Big Data Architectures for Vehicle Data Analysis

Christian Prehofer
DENSO Automotive Germany
Eching, Germany
C.Prehofer@eu.denso.com

Shafqat Mehmood
DENSO Automotive Germany and TU Chemnitz
Eching, Germany
S.Mehmood@eu.denso.com

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378397

***Abstract*—In this paper, we consider Big Data applications for connected vehicles, where vehicle data is sent several times per second, permitting fine-grained and near real-time analysis of vehicle status and operating behavior. We focus on typical streaming applications and present implementations in Apache Spark and Apache Flink in different architectures and analyze several aspects of Big Data architectures for connected vehicles. First, we aim to show the potential of fine-grained analysis in terms of time resolution and level of detail for vehicular services. For instance, we show how to compute energy efficiency, comparing used energy vs needed energy, on the level of seconds. Secondly, we compare the architecture challenges in vehicular systems, including in-vehicle processing as well as cloud and edge processing. For these, we compare different approaches and architecture solutions. Third, we show that the performance and scalability of our different Big Data processing options differ significantly different for our use case.**

***Index Terms*—Big Data, System Architecture, Vehicle Data, Connected Vehicles, Data Analysis, Edge Computing**

I. Introduction

The number of connected vehicles is increasing strongly, connecting a large set of in-vehicle sensors and actuators. Big Data analysis for vehicle internal CAN bus data and images, which are recorded by vehicle sensors, plays an important role in enabling new services. Today, a modern and connected car already generates up to 25 gigabytes of data every hour. This number goes up to 3600 gigabytes per hour for autonomous vehicles [1].

Many useful evaluations can be done from vehicle data, e.g. to evaluate and improve

- energy consumption,
- safety and comfort, and
- reliability and maintenance effort.

Handling and processing this vast amount of generated data is a big challenge. To provide data-driven services, it is vital to have computational systems that can deal with this huge volume of data, analyze and process it to extract meaningful insights. To address this need, we have seen a significant number of emerging processing engines, tools, and frameworks focused on Big Data analytics. Some of the most widely used big data engines include Apache Spark [2] and Apache Flink [3].

In this paper, we consider Big Data applications for connected vehicles, where vehicle data is sent several times per second, permitting fine-grained and real-time analysis of vehicle status and operating behavior. We focus on typical streaming applications and present implementations in Apache Spark and Apache Flink in different architectures.

First, we aim to show the potential of fine-grained analysis in terms of time resolution and level of detail for vehicular services. For instance, we show how to compute energy efficiency, comparing used energy vs needed energy, on the level of seconds. Secondly, we compare architectures for vehicular data analysis, including in-vehicle processing as well as cloud and edge processing. For these, we need to balance several key requirements:

- Flexibility: flexible analysis on both current and on historic data from different vehicles.
- Speed: responsive behavior requires efficient processing of different data from possibly multiple vehicles, including driver and environment data in a highly accurate way.
- Effort and cost: this includes the amount of vehicle data to be processed, but also the transmission and storage effort.

For instance, the most flexible approach is to collect all data in the cloud. Yet, this approach requires massive data transmission effort and may not be fast in response. Our goal is to show the possible benefits in a case study and then explore the architecture design space.

Our case study considers electric vehicles (EV) data for vehicle energy consumption and we build applications to provide fine-grained analysis wrt time. Our focus here is on vehicle status and driving data, not assisted or automated driving functions, which require real-time in-car processing of considerable amounts of data.

II. Vehicle Data for Use Case Evaluation

For our case study, we use one of the first comprehensive data sets for trips with electric vehicles. The Vehicle Energy Dataset (VED) is published in [4]. According to this paper, all data signals are obtained from OBD-II and they are using 3 electric vehicles, each with a weight of 1588 kg. In this dataset vehicle id and trip id, both are available. One vehicle id can have multiple trip ids and one trip id can have multiple vehicle ids but the combination of both vehicle id and trip id will be a unique id. The altitude of the vehicle is not provided, but can be obtained using commercial APIs, such as the Google Maps API.

In summary, the VED dataset has data from 504 trips (which covers 8000 km) with the following columns/values:

- Vehicle Id

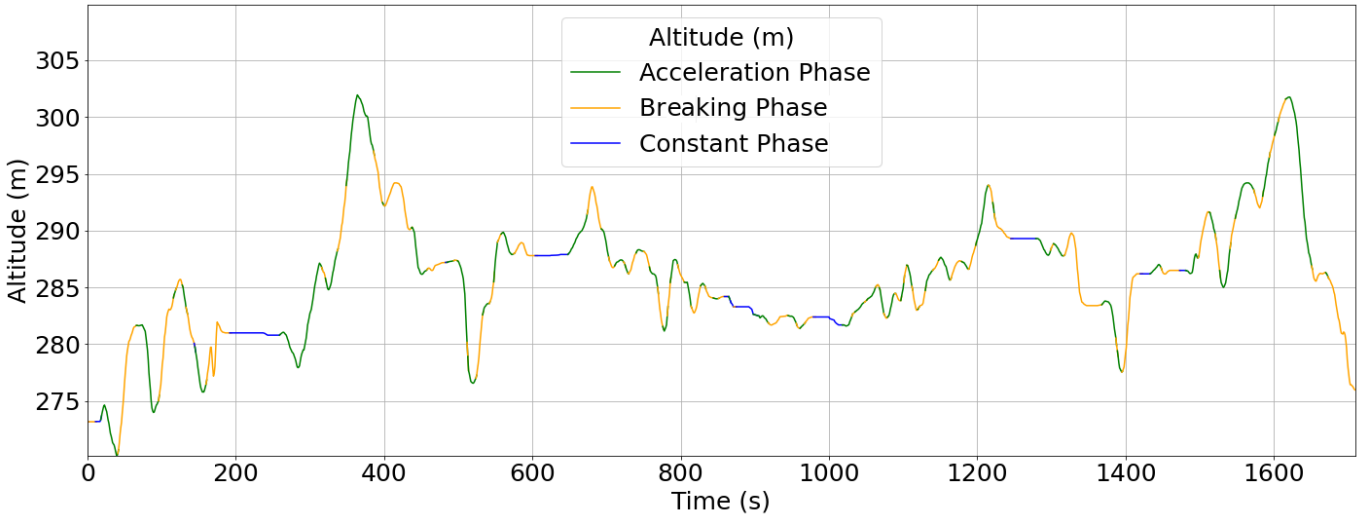


Fig. 1. Altitude with Acceleration and Breaking phases

- Trip Id
- Timestamp (ms)
- Latitude (deg)
- Longitude (deg)
- Altitude (m)
- Vehicle Speed (km/h)
- Outside Air Temperature (° C)
- Air Conditioning Power (watt)
- Heater Power (watt)
- HV Battery Current (A)
- HV Battery Voltage (V)
- HV Battery SOC (%)

For most of the data items, accurate data is available on the level of seconds, which permits highly accurate analysis.

III. Use Case: Fine Grained Vehicle Energy Consumption Analyis

In the following, we describe our use case to evaluate driving behavior and vehicle energy consumption on a fine-grained level. In particular, with this data set, we can evaluate the actually consumed energy versus the needed energy for the movement of the vehicle. This latter is formalized in a concept called VSP, vehicle specific power, and models the physics of moving a vehicle on the road. Then, we show different evaluations based on these results.

A. Data Preparation and Driving Phases

In the following, we describe our data preparation methods and the extraction of driving phases. Timestamps in our dataset are in milliseconds. First of all, we convert timestamps into seconds from milliseconds by interpolation. To reduce noise from raw data there are several filters. The moving average filter used for this, which is a low pass filter mostly used for smoothing time-series data.

Next, we divided the vehicle trips into three kinds of phases:

- Constant speed phase: If the vehicle is moving with constant speed or the vehicle is not moving (idle state).
- Acceleration phase: If the vehicle speed is increasing.
- Breaking phase: If the vehicle speed is decreasing.

Similarly, we use the following phases:

- Uphill phase: If altitude is increasing.
- Downhill phase: If altitude is decreasing

These phases are illustrated in Figure 1. For instance, after 450, 1385, and 1610 seconds, altitude is decreasing sharply but the driver still accelerating the vehicle and similarly braking while altitude is increasing sharply.

B. Actual Power Consumption

Instant current and voltage of the battery of electric vehicles are available in the dataset, so instant power consumption can be calculated by multiplication of instant current and voltage. Power consumed by the heater and air conditioner is also given in the dataset. To calculate actual power used for vehicle movement, heater, and air conditioning consumption should be subtracted from total power consumption.

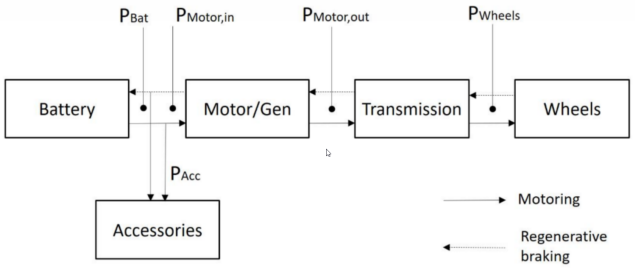
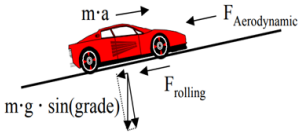


Fig. 2. Block diagram of power flow in electric vehicle [5]

The vehicle model and battery capacity are not disclosed due to some privacy concerns by Oh et al. [4]. After contacting the authors of the Vehicle Data set [4], he explained that the

battery capacity is in the range of 20-25 kWh, the power range of electric motor is 70∼100 kWh, and the EVs are equipped with heat pumps, so they do not consume as much battery power as other EVs for air-conditioning and heating. They do not use a 12V battery for air-conditioning nor heating. Thus, the power flow of electric vehicle is shown in Figure 2.



$$\text{VSP} = \frac{\text{Power}}{\text{Mass}} = \frac{\frac{\text{d}}{\text{dt}}(\text{E}_{\text{Kinetic}} + \text{E}_{\text{Potential}}) + \text{F}_{\text{Rolling}} \cdot \text{v} + \text{F}_{\text{Aerodynamic}} \cdot \text{v} + \text{F}_{\text{int ernal friction}} \cdot \text{v}}{\text{m}} =$$

$$\approx \text{v} \cdot \text{a} \cdot (1 + \varepsilon_{\text{i}}) + \text{g} \cdot \text{grade} \cdot \text{v} + \text{g} \cdot \text{C}_{\text{R}} \cdot \text{v} + \frac{1}{2} \rho_{\text{a}} \text{C}_{\text{D}} \frac{\text{A}}{\text{m}} (\text{v} + \text{v}_{\text{w}})^2 \cdot \text{v} + \text{C}_{\text{if}} \cdot \text{v}$$

Fig. 3. Formula and Illustration of Vehicle Specific Power [6]

TABLE I
VSP PARAMETERS EXPLANATION

Param.	Explanation
VSP	Vehicle Specific Power in kW/Metric Ton
ϵ	equiv. translational mass of rotating components
v	velocity in m/s
a	acceleration in m/s^2
grade	road grade or Vehicle vertical rise/horizontal run
g	acceleration due to gravity $g = 9.81 m/s^2$
C_R	coefficient of rolling resistance (dimensionless)
C_D	drag coefficient (dimensionless)
A	frontal area of vehicle
ρ_a	average density of air
v_w	heatwind into vehicle
m	vehicle mass in Metric Ton

C. Vehicle Specific Power (VSP)

Vehicle Specific Power Model basically estimates the required energy for vehicle movement in a physical model, which depends on velocity, acceleration and road grade. See Figure 3 for illustration and Table I for explanation of parameters.

IV. Architectures for Vehicle Data Analysis

In the following, we give an overview of the data collection, storage and processing options for connected vehicle data and compare these for connected vehicle applications. Edge computing aims to place the processing closer to the location where the data is created [7]. Figure 4 shows a schematic picture of the connected vehicles, edge computing and backbones. Clearly, there are several variations in terms of wireless communications, from 2G to upcoming 5G networks, and also in terms of placement of the edge computing [7]. Here, we show mobile edge computing in the sense of [8]: ”Mobile edge computing (MEC) is an emergent architecture where cloud computing services are extended to the edge of networks into the mobile base stations.” Edge computing could also be placed behind the fronthaul network, which is connecting the base stations. Another option is vehicle edge computing, placing the edge computing inside vehicles [9]. To avoid confusion, we refer to this as in-vehicle processing.

A simplified version of the VSP model [6] is as follows:

$$VSP \approx v \cdot [a \cdot 1.1 + 9.81 \cdot grade + 0.213] + 0.000426 \cdot v^3$$

A. Centralized Big Data and Batch Processing

Centralized Big Data means to store the data, including historic data from vehicles, in a central location and operate in batch processing on top of them. Thus, different kinds of evaluations can be performed using the complete set of historic data. This forms the most flexible option, as all data is available and also scalable, cost-efficient cloud processing resources are available [10].

The main drawback is the data transmission and also the storage. While 5G aims for a 1ms round trip time, wireless data transmission is typically much higher in delay [11]. Also, the wireless link is typically a bottleneck due to the shared nature of resources, and is expensive as 80% of the cost of a wireless network is in terms of the radio access network [12].

B. Edge Computing

The main goal of edge computing is to bring the computing closer to the data, thus reducing network load and offering faster round trip times. Compared to in-vehicle networks, edge computing devices are permanently connected to the Internet. This is important for many use cases. As connected vehicles use cellular networks, this wireless connection to vehicles is currently causing much higher delays than the backbone network. Depending on the network setup, the latency is hence not significantly reduced in most cases compared to cloud services. This problem may be improved significantly with 5G networks as discussed above.

With respect to a cloud-based solution, there are two major drawbacks. First, the processing in the edge only has a local view of the data. This can limit the processing and the analysis due to the lack of global knowledge. Furthermore, global data needs to be distributed and, depending on the use case, also be kept in sync at the edge devices. As a simple example, assume that a software update needs to be applied synchronously to some devices connected to different edge devices. This can create considerable overhead and management effort.

A second drawback is the limited computing resources compared to cloud computing, which can provide computing resources on demand in very high scalability. It is not clear if edge computing can compete here, even though computing may be outsourced to the cloud if needed.

C. In-Vehicle Processing

In-vehicle processing is desirable for many use cases which need highly responsive and reliable results. A typical case are driving assistant systems which require guaranteed response time, which is difficult over a wireless network.

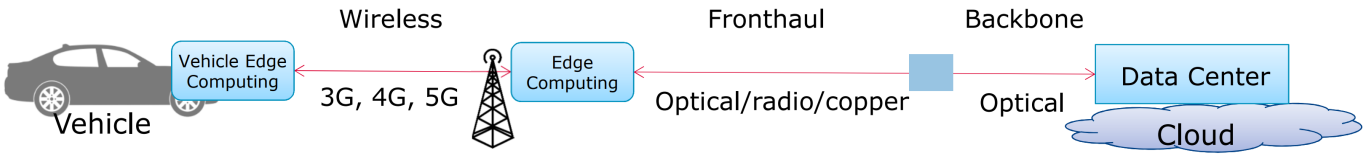


Fig. 4. Vehicle - Edge Cloud Computing Architecture

The challenges for in-vehicle processing are clearly energy consumption, higher cost of computing equipment and also provisioning. Energy consumption may also include cooling effort if needed. The higher cost of computing is due to higher robustness requirements regarding temperature and physical durability. Provisioning is another challenge as vehicles have a long life span of over 10 years. Computing needs to be planned during development as computing hardware is difficult to renew or retrofit.

V. Use Case Evaluations

In the following, we aim to show the potentials of Big Data analysis for our application area, and also compare different streaming and batch processing approaches.

A. Centralized Batch Processing with Apache Spark

Apache Spark 2 is an open-source collaborative data processing system used for a wide range of circumstances and big data analytics. Spark comes with APIs in Python, Java, Scala, and R for development. On top of the Spark computing framework, it has libraries for graph analysis, machine learning, SQL, and stream processing. Spark permits one to write a query with high-level operators abstracting from fault tolerance, distribution, and parallelism.

Spark allows the distribution of data and computations over clusters with multiple nodes. The status of the program finally reflects all data records in the data stream exactly once. For SQL jobs, SparkSQL uses an optimizer that supports rules and cost-based optimization. A computational model is based on micro-batches with low overhead of fault tolerance and with high latency and throughput. Resilient Distributed Dataset (RDD) is a core data structure but difficult to work with RDDs directly. Instead, Spark DataFrames, similar to SQL tables, that are easy to understand and are optimized for complex operations.

As mentioned, connected cars generate a huge amount of data, and as automobiles get increasingly automated and autonomous that will continue to increase. Thus, one of the framework's key requirements is to be scalable enough to execute the same on a vast volume of vehicle data.

The used Electric Vehicle Dataset has more than 500 trip data (which covers 8000 km). We use Apache Spark 2.4.4 for processing big data.

The evaluation scenario is the comparison between actual power consumption and vehicle specific power (VSP):

1) Read data from CSV file.
2) Calculate instantaneous power consumption from voltage and current.
3) Calculate instantaneous power in kilo-watts after subtracting AC and heater power consumption.
4) Apply smoothing on instantaneous speed and altitude.
5) Mark acceleration and braking phases using speed difference.
6) Compute acceleration using speed difference and time difference.
7) Calculate distance from the previous point using latitude and longitude.
8) Computing road grade using distance and altitude difference.
9) Computing instantaneous VSP using speed, acceleration and road grade.
10) Visualizing actual power, VSP and speed with acceleration and braking phases as shown in Figure 6.

TABLE II
Hardware specifications of machines

	Machine A	Machine B
OS	Ubuntu 18.04	Ubuntu 18.04
Processor	Intel Core i3 1.7GHz	Intel XeonW 3.7GHz
Cores	2	8
Memory	4GB DDR3 1600MHz	16GB DDR4 2666MHz
Storage	SATA	SSD

We evaluate this scenario on two different machines with different specifications in the local environment. Table II contains hardware specifications. Machine A is a laptop with 2 CPU cores, whereas Machine B is a recent desktop PC with 8 cores. Spark automatically distributes data and runs in parallel even in local/standalone mode. More cores with Spark means high performance as shown in Figure 5. Machine B is 8.63 times faster than Machine A. We have run multiple implementations and observed the time it took to calculate actual power and VSP for data from over 500 trips with 67 MB. Figure 5 demonstrates the effects of how long each implementation took on both machines to evaluate actual power and VSP. For a cluster of multiple nodes, we assume that Spark can even faster as the processing can be distributed.

B. Actual Power Consumption vs VSP

Comparatively, actual power consumption and estimated VSP are so close to each other but on some high/low points these are not the same. If power consumption is greater than VSP while acceleration, it is understandable as VSP is only the minimal, physically needed energy. Surprisingly, it is less than VSP during energy recuperation in some cases on several trips. As an example, see Figure 6. The answer to this question

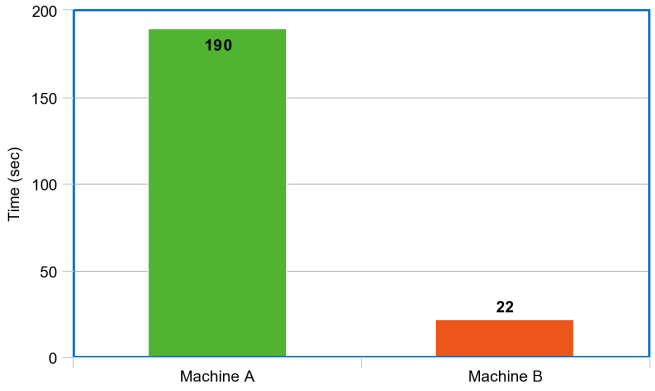


Fig. 5. Comparison of Spark batch execution time on two different machines

is requires a detailed inspection of the dataset: the GPS values have delays and at some points, up to 5 - 7 second delay for updating latitude and longitude. If the distance is zero, then the road grade will also be zero, and if the road grade is zero then the potential force value cannot be considered correctly for VSP. This is however a specific artifact of our data set and could be improved with higher accuracy positioning or more sophisticated data preparation, considering speed and positioning.

C. Energy Consumption w.r.t Temperature

Outside air temperature also has an effect on vehicle performance. To analyze energy consumption w.r.t temperature, we use temperature bins from -15 to 35 ° C with the window range of 5 of each bin, and calculate the ratio between actual power consumption and VSP. The results are shown in Figure 7. Ideally, this ratio should be close to 1. Our evaluation shows that for negative temperatures, vehicle performance is getting worse. About 90% trips have this ratio less than 1.8, while 70% have it less than 1.5. The explanation for this higher energy consumption could for instance be heating as well as less efficient driving.

VI. Stream Processing and Distributed Data with Apache Flink

Apache Flink [13] is an open-source framework proposed as the true streaming processing engine and designed to process distributed batch and stream data. As the core of Flink, streams and transformations are performed on dataflows. Batch processing is also developed on top of the streaming engine. True streaming, memory management and high processing speed are the main advantages of Flink.

Flink processes data stream in real-time while Spark process data in chunks, or micro-batches. In Flink, all incoming rows are processed as soon as they arrive without waiting for other rows. The status of the program finally reflects all data records in the data stream exactly once. Such optimizations are built in for Flink. Spark batch jobs must be manually optimized through detailed control of partitioning and caching and adjusted to specific data sets. Whenever possible, Flink is capable to have intermediate results on its data processing while Spark is not. An overview comparison of Spark and Flink is shown in Table III.

TABLE III
Hardware specifications of machines

	Spark	**Flink**
Guarantee	Exactly once	Exactly once
Latency	High	Very Low
Throughput	High	High
Computation model	Micro-batch	Streaming
Overhead of fault tolerance	Low	Low
Flow control	Problematic	Natural

A. Implementation

DataStream and DataSet are the core APIs of Flink. The DataSet API is for batch processing while the DataStream API is for stream processing. In our work, we are using the DataStream API.

Basic steps of DataStream API implementation and data flow as follows, and illustrated in Figure 8:

- Set up the streaming execution environment.
- Data from source (File, Kafka, Socket etc).
- Apply keyBy Window Aggregation using Session Window based on trip id and timestamp with the gap of 1 second.
- Apply other transformation operators and calculation.
- Add data sink (Kafka, Databases, File etc).

For performance evaluation, we have measured how many vehicle data streams can be handled as a maximum for the above machines. The result is illustrated in Figure 9 and shows that for Machine A, up to 15000 data sets from vehicles can be handled at a time. As each data set is about 100 Byte, and data is sent $2-3$ times per second, we get a total throughput of about $3{-}4MB/s$. The more recent and more powerful machine B yields an improvement of factor 3.

VII. Discussion

In the following, we discuss several aspects of connected use cases with different tools and architecture.

A. Comparing Streaming vs Batch Processing

In the above, we have evaluated Apache Spark for batch and Apache Flink for stream processing. As we are using batch and stream modes, as well as different APIs, we cannot directly compare both.

There are several papers comparing Apache Spark and Apache Flink. A detailed comparison of Spark vs Flink performance is presented in [14] for batch operations, but not using SQL APIs with window operations as needed in Apache Spark for our use case. The work in [15] compares Apache Flink, Spark and Storm for two applications using Linear Regression and Bayesian and Flink and Spark show very comparable performance results.

Here, we have evaluated only one use case, using different kinds of APIs in Apache Flink and Apache Spark. Yet, we can

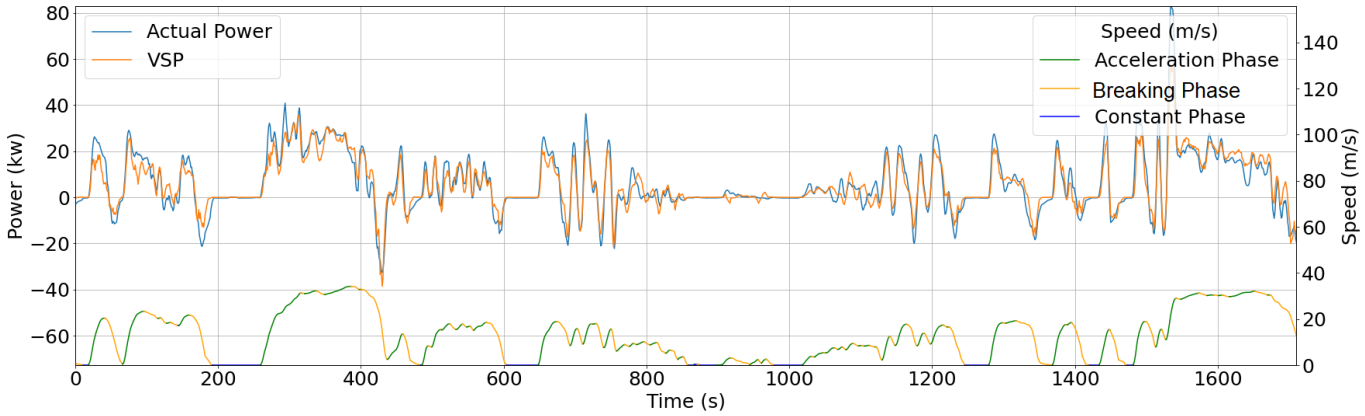


Fig. 6. Actual Power Consumption vs VSP

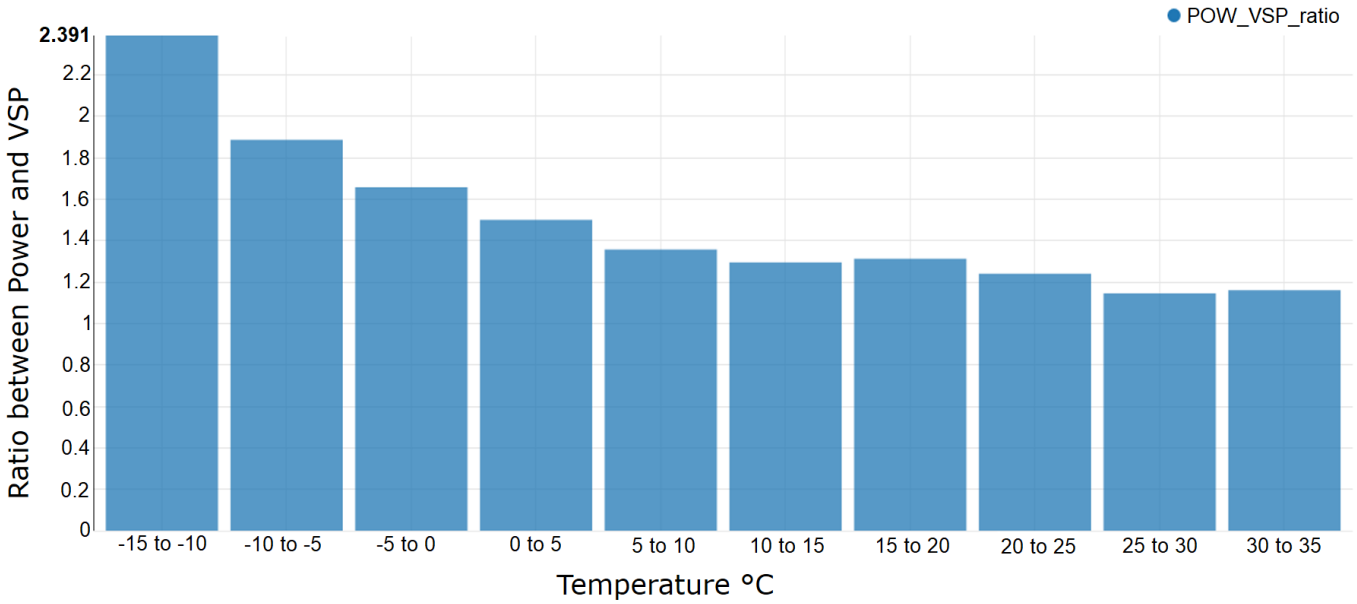


Fig. 7. Energy consumption w.r.t temperature

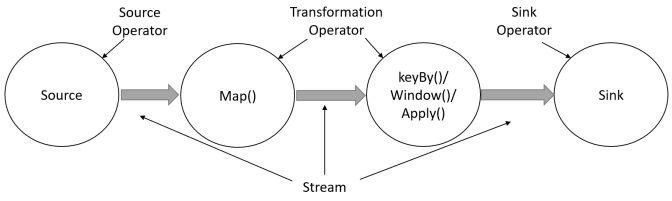


Fig. 8. Streaming Dataflow Diagram [13]

compare the overall processing performance in terms of data handled per second. We can however compare the throughput of both in terms GB/s processing performance between batch and stream. While this is does not consider system startup costs, it is a good option point to compare the efficiency and suitability of different approaches for larger datasets.

As discussed above, Machine A would yield $3-4MB/s$ and Machine B $9-12MB/s$. On the other hand, the total data size processed in the Apache Spark batch case for 500 trips data size is 67 MB. Machine A with Apache Spark computes 67MB in 190s, which is about $0,35MB/s$, while Machine B is about $3MB/s$ Thus, we can see that Apache Flink in streaming mode has significantly higher throughput, especially for the lower end machine. This result is different from other evaluations, and we assume that our use case is more suitable for streaming, and also Apache Spark requires more sophisticated SQL APIs for our usage of window operations. Flink did offer very expressive windowing operations outside the SQL APIs.

Secondly, we can compare scalability across different machines. For Spark batch mode, we got a difference of factor 8.6 between the machines, while with Flink, the factor was

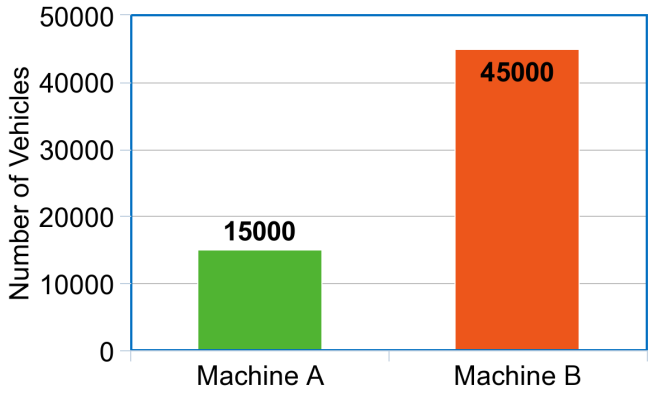


Fig. 9. Streaming Performance Results

only 3.

B. Discussion of potential Usage and Extensions

Considerable research focuses on energy and pollution using data analysis. Eco-driving suggests maintaining a steady speed, and moderate acceleration that will change the driving behavior but also help in maintaining your vehicle [16]. Ericsson [17] researched on fuel consumption and its effects on pollution. He decided to focus on the relation that how different driving behaviors affect fuel consumption. Through his experiments, he found out that vehicles between 50-70 km/h would decrease the fuel consumption significantly if immediate acceleration or braking is avoided. De Cauwe et al. [18] considers vehicle range issue by providing an energy-consumption predictive approach. It is intended for routing in an energy efficient way. The approach takes real-world data from regional and environmental data and by integrating them, forecasts consumption on every path or path network. Frey et al. [19] tried to cover the gap between the transport and emission model. Vehicles Specific Power (VSP) bins are used for real-world distributions and they are linked with an average speed of different road classes.

Many useful evaluations can be done from vehicle data, e.g. to evaluate and improve

- Energy consumption [18], [20], [21]
- Safety [22]
- Reliability and maintenance effort [23]

C. Discussion of Architecture Options

In the following, we summarize our findings regarding the different Big Data Processing options. This considers the challenges of the vehicle environment, such as mobility, intermittent and varying data connections as well as the huge amount of data, and then compares both approaches in a systematic way.

First, there are different goals and requirements of different use cases: Online and near real-time results in cases where data has to be evaluated within a few seconds. For these, stream processing appears the more natural approach. In other applications, we aim for hypothesis testing including all prior, historic data to maximize the validity of the claim. In this case, batch processing and central collection of all data are suitable, as current Big Data solutions require a local database and a local processing cluster for performance.

However, data collection over wireless networks is challenging due to varying quality of networks, possible overload and also the cost of the mobile network which is significantly higher than wired backbone networks.

Regarding local data with stream processing in edge computing, we aim for better scalability in terms of data processing effort as data is processed for many cars in one unit. Also, round trip time can be improved compared to cloud networks. However, with current technology before 5G, the highest delay is in the wireless link.

We summarize the different options for Big Data processing in Table IV.

D. Big Data Use Cases and Suitable architectures

Regarding connected vehicles, we can now discuss which use case would be handled best in what architecture.

First, calculating the VSP ratio for different trips considering the average trip temperature is an example which requires a considerable amount of historic data. This must be distributed over a long period of time (summer and winter). It must also be distributed over geographic regions as some may not have very cold or very hot temperatures.

This means that edge computing is not directly applicable as one edge device covers only one geographic region. Of course, preprocessing would be possible in local regions, but the full evaluation requires some exchange between edge devices or a central solution.

A second example is the energy consumption of different vehicles in the same locations. This would be an ideal application for edge computing, as information from vehicles in one geographic area is sufficient.

A third example is to calculate the breaking and acceleration phases. This requires, as discussed above, to calculate the road inclination from GPS or high-accuracy maps. If this is known locally in a vehicle, it can be calculated inside a vehicle or also in edge computing. Edge computing would however assume that the full data is transmitted, as in applications with edge and cloud processing. In addition to the effort and cost, this is also more error prone in case of intermittent wireless connectivity. Thus, it may also be challenging in case of real-time needs.

VIII. Related Work

While there is considerable literature on the vehicle, edge and cloud software, it is still difficult to compare these, especially considering Bid Data applications. A method to compare in vehicle and cloud processing was presented in [24], yet not considering Bid Data nor detailed cost or resource model. Similarly, other papers aim to distribute specific applications across the devices, edge and cloud, but without a clear, more general methodology, e.g. [25].

TABLE IV
COMPARING DATA ANALYSIS OPTIONS FOR CONNECTED VEHICLES

	Centralized batch processing	**In-vehicle stream processing**	**In-edge stream processing**
Available Data	Collected vehicle data, external context	In-vehicle data	Local vehicle data streams, external context possible
Processing Power	High, flexible allocation	Low power, limited	Medium to high
Storage Capability	Very high	Low	Medium
Networking Demand	Very high	Very low or none	High on wireless network, low on backbone

A recent survey [10] considers elasticity in data stream processing, but not considering performance measurements. In terms of scalability, the work in [26] shows that Flink can be used in a lower-end environment with Raspberry Pi 3b+ (Quad-core 64-bit ARM processor, 1GB of RAM, 32GB of storage), but does not compare to other solutions.

Regarding vehicles, other research [21] analyzes the relation between electric vehicle power, velocity, acceleration, and road condition. Base on the analysis output Electric Vehicle power estimation model can be developed. The evaluation results from test vehicles suggest that the proposed model can estimate the electric vehicle instantaneous power and trip energy consumption successfully. Similarly, [18], [20] employ Big Data for potential evaluation of vehicle data, but do not consider such fine grained analysis, nor discuss general architecture options.

IX. SUMMARY

In this paper, we did consider Big Data applications for connected vehicles. First, have discussed the potential of fine-grained analysis in terms of time resolution and level of detail for vehicular services. For instance, we show how to compute energy efficiency, comparing the use of energy vs needed energy, on the level of seconds. Secondly, we have compared the architecture challenges in vehicular systems, including in-vehicle processing as well as cloud and edge processing. We have considered the challenges of the vehicle environment, such as mobility, intermittent and varying data connections, as well as the huge amount of data for wireless transmission, and have compared the approaches in a systematic way.

Third, we have compared the performance and scalability of our different Big Data processing options that differ significantly different for our use case. The main findings of our use case are as follows:

- The overall processing throughput is much higher for Flink Streaming compared to Apache Spark batch processing, even for machines with 16 virtual cores.
- Apache Spark batch processing scales better for higher-performance machines, compared to the scalability of Flink streaming for the same machines.

Clearly, our performance evaluation considers a specific use case, implemented also in different APIs. In Apache Spark, we had to resort to the SQL API, whereas Flink did offer very expressive windowing operations outside the SQL APIs. Thus, we assume that our use case is more suitable for streaming, as the data is essentially processed locally in the data stream.

REFERENCES

[1] F. Fleutiaux, "Vehicle data is more profitable than the car itself," 2018. [Online]. Available: https://www.telekom.com/en/company/management-unplugged/francois-fleutiaux/details/vehicle-data-is-more-profitable-than-the-car-itself-516208

[2] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin *et al.*, "Apache Spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[3] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[4] G. Oh, D. J. Leblanc, and H. Peng, "Vehicle energy dataset (ved), a large-scale dataset for vehicle energy consumption research," *arXiv preprint arXiv:1905.02081*, 2019.

[5] K. N. Genikomsakis and G. Mitrentsis, "A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications," *Transportation Research Part D: Transport and Environment*, vol. 50, pp. 98–118, 2017.

[6] J. L. Jimenez, P. McClintock, G. McRae, D. D. Nelson, and M. S. Zahniser, "Vehicle specific power: A useful parameter for remote sensing and emission studies," in *Ninth CRC On-Road Vehicle Emissions Workshop, San Diego, CA*, 1999.

[7] 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, 2016.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[9] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked vehicle edge computing: Exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, pp. 66 649–66 663, 2018.

[10] M. D. de Assuncao, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications*, vol. 103, pp. 1–17, 2018.

[11] M. Lauridsen, L. C. Gimenez, I. Rodriguez, T. B. Sorensen, and P. Mogensen, "From lte to 5g for connected mobility," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 156–162, 2017.

[12] "5g mobile operator strategies to cut their huge power cost," 2020. [Online]. Available: https://www.telecomlead.com/5g/5g-mobile-operator-strategies-to-cut-their-huge-power-cost-94645

[13] "Apache Flink 1.10 documentation: Dataflow programming model," https://ci.apache.org/projects/flink/flink-docs-release-1.10/concepts/programming-model.html#programs-and-dataflows, 2020, (Accessed on 09/11/2020).

[14] O.-C. Marcu, A. Costan, G. Antoniu, and M. S. Pérez-Hernández, "Spark versus flink: Understanding performance in big data analytics frameworks," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2016, pp. 433–442.

[15] H. Lee, J. Oh, K. Kim, and H. Yeon, "A data streaming performance evaluation using resource constrained edge device," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2017, pp. 628–633.

[16] C.-F. Yeh, L.-T. Lin, P.-J. Wu, and C.-C. Huang, "Using on-board diagnostics data to analyze driving behavior and fuel consumption," in *International Conference on Smart Vehicular Technology, Transportation, Communication and Applications*. Springer, 2018, pp. 343–351.

[17] E. Ericsson, "Independent driving pattern factors and their influence on fuel-use and exhaust emission factors," *Transportation Research Part D: Transport and Environment*, vol. 6, no. 5, pp. 325–345, 2001.
[18] C. De Cauwer, W. Verbeke, T. Coosemans, S. Faid, and J. Van Mierlo, "A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions," *Energies*, vol. 10, no. 5, p. 608, 2017.
[19] H. C. Frey, N. M. Rouphail, and H. Zhai, "Speed-and facility-specific emission estimates for on-road light-duty vehicles on the basis of real-world speed profiles," *Transportation Research Record*, vol. 1987, no. 1, pp. 128–137, 2006.
[20] G. M. Fetene, S. Kaplan, S. L. Mabit, A. F. Jensen, and C. G. Prato, "Harnessing big data for estimating the energy consumption and driving range of electric vehicles," *Transportation Research Part D: Transport and Environment*, vol. 54, pp. 1–11, 2017.
[21] X. Wu, D. Freese, A. Cabrera, and W. A. Kitch, "Electric vehicles' energy consumption measurement and estimation," *Transportation Research Part D: Transport and Environment*, vol. 34, pp. 52–67, 2015.
[22] S. Amini, I. Gerostathopoulos, and C. Prehofer, "Big data analytics architecture for real-time traffic control," in *2017 5th IEEE international conference on models and technologies for intelligent transportation systems (MT-ITS)*. IEEE, 2017, pp. 710–715.
[23] E. Husni, G. B. Hertantyo, D. W. Wicaksono, F. C. Hasibuan, A. U. Rahayu, and M. A. Triawan, "Applied internet of things (iot): car monitoring system using ibm bluemix," in *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. IEEE, 2016, pp. 417–422.
[24] T. Furuichi and K. Yamada, "Next generation of embedded system on cloud computing," *Procedia Computer Science*, vol. 35, pp. 1605–1614, 12 2014.
[25] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.
[26] D. Battulga, D. Miorandi, and C. Tedeschi, "Fogguru: a fog computing platform based on apache flink," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2020, pp. 156–158.