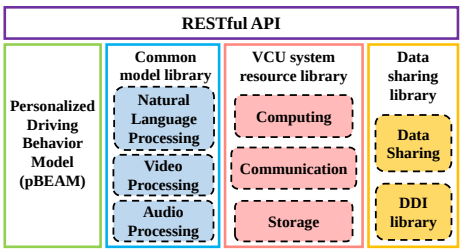

Fig. 8. The overview of `libvdap`.

As shown in Figure 8, `libvdap` provides a uniform RESTful API. By calling the API, developers can access all software and hardware resources. The resources can be grouped into four categories: Personalized Driving Behavior Model (pBEAM), Common model library (cBEAM), VCU system resources library, and Data sharing library. For example, developers can use the real-time or history data acquired by the Data sharing library, leverage the pBEAM or the AI models provided by the Common model library to build their application. The application will run on the *OpenVDAP* by calling the VCU System resources library. In this way, developers can create applications that provide value in the easiest possible way.

Personalized Driving Behavior Model (pBEAM), the core component of `libvdap`. pBEAM models personalized driving behaviors based on driving data. It has been built and deployed on vehicles. Developers can build third-party applications on top of pBEAM. By leveraging pBEAM, developers can acquire the driver's behavior without a long-running process of collecting and analyzing data. For example, the insurance company can evaluate whether the driver is aggressive or not based on pBEAM.

As shown in Figure 9, pBEAM is pre-trained on the cloud server and trained again on the XEdge (*OpenVDAP*) to obtain the personalized characteristics. On the cloud server, we build a *Common Driving Behavior Model (cBEAM)* based on a large training dataset which includes many drivers' driving data. The input data includes the location, speed, acceleration, and so on. Although cBEAM is powerful, it's scale is large. So it requires large footprint on both the storage and compute power, which is impractical for the XEdge. To save storage and computing resource, a compression algorithm based on Deep Compression [31], [32] is used, in which cBEAM is pruned first to reduce the number of connections by learning only the important connections, then the number of bits for representing each weight is reduced via the weight sharing technique. The compressed cBEAM is then downloaded to the vehicle. Transfer learning [33] is used to transfer the compressed cBEAM to pBEAM by learning the personalized driving data which stores in the DDI. When pBEAM is already trained, it can provide service for third-party applications.

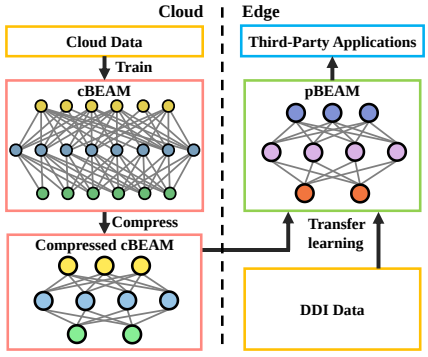

Fig. 9. The building process of pBEAM.

The *common model library* contains many common algorithms and models that are used frequently in vehicle-based applications, such as Natural Language Processing, Video Processing, Audio Processing and so on. The most powerful models that we leverage today are too large for the *OpenVDAP* to run, so the models that are in the Common model library are compressed based on the powerful models. After compressing, the models are optimized based on the computing power of VCU. The *VCU system resources library* provides developers with the library of system resource. By calling the library, users can access the computing, storage and communication

resources. The *data sharing library* provides to the user the uniform data interface. The data comes from two sources: Data Sharing module of EdgeOS$_v$and DDI.

Open Problems: Several open problems that need to be addressed for `libvdap` include: Firstly, although we compressed the large-scale artificial intelligence models in the cloud, they are still too large to leverage on the XEdge. Can we design a model which has a smaller size based on personalized dataset? Secondly, how can we consider more context information when build pBEAM, such as driver's age group? At last, the driver may show different behavioral characteristics due to the different kinds of weather. How can we take the environment into consideration when building the pBEAM?

V. Related Work

The concept of connected vehicles has been proposed since 1996, and there are many researchers working on this area. Feng *et al.* [34] proposed a real-time adaptive signal control approach using connected vehicles. Wang *et al.* [35] proposed a real-time path planning approach to relieve traffic congestion in urban scenarios. All of the works are promising to improve the safety and convenience of driving. All autonomous vehicle are connected vehicle, which allows the vehicle connects to the Internet and share internet access with other devices both inside as well as outside the vehicle [36]. To improve the user experience, auto vendors have also developed infotainment systems, such as BMW's iDrive, Audi's Audi connect, General Motors' OnStar Chrysler's Uconnect, Honda's Blue Link, and so on. Beside auto vendors, Google presented the Android Auto [37] in 2015, a mobile app that allows enhanced use of an Android device within a vehicle equipped with a compatible head unit. By this app, the Android device can broadcast apps onto the vehicle's display, such as the map app. Apple provided a similar system, CarPlay [38], in 2014. Baidu and Alibaba also presented similar systems by modifying their existing systems, DuerOS [39] and AliOS [40]. All four of these systems provide a hands-free operation through voice commands to minimize driving distraction.

Now, there are many companies, including traditional auto makers and technology companies, working on the autonomous vehicle. Here, we only list some of them. Ford declares that a fully autonomous vehicle (SAE Level 4) will be possible in commercial operation in 2021 [11]. General Motors has prepared the first production-ready car with no steering wheel or pedals, which means it is an autonomous vehicle, and General Motors is asking DOT permission to safely deploy autonomous vehicles in 2019 [12]. Renault-Nissan alliance plans to build an autonomous vehicle on the road by 2020 [41]–[43]. Bosch and Daimler have entered into a development agreement to bring fully automated (SAE Level 4) and driverless (SAE Level 5) driving to urban roads by the beginning of the next decade [44]. The Volkswagen Group introduce its first SAE Level 5 fully autonomous concept vehicle, Sedric in 2017 [45]. The BMW has been testing autonomous vehicles on public roads for several years, and the goal of reaching the consumer market is 2021 [46]. The Tesla has built its partial autonomous vehicles on the road.

Besides traditional auto vendors, many technology companies are also working on the autonomous vehicle, but most of them only focus on software platform or algorithms, and collaboration with traditional vendors for autonomous vehicle commercial productions. Google started to research autonomous vehicle in 2009, and in 2016, it became a new company, Waymo [1]. In 2015, the world's first fully self-driving ride on public roads is done by Waymo without a physical steering wheel, pedals and driver. In 2017, its technology has been added to the Chrysler Pacifica Hybrid minivan, a mass-production platform. Baidu started to research autonomous vehicle in 2013 and had its first road testing in 2015. In 2017, its platform Apollo is open [2], including an open autonomous driving software platform, suggested hardware platform, and some real data for simulations. Using these data, many researchers can train, improve and verify their algorithms.Apple also studies autonomous driving and, at the end of 2017, it published the first paper [47] related to autonomous driving, in which the authors proposed VoxelNet to detect the 3D object from the lidar's data. The Nvidia, as a chip maker, also studies autonomous driving, and it also provides computing units for autonomous driving, such as Nvidia Drive PX [17]. There are also some startup companies working on autonomous driving, such as Drive.ai, Zoox, Pony.ai and NIO.

VI. Conclusions

In this paper, we proposed a security, privacy-aware, and isolation-supported computing architecture for the future CAVs that is envisioned as a sophisticated computer on wheels with a variety of computation-intensive yet latency-sensitive services running on top. In particular, we identified three possible computing architectures for these on-board services and pointed out their challenges and limitations in ensuring these services be accomplished at the *right* time. Inspired by the promising of edge computing in dealing with latency-sensitive applications, we therefore proposed an edge computing based platform (*i.e., OpenVDAP*), which is a full-stack hardware/software platform that consists of an on-board heterogeneous computing units, an isolation-supported and security/privacy-aware vehicle-dedicated operating system, as well as an edge-aware application library. In addition to supporting on-board computing, *OpenVDAP* is also featured as a two-tier computing architecture via proposing a series of systematic mechanisms that enable CAVs to dynamically detect service status and to identify the optimal offloading destination (*e.g.,* in-vehicle, XEdge, or cloud) so that each service could be finished at the *right* time. Most importantly, unlike the proprietary platforms, *OpenVDAP* offers an open and free edge-aware library that contains how to access and deploy edge-computing based vehicle applications and various common used AI models, which will enable the researchers and developers in the community to deploy, test, and validate their applications in the real environment.

REFERENCES

[1] (2017, Dec.) Waymo. [Online]. Available: https://waymo.com/

[2] (2017, Dec.) Apollo. [Online]. Available: http://apollo.auto/index.html

[3] (2017, Dec.) The world's first self-driving ubers are on the road in the steel city. [Online]. Available: https://www.uber.com/cities/pittsburgh/self-driving-ubers/

[4] S. Liu, J. Tang, Z. Zhang, and J. L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[5] P. Nelson. (2016) Just one autonomous car will use 4,000 gb of data/day. [Online]. Available: http://www.networkworld.com/article/3147892/internet/one-autonomous-car-will-use-4000-gb-of-dataday.html

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

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

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16.

[9] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, Apr 2015.

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

[11] (2018, Jan.) Ford will have a fully autonomous vehicle in operation by 2021. [Online]. Available: https://corporate.ford.com/innovation/autonomous-2021.html

[12] (2018, Jan.) Meet the cruise av: the first production-ready car with no steering wheel or pedals. [Online]. Available: http://media.gm.com/media/us/en/gm/home.detail.html/content/Pages/news/us/en/2018/jan/0112-cruise-av.html

[13] S. international. (2016) Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. [Online]. Available: http://standards.sae.org/j3016_201609/

[14] G. Kar, S. Jain, M. Gruteser, J. Chen, F. Bai, and R. Govindan, "Predriveid: Pre-trip driver identification from in-vehicle data," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17. New York, NY, USA: ACM, 2017, pp. 2:1–2:12. [Online]. Available: http://doi.acm.org/10.1145/3132211.3134462

[15] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Enhancing amber alert using collaborative edges: Poster," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17, 2017, pp. 27:1–27:2.

[16] (2018, Jan.) Intel movidius neural compute stick. [Online]. Available: https://developer.movidius.com/

[17] (2018, Jan.) Autonomous car development platform from nvidia drive px2. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-px/

[18] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *Technical Report MIST-TR-2017-002*, 2017.

[19] Z. Ning, F. Zhang, W. Shi, and W. Shi, "Position paper: Challenges towards securing hardware-assisted execution environments," in *Proceedings of the Hardware and Architectural Support for Security and Privacy*, ser. HASP '17. New York, NY, USA: ACM, 2017, pp. 6:1–6:8. [Online]. Available: http://doi.acm.org/10.1145/3092627.3092633

[20] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots hpc systems," in *International Conference on Machine Learning*, 2013, pp. 1337–1345.

[21] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *International Symposium on Computer Architecture*, 2010, pp. 247–257.

[22] G. J. García, C. A. Jara, J. Pomares, A. Alabdo, L. M. Poggi, and F. Torres, "A survey on fpga-based sensor systems: towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing," *Sensors*, vol. 14, no. 4, pp. 6247–6278, 2014.

[23] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for ssd performance." in *Usenix Technical Conference, Boston*, 2008, pp. 57–70.

[24] D. Nellans, D. Nellans, and P. Bonnet, "Linux block io: introducing multi-queue ssd access on multi-core systems," in *International Systems and Storage Conference*, 2013, p. 22.

[25] J. Cao, L. Xu, R. Abdallah, and W. Shi, "EdgeOS_H: A home operating system for internet of everything," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 1756–1764.

[26] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support paas," in *2014 IEEE International Conference on Cloud Engineering*, March 2014, pp. 610–614.

[27] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer, "SCONE: Secure linux containers with intel SGX," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 689–703.

[28] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17, 2017, pp. 615–629.

[29] (2018, Jan.) Redis. [Online]. Available: https://redis.io/

[30] (2018, Jan.) Mysql. [Online]. Available: https://www.mysql.com/

[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[32] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 243–254.

[33] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[34] "A real-time adaptive signal control in a connected vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 460 – 473, 2015, engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue.

[35] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, and F. Bai, "Real-time path planning based on hybrid-vanet-enhanced transportation system," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 5, pp. 1664–1678, May 2015.

[36] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, Aug 2014.

[37] (2018, Jan.) Android auto. [Online]. Available: https://www.android.com/auto/

[38] (2018, Jan.) Apple carplay - the ultimate copilot. [Online]. Available: https://www.apple.com/ios/carplay/

[39] (2018, Jan.) Dueros for smart automobile unit solution. [Online]. Available: https://dueros.baidu.com/en/html/2017/jjfasj_0608/10.html

[40] (2018, Jan.) Ford and alibaba explore strategic collaboration to reimagine vehicle ownership experience, expand mobility services. [Online]. Available: https://media.ford.com/content/fordmedia/fna/us/en/news/2017/12/07/ford-and-alibaba.html

[41] (2018, Jan.) Renault-nissan plans driverless ride-hailing and ride-sharing services. [Online]. Available: http://www.autonews.com/article/20170622/COPY01/306229943/renault-nissan-plans-driverless-ride-hailing-and-ride-sharing-services

[42] (2018, Jan.) Autonomous vehicles. [Online]. Available: https://group.renault.com/en/innovation-2/autonomous-vehicle/

[43] (2018, Jan.) Nissan's self-driving car. [Online]. Available: https://www.nissanusa.com/blog/autonomous-drive-car

[44] (2018, Jan.) Future mobility: Bosch and daimler join forces to work on fully automated, driverless system. [Online]. Available: http://media.daimler.com/marsMediaSite/doc/en/16390030

[45] (2018, Jan.) Individual mobility redefined: Autonomous driving at the touch of a button. [Online]. Available: https://www.volkswagenag.com/en/news/2017/03/Autonomous_driving.html

[46] (2018, Jan.) Autonomous driving – 5 steps to the self-driving car. [Online]. Available: https://www.bmw.com/en/automotive-life/autonomous-driving.html

[47] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396