Energy-Optimized Elastic Application Distribution for Automotive Systems in Hybrid Cloud Architectures

Philipp Weber
Technical University of Munich
Munich, Germany
philipp.weber@tum.de

Philipp Weiss
Technical University of Munich
Munich, Germany
philipp.weiss@tum.de

Dominik Reinhardt
Corporate Research
BMW Group
Munich, Germany
dominik.reinhardt@bmw.de

Sebastian Steinhorst
Technical University of Munich
Munich, Germany
sebastian.steinhorst@tum.de

***Abstract*—The increase of resource-intensive applications in modern vehicles used for streaming, gaming or autonomous driving results in rising energy-consumption of its advanced computing and connectivity hardware. Especially in electric vehicles, this leads to much higher hardware costs and a decreased vehicle range. Modern premium cars use distributed heterogeneous hardware and mostly communicate via APIs to large cloud backends. Current approaches to reduce on-board energy consumption offload applications partly and make use of limited network connectivity assumptions to their backends. In this paper, we propose a hybrid electric and electronic architecture that manages vehicle hardware by using cloud computing frameworks. Our hybrid cloud architecture is a connection of the local vehicle cloud and a large data centre community cloud. We propose an online optimization algorithm that shifts applications from on-board ECUs to data centre servers and vice-versa. The optimization algorithm minimizes the local energy-consumption while satisfying predicatively dynamic constraints like data rate limitations, application policies and resource limitations. Our approach outperforms the non-predictive approach in average by 16%, in the best case by 21% and in the worst case both behave equally well.**

***Index Terms*—Hybrid Cloud Computing, Elastic Application Reallocation, Energy Optimization**

I. Introduction

The automotive industry is confronted with new challenges to fulfil future customer expectations within the upcoming decade. Autonomous driving, highly interconnected cars with different ecosystem (IoT) and always changing requirements for infotainment systems are only a subset of already known topics. In parallel, political and environmental restrictions impede future innovations for cars [4]. Therefore, the need for computation power is significantly increasing by the rising amount of software within future cars.

Thus, the amount of electronic control units (ECUs) is increased by every vehicle generation and E/E (electric and electronic) architectures are getting bigger and more complex [4, 18]. To tackle that trend, Original Equipment Manufacturers try to centralize software (see Figure 1, Computing Platform) on capable computing platforms or vehicle servers [3, 17]. These servers are built up monolithically and typically integrate strong CPUs and adequate memory, similar to consumer electronic devices.

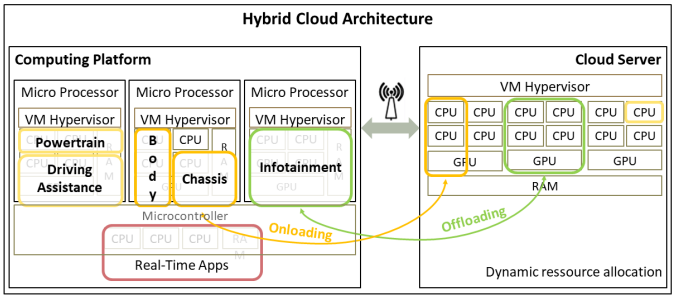

Figure 1: Our idea shows a hybrid cloud architecture as Car-as-a-Cloud. By connecting vehicle and data centre Infrastructure-as-a-Service approaches, we can provide elastic applications using location independent resources.

The rising energy consumption of on-board ECUs will have a significant influence on the maximum possible driving range of electric cars. For future E/E architectures an additional power consumption of up to 4 kW is assumed [9]. With a battery capacity of 42.2 kWh for the BMW i3, on-board high performance computing and electrical vehicles are not practical. A 1 kW Powernet-Load increases CO2-Emission by more than 30 g/km. Therefore, efficient use of on-board resources is essential for future automotive systems and will be discussed in this paper.

Due to safety and security concerns, up to now some software parts must be partitioned on problem specific microcontrollers to fulfil Automotive Safety Integrity Level (ASIL) requirements up to ASIL-D. It is common practice to integrate all remaining software parts on a general-purpose CPU within each single computing platform [5, 16]. Vehicle software with stricter real-time requirements remains on appropriate

With the support of the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763.

microcontrollers. It is tried to gradually shift even that software with minor changes to performance units.

However, this approach is restricted to available qualified automotive semiconductors and their current maximum performance capabilities on the market. The already mentioned requirements for power consumption and required computation power are hard to achieve locally within a single ECU and over a lifetime of 25 years [2]. Therefore, another strategy to offload computation power into data centres is necessary. Software companies which are working on mobile devices (e.g. Google, Microsoft) are using more and more computation power of their data centres inside these devices [7]. Compared to a car, several ECUs are already interconnected to each other and software relocation between ECUs that serve on-demand computation requirements are needed ('Car-as-a-Cloud' - CaaC). Current solutions split up applications into modules, in order to reduce the data size which must be transferred during that process. This leads to the need of redesigning all applications and to a communication overhead between their offloaded and on-board modules.

To cope with the rising number of complex ECUs inside vehicles and a more and more challenging expectant attitude towards in-vehicle applications, we need to rethink their computing, network and storage resource management. When offloading applications, the most efficient computing resource in terms of energy consumption must be determined. Due to unstable network connections to the offloading destinations, we need to consider dynamic conditions, under which the optimum location for each application is constantly changing.

By proposing and building a hybrid cloud architecture as Car-as-a-Cloud which connects multiple vehicle ECUs with multiple data centre servers, we make the following distributions:

- We firstly categorize and give insights into current cloud-based vehicle functions in Section II. We analyse current offloading algorithms and energy-consumption saving aspects.
- Based on our System Model in Section III, we use Infrastructure-as-a-Service approaches which allow applications to run on on-board ECUs or data centre servers. Our presented system architecture manages these on-board ECUs like regular cloud resources.
- A predictive broadband channel algorithm, which is able to predict the network quality for a time and location, is used in Section IV for our predictive distribution algorithm. Instead of using look-up tables, we use an optimization algorithm to dynamically shift applications under changing broadband channel prediction.
- We provide a case study in Section V to explain aspects of our predictive application offloading algorithm.
- We present our test bench set-up and evaluate our approach in Section VI. With our scalability analysis we analyse multiple test application scenarios. Our approach outperforms the non-predictive approach in the best case by 21%, on average by 16% and in the worst case both algorithms have equal energy consumptions.

II. Related Work

In this section, we classify our architecture in the context of cloud computing and definitions for cloud-based vehicle functions. Furthermore, we analyse current dynamic cloud offloading approaches, features and leading concepts. Last, we discuss approaches to minimize on-board energy consumption.

A. Classification of cloud-based vehicle functions

Modern vehicle application architectures are not restricted to on-board hardware anymore. The authors of [6] define applications which run instead of on-board ECUs at least partly in a cloud as "Cloud-based Vehicle Functions". When these applications aren't restricted to on-board resources, then they can provide new features that were unfeasible before [6]. Real-time and safety-critical functions still have to remain on-board and have to be treated separately.

The authors of [6] are defining four categories of Mobile Cloud Computing (MCC) application areas. These four categories are extended and adapted to the vehicular environment by [11]. The classified domains are:

A Cloud Only: Applications run only in a data centre
B Fall-Back Method: Applications run as backup in a data centre
C Duplicate Function: Applications could run in lock-step or as double-check in a data centre
D Elastic Application: Applications can run on-board or in a data centre

According to the classification system of [11], we use the domain of Elastic Applications, which is the most flexible and sophisticated approach, to describe the architecture presented in this paper. In this category, applications can run seamlessly either in data centres or on-board ECUs.

B. Dynamic cloud offloading approaches

In the domain of dynamic cloud offloading, we analyse and explain state-of-the-art algorithms and system models for shifting applications, modules or code between cloud and mobile device.

[22] is describing a context sensitive offloading scheme for general mobile devices. Based on falling distinctive, the context-awareness switches between different wireless connections like Wi-Fi or 5G and between best cloud resource to use. These resources can be mobile ad-hoc networks, cloudlets or public clouds. In contrast to our client-server communication model, [22] only allows the mobile device to be client and the cloud resource to be server.

[1, 2] and [20] split up applications in smaller modules, tasks or code. On one hand this allows specific resource-intensive functions to be off-loaded. On the other hand, the application design must be modified and modular, while creating more interfaces and a communication overhead. Both, [2] and [22] use application and network profiler in order to classify application characteristics and track the current network connectivity state in their system design.

Furthermore, [2] presents a heuristic mechanism for scheduling and placing modules on a cloud server or an on-board unit. The mechanism is set up of three stages. The first stage does proactive placement to meet current network data rate limitations and deadlines. In the next step, the placement is refined to fulfil a set of predefined policies. During the last step, the schedule for the application placement is executed.

To the best of our knowledge there is no contribution to predictive cloud offloading, where predictive connectivity estimations are influencing online distribution decisions.

C. Energy consumption of vehicle functions

In the following, we analyse different approaches to quantify the energy calculations for cloud offloading. [12] and [20] present a mathematical model for energy consumption which describes functions running on on-board vs cloud-based. When running cloud-based, an additional communication overhead is generated which leads to higher energy consumption that has to be considered in energy analysis according to [20].

A decision manager is presented in [8] and [12]. The latter one can be used in a predictive manner in context of a car ride or online during this ride. This approach calculates the location of functions (on-board or data centre) upfront, depending on the vehicle position. Therefor the data-rate along the route is needed, but there is no information about the time-variation of the data-rate. [15, 19] proves that all wireless broadband channels are time-variant and are subject to many factors like number of clients or weather conditions.

The authors of [20] prioritize economic factors (costs and energy savings) over possible time savings by on-demand cloud resources. The time savings can be achieved by faster computation on specialized hardware.

III. System Architecture

We model our system as a hybrid cloud architecture with a wireless network connection between the local cloud (on-board ECU managed as a cluster) and the global cloud (multiple data centres), which will be presented in the following.

The hybrid cloud is a connection between a private cloud and a community cloud. Both clouds are connected over a wireless broadband peer-to-peer connection. The private cloud is a local distributed cloud running on different embedded systems. It is based on heterogeneous low-power platforms, with limited computing power, storage and networking capabilities. The name local cloud describes its operating location inside a vehicle. The community cloud is a global centralized server system running in data centres. Due to its location in data centres, we define the community cloud as the global cloud. In our test bench implementation in Section VI, we use Openstack and its features as our cloud management system.

A. Local Cloud

The local cloud is a cluster of embedded systems. It is organized like data centre servers to provide the same characteristics like flexibility, upgradeability, hardware independence etc. Each node of the local cloud is connected to a high-throughput network, which in turn is connected via a mobile radio network to a data centre. The different computing nodes are heterogeneous and have special features e.g. real-time capabilities, dedicated GPU hardware, high-capacity storage or different hardware virtualization drivers. Our predictive distribution algorithm from Section IV is running on the local cloud.

B. Global Cloud

The global cloud is a cluster of high-performance data centre servers. It differs from the local cloud in multiple factors:

- It has highly specialized hardware like high-performance GPUs, Tensor Processing Units (TPU) or high-speed and high-bandwidth storage, which allows faster execution or running resource-intensive applications.
- It provides a low latency and high data rate connection to the Internet and IoT systems, when running inside data centres [6].
- If other OEM applications and databases are running in the same or nearby data centre, then low latency, high data rate and unrestricted connection to protected fleet data can be provided. An example is traffic jam or road surface quality data, which is tracked by fleet vehicles and sent to the OEM application backend.
- The limiting factor is the restricted connection and access to sensors and actuators inside the vehicle. The data has to be transmitted over a wireless and non-deterministic connection from the vehicle to the data centre and vice-versa.

C. Applications

Our system consists of multiple modules (Connectivity Monitor, Application Manager, Optimization Algorithm, Control, Database and User Input) to monitor and distribute applications between the global and local cloud. All applications are distributed dynamically on different computing nodes. These computing nodes have different characteristics as the hardware varies between the local and global cloud. The applications can be real-time tasks, vehicle functions or multimedia entertainment programs.

IV. Predictive Distribution Algorithm

We present a block diagram of our predictive distribution algorithm in Figure 2, which is running on the local cloud. The *Connectivity Monitor* consists of current network sensing and network channel forecast. It gathers current and future connectivity parameters as input for the *Optimization Algorithm*. The current parameter values are measured actively. For our approach, the connectivity parameters are limited to data rate and latency. The *Optimization Algorithm* optimizes the placement of the applications on different compute nodes, using the predictive broadband channel parameters. The network channel prediction is explained in detail in Section IV-A.

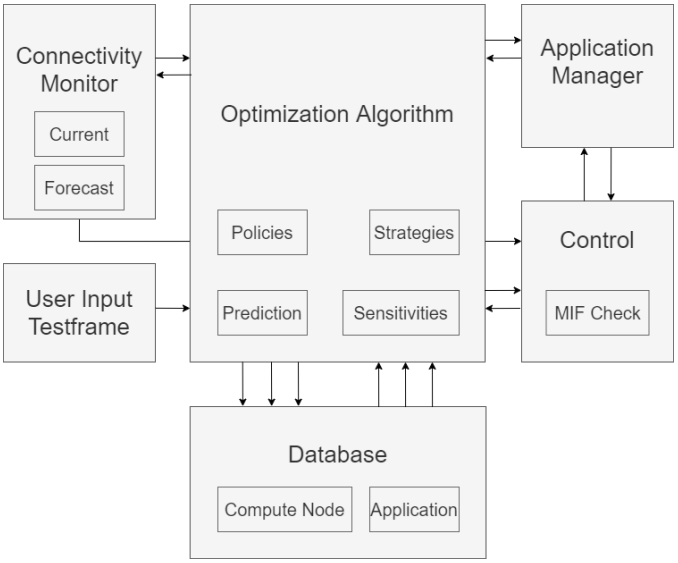

Figure 2: The block diagram from the presented software architecture of our predictive distribution algorithm is separated into different modules.

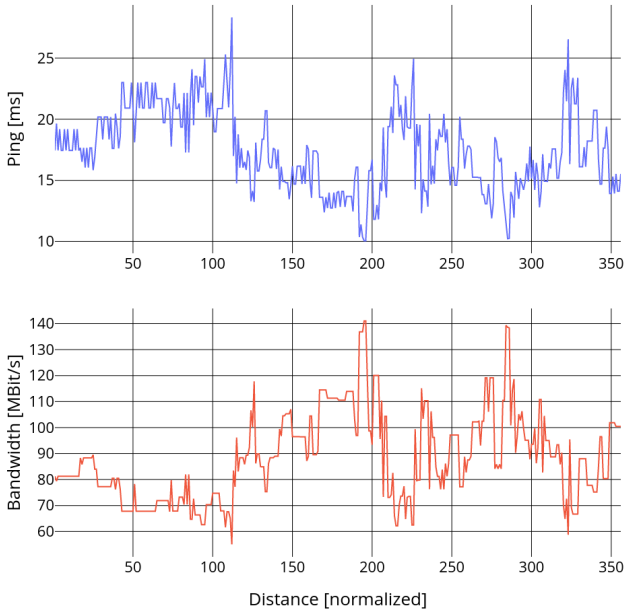

Figure 3: The network data rate and ping are plotted over GPS positions. We normalized the GPS positions as the distance from the starting position. The prediction is from November 19th 2019 and the route is in the area of Munich. The shown values are a randomized copy of the original data with same dynamic and absolute properties.

Characteristics and profiles of all compute nodes and applications are saved in the *Database*. This data is accessed by the optimization algorithm during its execution. When the optimization algorithm calculates a new optimum distribution, which differs from the previous optimum distribution, applications have to be shifted.

All operative tasks that are needed for this shifting like starting, stopping or migrating applications, are realized in the *Application Manager*. The module is sensing current application profiles such as CPU usage, RAM and data rate consumption. The responsibility of the *Control* module is to guarantee operations of the most important functions and override commands from the *Optimization Algorithm* to the *Application Manager*, if necessary. It also uses current network channel parameters from the *Connectivity Monitor*. The importance of each application is saved inside its profile in the *Database*.

The *User Input Testframe* has two major tasks. Firstly, it is used to add applications to the system or to remove them if they are not required anymore. Secondly, it operates as a test framework to run simulation inputs for the system model.

A. Network Channel Prediction

A moving vehicle succumbs changing wireless network conditions. These conditions are not only time- and location dependent, but also rely on other clients in the same network cell, on the geometry of the surrounding and climate conditions. A constant network quality is not realistic [15, 19]. To tackle this problem, telecommunication companies try to predict short-term future channel parameters.

For our approach, we consider data rate and latency data because these values majorly influence migration time, offloading capability and capacity. If an application is running inside the global cloud, the request and the calculated response have to be sent via the wireless network channel. The round-trip time (RTT or ping) is crucial for time sensitive applications. In the scope of this work, we use predictive Quality-of-Service (QoS) information for our *Connectivity Monitor*. Predictive QoS is a mechanism that enables the mobile network to provide notifications about future QoS changes to consumers [10]. An exemplary throughput and ping prediction is shown in Figure 3. We receive this data from a mobile network operator. The diagram shows the predicted data rate over a travelled distance. There can be high dynamic changes in short distances, but also sequences with stable values. Concerning the ping, there is smaller change in the absolute value, but the changes are more frequent.

The prediction in low speed use cases (stationary or slow driving) is more accurate than the prediction in high mobility scenarios [21]. We expect that near future prediction models will have $> 95\%$ accuracy [13].

The prediction data is used as one of our parameters for our optimization algorithm to relocate applications before the network data rate or latency parameter values do not allow a seamless shifting anymore. Especially the on-loading process of an application from the global cloud to the local cloud must be completed before a data rate decrease.

B. Framework and Strategies

Our distribution algorithm is executed with a constant period Θ. Depending on the current and predicted network values and

the active set of applications, a new optimum distribution is calculated. The algorithm considers the future time span τ, in which it optimizes the best possible distribution for each point in time. Thus, after the period Θ a new τ is calculated. For the future time interval τ the migration times for all applications are calculated with the predicted network profile.

We reserve a data rate buffer that is not used by application traffic, to guarantee migration times. This reserve is designed to ensure that enough time is available to offload the application with the longest migration time within τ.

To reduce the number of shifting attempts, we filter positive peak data-rates within the time interval τ. Before negative peaks occur, the data rate usage has to be reduced by on-loading applications or by accepting the outage of functions. Outage acceptance for every application is stored in the database. We defined a strategy for special cases such as a sudden connection loss where no time is available to on-load applications. In this case, we skip optimization and run applications with the highest priority, which are determined by the applications parameters.

The system is able to save more energy compared to a non-predictive approach by shifting applications sooner using the predicted data rates. The sooner an application can be offloaded, the less energy it consumes on local hardware. On the other hand, if a future decrease of the data rate is known, then the on-loading of an application can be timed exactly. Thereby, application downtimes can be planned or avoided.

C. Optimization Algorithm

For every application $a \in A$ we define application profile parameters, such as total data size, average RAM consumption, average CPU usage, start-up time, resume time and stop time, which are stored in the database. Correspondingly, we define profile parameters for available resources on compute nodes (CN) $c \in C$ of the hybrid cloud. These parameters are updated continuously by the algorithm and saved in the database.

Using all the parameters from C and A, we define a resource allocation problem. The goal of our optimization algorithm is to calculate the best distribution for all applications on the compute nodes, in order to minimize the energy consumption of all local CNs. Therefore, the total energy consumption on the local CN must be minimized, while meeting multiple constraints. We formulate our minimization objective Ω_{lc} for all applications on local compute nodes as follows:

$$OP : \min_{\forall (c \in C, a \in A} \Omega_{lc} =$$

$$\sum_{c \in C_i} \sum_{a \in A_j} E_{\mathrm{dyn}(c,a)} + \sum_{c \in A_k} E_0(c), \tag{1}$$

$$\text{where } a, c \in \mathbb{Z}.$$

The sum of all energy consumptions $\mathrm{E}_{\mathrm{dyn}}(c, a)$ is built by the iteration over all applications a running on all local compute nodes c. A compute node can be shut down completely, if there is no application running on it. The sum over $E_0(a)$ expresses the basic operating energy consumption of a compute node.

The constraints of our optimization algorithm describe the resource limits of our local hardware and the predicted data rate for each point in time. The total CPU capacity of all local resources must not be exceeded by allocating too many applications on the local compute nodes, which is expressed by the following:

$$\forall\, c \in C, a \in A :$$

$$\sum_{c \in C} \sum_{a \in A} r_{\mathrm{cons}}(a) < r_p(c) \tag{2}$$

where a consumed resource $r_{\mathrm{cons}}(a)$ over all applications a must be smaller than the provided resources $r_p(c)$ on the corresponding compute c, that they are running on. Another constraint is that the number of applications running in the global cloud is limited by the predicted data rate. We sum up the occupied data rate of every application not running inside the local cloud and compare it with the predicted data rate:

$$\forall\, c \in C, a \in A :$$

$$\sum_{c \in C} \sum_{a \in A} \mathrm{B}_{\mathrm{cons}}(a, c) \leq B_p(t) \tag{3}$$

where the sum of the consumed data rate $B_{\mathrm{cons}}(a, c)$ over a applications A running globally, must be smaller than the predicted data rate $B_p(l, t)$.

V. Case Study: Pre-emptive Application Offloading

We use a case study with six applications distributed over three compute nodes on the local and the global cloud as shown in Figure 4. Through the different steps A to C, network conditions are changing, which simulates a vehicle that succumbs changing network conditions. We show how the optimization algorithm is changing the optimum solution (distribution of applications) pre-emptively and shifting one application in advance, before the data rate reserve is not sufficient anymore.

The current data rate is presented on the left side of Figure 4. The on-board compute nodes (CN1, CN2) as local cloud are marked yellow and the green sections show compute node (CN3) as global cloud. Before step A, the optimization algorithm is fed with current and future connectivity data. The different algorithms that we explained in previous sections calculate in Step C that the data-rate is not sufficient anymore for APP4, APP5 and APP6 to run simultaneously in the global cloud. Therefore, it determines when to start the on-loading procedure to shift the best fitting application to on-board hardware under the current conditions. With the predicted data rate and the application size from the database of 12 MB for APP6, 18 MB for APP4 and 9 MB for APP5, the resulting on-loading time is calculated. In this case, the best fitting application is APP5 because it has the lowest energy

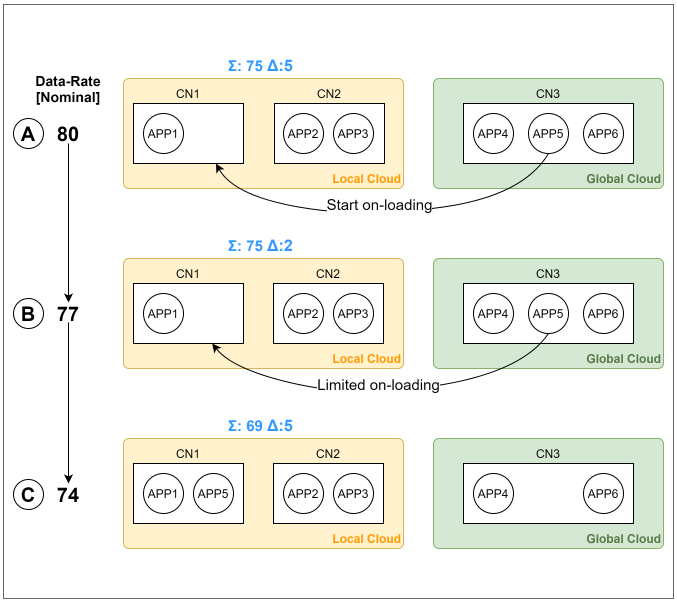

Figure 4: Predictive distribution algorithm scheme: The green part shows all applications running in the global cloud (data centres) and the yellow section shows its counterpart inside a vehicle. Through different data-rates the shifting algorithm starts soon enough to avoid failure through data rate limitations.

consumption (retrieved from the database) and the smallest size.

In step A, the application shift of APP5 starts with a data rate buffer of 5 Mbit/s. In the next iteration B, the data-rate sinks and the remaining shifting data rate is 2 Mbit/s. Until step C, APP5 is completely migrated to local resources. If APP5 would not be on-loaded completely until C, then the service would be unreachable for a moment for local resources. A security data rate buffer is added, to avoid these downtimes.

VI. Performance Evaluation

We have implemented the system architecture that we introduced in Section III in our development and simulation cluster. For the local cloud we used Single Board Computers (SBC) as embedded hardware and an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) from the BMW research and development data centre as global cloud. Furthermore, we built a test framework to run the hardware in a driving simulation and provide input stimuli as predictive connectivity data.

A. Experiments Setup

Our on-board hardware resources in Figure 5 are simulating embedded hardware clusters in future vehicles. They are built up from Raspberry Pi 3B, Raspberry Pi 4, Rock Pi and Intel Minnow Board together as one local cluster. As connection between the nodes we use a standard network switch from Netgear with ethernet CAT 6 cables. Connected to the network

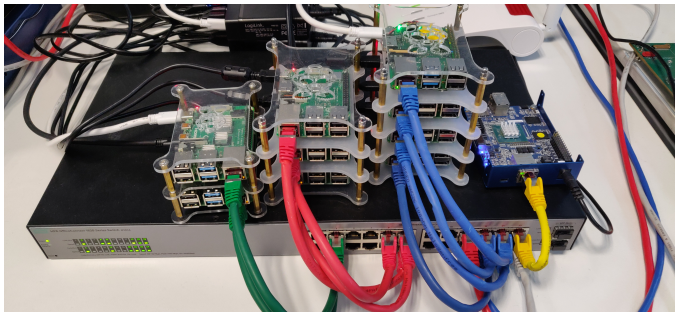

Figure 5: Our test bench with heterogeneous embedded hardware is connected via a switch over a wireless broadband connection to the global cloud. Our set-up simulates a modern vehicle architecture.

switch is a Fritzbox 7590 with an LTE SIM card to provide connection to the Internet.

The global cloud is set up of three AWS EC2 instances with each 16 GB RAM, six virtual CPUs (VCPUs) and 100 GB of storage. The instances are located in a data centre in Frankfurt with 10 Gbit/s Internet connection data rate and an average ping of 20 ms. We use standard Ubuntu 18.10 as the OS on top of the data centre virtualization and configured each instance to run as a computing node. Over an extra instance, the local cloud is connected via a VPN to the global cloud, so that we can use small LAN Classless Inter-Domain Routing (CIDR) segments to address the nodes.

As a cloud framework we use Openstack [14]. Openstack is a decentralized cloud computing framework which provides different software packages for different hardware resources to run as one cluster. The controller node is the main node in the Openstack framework, where the main functions like identity service, message queue or computing manager are located. It is running in the local cloud. The remaining hardware nodes are compute and storage nodes which provide their hardware resources to the cloud. Openstack uses the IP protocol to communicate to the different nodes in local and global cloud.

To solve our optimization problem of minimizing the energy consumption on the local cloud, we use the Gurobi solver in linear programming mode. After we defined the objective and all constraints which we presented in Section III, we set up the solution space as binary. The *User Input Testframe* module feeds the optimization algorithm with predictive network channel parameters. The Gurobi solver calculates a matrix which provides the optimum placement of the applications on the compute nodes. The solution matrix is passed to the controller node, which is shifting the applications on the compute nodes accordingly.

For our test bench we define a set of test applications. The constant period Θ is set to 1 s and τ is calculated dynamically. We measured the energy consumption for all applications running on the different compute nodes.

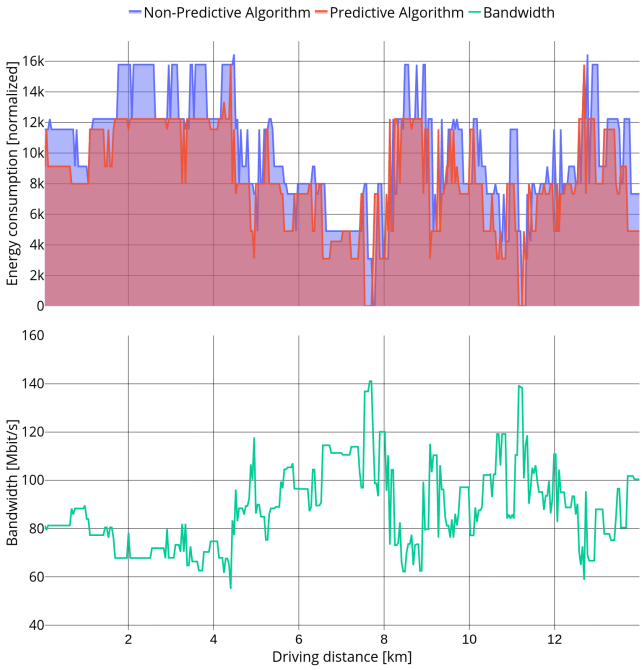

Figure 6: The green line shows the data rate over a driving distance of 14 kilometres. If the data rate is high, then many applications can be offloaded. The red area shows the energy consumption (EC) of the local cloud, when using our predictive offloading algorithm. In comparison, the non-predictive Algorithm is represented by the blue area.

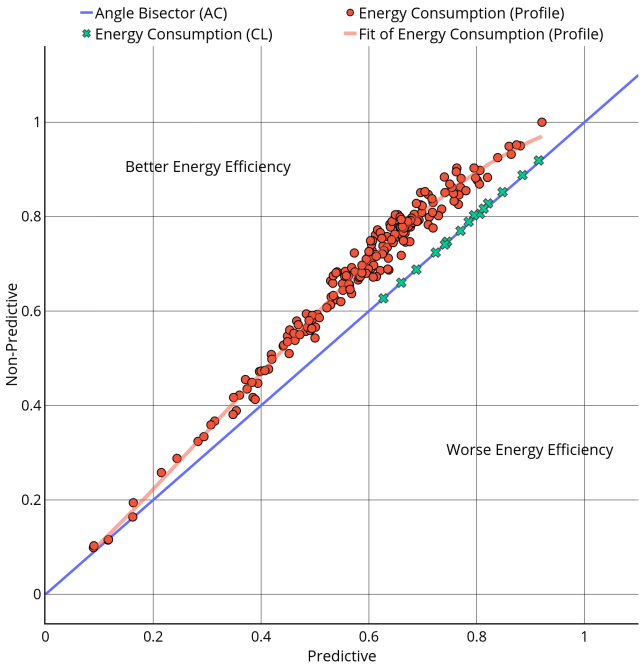

Figure 7: Our predictive approach is plotted over the x-axis, the non-predictive algorithm over the y-axis. The values on both axes is the normalized energy consumption. Every orange dot represents the relation of the energy consumption for both algorithms over a predefined distance. The blue line shows the first angle Bisector.

B. System Evaluation Analysis and Results

In order to efficiently test as many scenarios as possible, we obtain the energy data measurements in our simulation framework. The blueprint for all application profiles are based on real vehicle body and entertainment functions. Thus, we provide a mix of smaller and heavier (resource-intensive) application profiles. To obtain the same starting condition for our tests, the initial application distribution is fixed.

We compared our predictive offloading algorithm in the simulation on the test bench with a non-predictive shifting algorithm that has no information about future channel parameters, as presented in Figure 6. In contrast to our algorithm, the non-predictive algorithm shifts applications as a reaction on the changing network parameters.

Figure 6 presents the changing local energy consumption and predicted data rate over a driven distance. We can see in the energy consumption change in comparison with the data rate that the predictive algorithm is reacting faster to the changes in the data rate, while being able to offload more applications. Between the distance of 6 and 8 kilometres, the algorithm can offload all applications to the global cloud and reduce the local energy consumption to a minimum.

In comparison to the non-predictive algorithm, our predictive approach shows a more constant energy consumption. This indicates that the applications are shifted less frequently. Due to the data rate peak filtering, the location change of applications is less dynamic. Before the data rate is rising, the predictive algorithm starts offloading applications. This can be observed in a small displacement between the green and red area. When forming an integral over both areas, our algorithm outperforms the non-predictive approach by 19% less energy consumption over a distance of 14 kilometres.

C. Scalability Analysis

As extension to the analysis based on Figure 6, we present a scalability analysis, in which we sweep application profile parameters. The result of the simulation is shown in Figure 7. Every orange dot represents one simulation result. It compresses all information from the integral over the red and blue area graphs from Figure 6. This aggregated information of the relation between both algorithm's energy efficiency is plotted over the standard driving distance profile. The area above the blue bisector shows an improved energy efficiency for our predictive distribution algorithm.

Our algorithms performs especially better with medium-sized applications, leading to medium-sized relative energy consumption. This can be seen by the accumulation of simulation results at location (0,55-0,75/0,6-0,8). The closer an orange dot is to the spot (0,1), the more energy efficient it has performed in the simulation compared to the non-predictive approach. For a lower energy consumption, the application sizes are too small, which leads to fast migration times. If the energy consumption is higher, then the application sizes are

too large, which leads to long migration times. In both cases, our algorithm is still better than the non-predictive approach.

On average over more than 200 simulations, our algorithm is 16% more energy efficient than the non-predictive approach. In the best case it is 21% better and in the worst case both algorithms show equal results.

An additional parameter sweep of the compute node profiles is not necessary, because the compute node parameter design depends on application parameters. For the case that there is a sudden connection loss (CL), our algorithm cannot perform well as shown in green crosses in Figure 7. In this case the advantage of predictive data rate information is less effective.

VII. Conclusion

In this paper, we have presented a local energy efficient hybrid cloud architecture for distributed embedded systems. By connecting embedded hardware with data centre high performance servers on an IaaS layer, we made shifting applications between the two domains possible. Our system uses predictive network channel information in combination with a live optimization algorithm to find the best distribution for applications on compute nodes. The best solution is a distribution with the least energy consumption on the local embedded hardware.

In an experimental setup and evaluation, we have shown that the algorithm leads to an average of 16% lower energy consumption with prior knowledge of predictive network channel information than without. In the best case our approach is up to 21% more efficient and in the worst case the results are equal. Our presented architecture and algorithm are a first approach to highly connected future automotive E/E architectures that make high-performance computing with minimum local energy consumption possible.

References

[1] Arun Adiththan, S. Ramesh, and Soheil Samii. "Cloud-assisted control of ground vehicles using adaptive computation offloading techniques". In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 589–592.

[2] Ashwin Ashok, Peter Steenkiste, and Fan Bai. "Adaptive cloud offloading for vehicular applications". In: *2016 IEEE Vehicular Networking Conference (VNC)*. IEEE, 8.12.2016 - 10.12.2016, pp. 1–8.

[3] BMW AG, ed. *Environment Model for Autonomous Driving – The upcoming Challenges*. 2017.

[4] Manfred Broy. "Challenges in automotive software engineering". In:

[5] Mario Kovač et al. "European Processor Initiative (EPI)—An Approach for a Future Automotive eHPC Semiconductor Platform". In: Cham: Springer International Publishing, 2019.

[6] Dejan Kovachev, Yiwei Cao, and Ralf Klamma. *Mobile Cloud Computing: A Comparison of Application Models*. URL: http://arxiv.org/pdf/1107.4940v1.

[7] Dejan Kovachev, Tian Yu, and Ralf Klamma. "Adaptive Computation Offloading from Mobile Devices into the Cloud". In: IEEE.

[8] Ying-Dar Lin et al. "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds". In: *IEEE Systems Journal* (2015).

[9] Andreas Lock. *Trends of Future E/E-Architectures: How new Architectures change the Automotive Industry*. 2019.

[10] *Making 5G Proactive and Predictive for the Automotive Industry*. 2019. URL: https://5gaa.org/wp-content/uploads/2020/01/5GAA_White-Paper_Proactive-and-Predictive_v04_8-Jan.-2020-003.pdf.

[11] Farzaneh Milani and Christian Beidl. "Cloud-based Vehicle Functions: Motivation, Use-cases and Classification". In: IEEE, pp. 1–4.

[12] Farzaneh Milani et al. "Energy Consumption by Cloud-based Vehicle Functions". In:

[13] Ihab Ahmed Najm et al. "Machine Learning Prediction Approach to Enhance Congestion Control in 5G IoT Environment". In: (2019).

[14] Openstack Foundation, ed. *OpenStack Foundation*. 2020. URL: https://www.openstack.org/.

[15] Johannes Pillmann et al. "Empirical evaluation of predictive channel-aware transmission for resource efficient car-to-cloud communication". In: IEEE.

[16] Dominik Reinhardt and Markus Kucera. "Domain Controlled Architecture - A New Approach for Large Scale Software Integrated Automotive Systems". In: 2013, pp. 221–226.

[17] Dominik Reinhardt et al. "High Performance Processor Architecture for Automotive Large Scaled Integrated Systems within the European Processor Initiative Research Project". In: 2019.

[18] Matthias Traub, Alexander Maier, and Kai L. Barbehon. "Future Automotive Architecture and the Impact of IT Trends". In: *IEEE Software* (2017).

[19] Wantanee Viriyasitavat et al. "Vehicular Communications: Survey and Challenges of Channel and Propagation Models". In: *IEEE Vehicular Technology Magazine* (2015).

[20] Huaming Wu, Qiushi Wang, and Katinka Wolter. "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems". In: *2013 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 9.06.2013 - 13.06.2013, pp. 728–732.

[21] Chaoqun Yue et al. "LinkForecast: Cellular Link Bandwidth Prediction in LTE Networks". In: (2018).

[22] Bowen Zhou et al. "A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service". In: