Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 9, November 2021: 8222-8230

Edge Computing- The Distributed Computing Paradigm in Self-Propelled Applications

Parul[1]
Research scholar
Baba MastnathUniversity, Rohtak
dparul79@gmail.com
Deepak Kumar[2]
Assistant Professor
Department of Computer Science,
Government college, Panchkula
dks.karnal@gmail.com
Dr .Pooja Vyas[3]
Assistant Professor
Department of management
Indira Gandhi University, Meerpur, Rewari, Haryana
pooja.vyas9@gmail.com

Abstract:
Now a days most of the techniques are working on critical time data processiong, cloud computing is one of the most popular soft computing techniques, is hard to implement because it need higher latency and bandwidth. To rectify this problem edge computing is used. Edge computing isthe prototype or methodology of critical time data processing which processes data at network edge. Edge computing's main work is that it reduces the location gap in between the storage where data is stored and the computation location. As the name suggest its an time critical application so used where time is crucial. In this paper we propose a model to create edge computing device by using linux platform. It is basically application which is used in latest model car which have facility of on board and gps sensor. Multiple data segments has been collected and analysed by Edgex which supports to implement IoT applications. This research shows that edge computing can improve IoT applications performance.
Keywords:- critical time, data processiong, soft computing, IoT.

I. INTRODUCTION

This paper will discuss the edge computing model we developed for an automotive application. The hardware that was used includes Raspberry Pi Model 3b as the processing unit, whereas we have an external GPS unit for getting data about the location. This model also has additional hardware which is used to extract data from the OBD-II port to the processing unit.
The Introduction section is followed by a background about OBD-II data and overviews of edge computing and OBD-II data port.

II. BACKGROUND

Since the early 1980s car manufacturers began using electronic techniques to diagnose engine problems and control engine functions. But the systems developed had separate hardware interfaces, which forced the mechanic to buy a different tool to access each car.

To overcome this issue international Standardization Organization (ISO) and Society of Automotive Engineers (SAE) in the 1990's issued international Standardization Organization (ISO) standards which allowed mechanics to view the digital information using a single connector (SAE J1962). This was the evolution of OBD-II (On-Board Diagnostics II) system ports till now. Getting OBD-II data has become a common thing these days for any person. Even a layman can access his car's information using in market tools. With the introduction of Bluetooth adapters and the immense number of mobile applications available in app stores it has become very simple. For advancement in the use of OBD-II ports we can develop solutions using edge computing and Internet of things.

III. OVERVIEW OF EDGE COMPUTING

Firstly, what is edge-computing? Edge-Computing is a networking philosophy focused on bringing computing as close to the source of data as possible in order to reduce bandwidth use and latency.

Now, why do we need edge-computing? Edge computing is used to process time-sensitive data, i.e., data that has to be processed at a very high speed. Also, edge computing can be accessed in remote locations where there is no connectivity to a centralized server, giving it an advantage over cloud computing.

What can we use edge-computing for? There are absolutely no limits to the number of applications that can be developed using edge-computing. Few of the famous and forthcoming innovations include autonomous or self-driving cars and the latest generation of wireless technology 5G.

As we can see that the data is being processed at the edge of the network and is not being processed in a remote cloud server. Edge-computing also has better security compared to cloud-computing. Another major advantage for using edge-computing is that it decreases cost due to less bandwidth usage and server resources. It helps to save a good amount of money which is being used for these resources. It also helps in giving each device an added functionality so that they can be made smart easily.

Edge-computing is the future of IOT and will play an important role in making devices and systems faster and more efficient. Thus, making it very valuable for real-time projects.

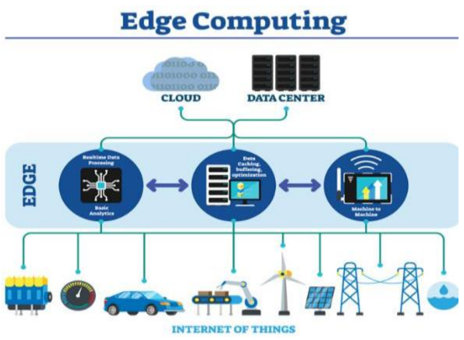


Fig 1: Brief Layout of Edge-computing (https://link.springer.com/chapter/10.1007/978-981-13-1056-0_66)

IV. OVERVIEW OF OBD-II DATA

OBD is an automation of electronic systems that provides a vehicle with self-diagnosis and reporting capabilities which is formally known as On-Board Diagnostics, which is helpful in accessing subsystem information of the vehicle, the purpose is to repair and also check the performance monitoring.

It is a typical one, where light-duty vehicles use it across many countries to get exact whereabouts and data of the vehicle. Engine control module in a vehicle creates data that needs to be sent. We can consider this as the CPU of a vehicle.

When a car is taken for servicing, a mechanic can identify the problem by connecting OBD, with the help of a scanning tool which reads trouble codes. This helps a mechanic for easy identification and to accurately diagnose malfunctions. It also helps in inspecting quickly, fixing everything, even a minor one, before it appears as a major problem.

V. WORKING PRINCIPLE

The main principle behind the working of this model is edge-computing. It is a computing technique focused on bringing the source of data or the network's edge closer to computing to reduce bandwidth and latency. The principle is relatively simple and easy to understand. The implementation is also simple and is a collection of multiple technologies. The model has three main sections or development places.

Firstly, the hardware, most of the hardware used in this prototype is open in the market for purchase any time. But for configuring the hardware we had to use docker to etch an Operating system on to the Raspberry-pi in order to get Linux in place of Raspbian. Another location we had to program and configure was the UART receiver in order to check if proper data was being received.

The middle-ware for this project has to be the EdgexFoundry server. The server was established with simple commands. But the data to be processed into the server took a lot of code which was available for open-source. Once it was in and received into the EdgexFoundry server. The next and most major phase of the model, that is the application part.

The application layer of this project used many popular technologies and tools. The first part of the application layer was processing the data in separate SQLite DB's so that they can be used for particular applications. Based on the application the manipulations were done in the DB file. The DB file was sent to Jupyter for the use of classification applications. The final DB file was sent to a third-party software to display essential values in a mobile application.

VI. ADABOOST FOR OBD-II DATA CLASSIFICATION

The model that was implemented used a machine learning algorithm called Adaboost. We know that Ada-boost's specialty is that it takes multiple weak learners and boosts them to make a strong classifier of data. We are implementing the algorithm using sqlite3 database and Jupyter notebook. Sqlite3 gets the data from the EdgeX platform. This data is processed there using formulas which help in reducing the data to the form which we require. Currently we are doing 3 manipulations to process the data so that we can get a criterion to classify the data.

- Throttle position % $A * 100/255$

- Where throttle position is converted to percentage for easier calculation of changes in the throttle.
- Finding Speed rate, rpm rate, throttle change rate.
- Using this basic formula
- ●Find relative ratio throttle position and engine speed, relative ratio of the vehicle speed and engine speed.
- ●Using the following formulae, we are able to calculate the relative ratios. These are the boosted conditions for our algorithm.

$$R_{cz}(t) = \frac{cs(t)}{220} \div \frac{zs(t)}{8000}$$

●For vehicle speed where Rcz(t) is the relative ratio of the vehicle speed and engine speed, cs(t) is the vehicle speed at time t with maximum speed of 220, zs(t) is the engine speed at time t with maximum engine speed of 8000.

$$R_{jz}(t) = \frac{jq'(t)}{\max(jq')} \div \frac{zs'(t)}{\max(zs')}$$

●For throttle position where Rjz(t) is the relative ratio of throttle position and engine speed, jq'(t) is the change rate of throttle position at time t, zs'(t) is the change rate of engine speed at time t, max(jq') and max(zs') are the maximum change rate values of throttle position and engine speed, respectively
●The algorithm considers three conditions to classify the behavior to be good. The three conditions -are:

✓ The relative ratio of the vehicle speed and engine speed is maintained between 0.9 and 1.3.

✓ The relative ratio of the engine speed and throttle valve is maintained between 0.9 and 1.3.

✓ Finally, the engine load is maintained between 20% and 50%.

$$D(t) = \frac{data(t_2) - data(t_1)}{t_2 - t_1}$$

●This algorithm has already been implemented and has almost an accuracy above 90% in real scenarios. We have taken Jupyter and used python in place of the traditional GML toolbox using MATLAB.

VII. PROCEDURE OF THE MODEL

Step 1: Insert the OBD-II to DB9 Serial cable to the car's OBD port and the other part of the cable and set it up with the Sparkfun OBD-II UART Module.
Step 2: Now Check if the module is powered on by seeing the led.
Step 3: Connect the output of the UART module to the USB - TTL UART converter module and give the output to the Raspberry-PI.
Step 4: In the meantime, also connect the GPS module to the raspberry-pi
Step 5: You can power on the Raspberry-PI giving supply from the charging port of the car as it uses a normal micro-USB charger with 5V power supply.

Step 6: Once you have given all the connections and power supplies you can access the EdgexFoundry server on your laptop.
Step 7: Once you are on the server you can enable a virtual environment using a simple command.
Step 8: This should allow all the data to be collected in the server from the OBD-II port.
Step 9: The server analyzes the data and sends it to the three separate SQLite DB's in the server.
Step 10: The data is processed there and sent to the respective application parts.
Step 11: The data is sent to Jupyter Notebook for classification of driving behavior
Step 12: Similarly, for the case of Smart City data is processed and it stores the GPS coordinates of accident-prone zones.
Step 13: The last server sends a group of data to the Third-party software Openasapp to process and display a few values and GPS map of the last seen location of the vehicle.
Step 14: All of these applications happen simultaneously and are independent to each other.

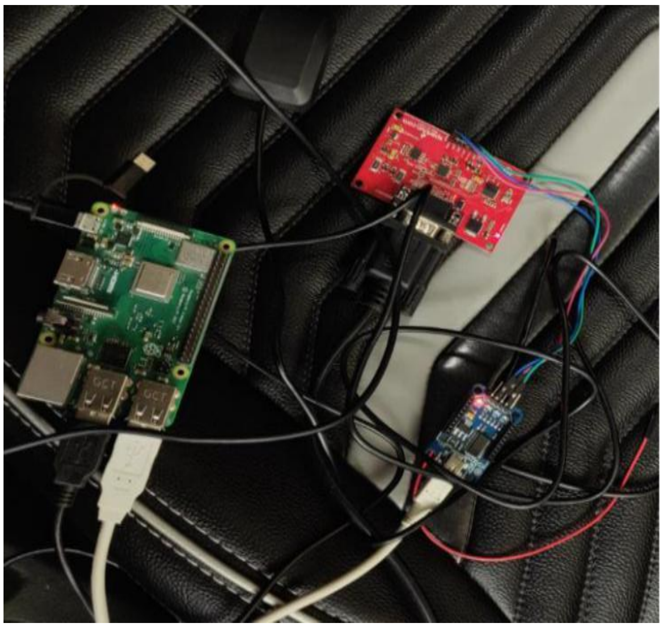
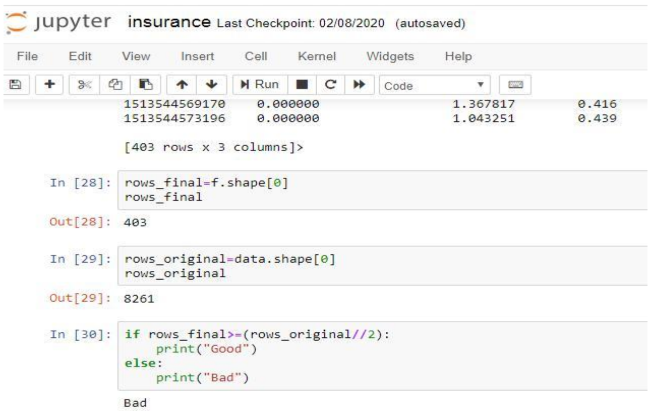


Fig 2: Circuit used for the model powered-on

VIII. RESULTS

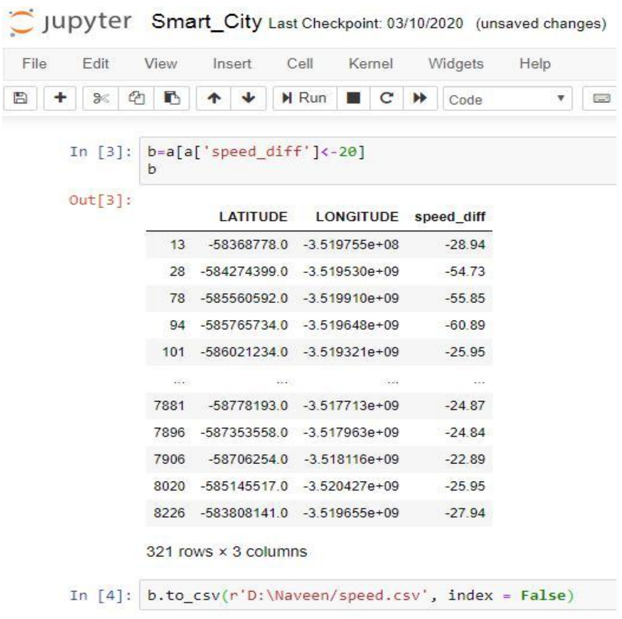
First, we see the output of the Jupyter Notebook output for both Smart city and Insurance applications to classify instances. As for the mobile application we don't use Jupyter notebook the data is sent directly to the application.

The mobile application outputs are given towards the end.


jupyter insurance Last Checkpoint: 02/08/2020 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

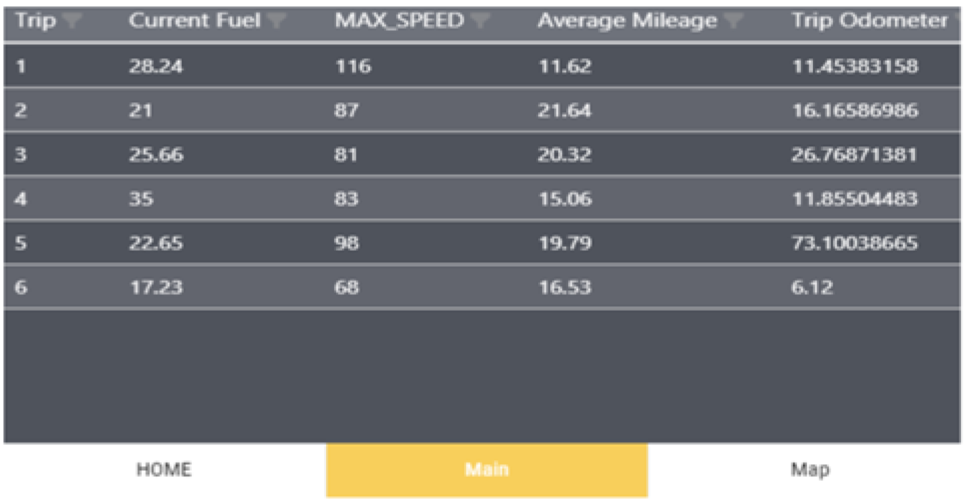
```
            1513544569170    0.000000              1.367817      0.416
            1513544573196    0.000000              1.043251      0.439

            [403 rows x 3 columns]>

In [28]:    rows_final=f.shape[0]
            rows_final

Out[28]:    403

In [29]:    rows_original=data.shape[0]
            rows_original

Out[29]:    8261

In [30]:    if rows_final>=(rows_original//2):
                print("Good")
            else:
                print("Bad")

            Bad
```

Fig 3: Output from Jupyter Notebook for Insurance


jupyter Smart_City Last Checkpoint: 03/10/2020 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [3]:   b=a[a['speed_diff']<-20]
          b

Out[3]:
                LATITUDE      LONGITUDE   speed_diff
            13   -58368778.0  -3.519755e+08   -28.94
            28  -584274399.0  -3.519530e+09   -54.73
            78  -585560592.0  -3.519910e+09   -55.85
            94  -585765734.0  -3.519648e+09   -60.89
           101  -586021234.0  -3.519321e+09   -25.95
           ...           ...            ...      ...
          7881   -58778193.0  -3.517713e+09   -24.87
          7896  -587353558.0  -3.517963e+09   -24.84
          7906   -58706254.0  -3.518116e+09   -22.89
          8020  -585145517.0  -3.520427e+09   -25.95
          8226  -583808141.0  -3.519655e+09   -27.94

          321 rows × 3 columns

In [4]:   b.to_csv(r'D:\Naveen/speed.csv', index = False)
```

Fig 4: Output from Jupyter Notebook for Smart City



Trip	Current Fuel	MAX_SPEED	Average Mileage	Trip Odometer
1	28.24	116	11.62	11.45383158
2	21	87	21.64	16.16586986
3	25.66	81	20.32	26.76871381
4	35	83	15.06	11.85504483
5	22.65	98	19.79	73.10038665
6	17.23	68	16.53	6.12

Fig 5: Output of all the datasets stored as a table

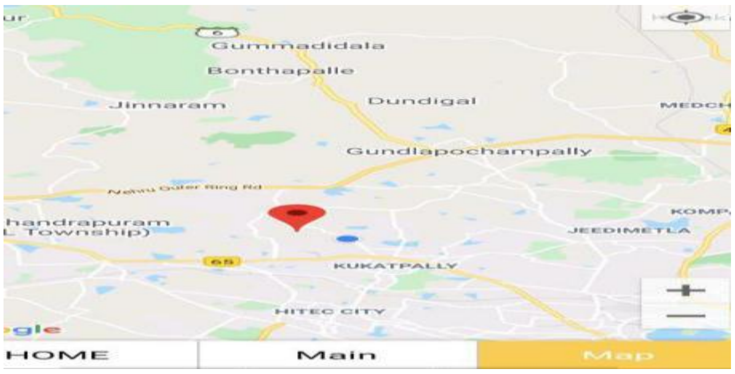


Fig 6: GPS location of last location of dataset-2

IX. FUTURE SCOPE OF EDGE COMPUTING IN AUTOMOTIVE INDUSTRY

Driving

Similar to how in this proposed model we classified OBD-II sensors data at edge. The edge can process more data from thousands of sensors. For an autonomous vehicle there are many cameras and Lidar sensors which are used for self-driving. If the data can be processed at the edge, the efficiency and response time will get better. If proper efficiency is done, we can be able to develop a level-5 fully autonomous vehicle.

Infotainment systems

Infotainment systems are the entertainment systems that are available on the car. With the ability to connect edge-computing to this system. We can try to understand the driver's interests and make the experience feel more specific to the user. It can be done with help of machine learning algorithms which can learn the tastes and interests of the user.

Electric vehicle battery maintenance Predictivity

By analyzing the driver's driving behavior and average usage of the car on a day-to-day basis. The edge-computing platform can be able to learn and predict the user's battery usage. This can help in making the battery maintenance better for the user and charging it for trips easier for the user.

Keyless car-entry

This application can be done with the help of multiple sensors such as fingerprint and camera. This can help the user with getting rid of the key and directly using the car. Even the ignition can also have a camera which can face-recognize the driver and on the ignition, making the driver truly independent from car keys.

Autonomous Parking

Similar to how autonomous driving is done. If we keep SONAR sensors on the back of the car it can automatically park itself. This can help the driver in tight parking situations. Also, it can be used to make the car get out from tight parking situations.

Monitoring and Alerting for the Car

This can be used for car rental applications. With the help of GPS sensor, we can achieve this. If the driver of the car were to rent a car for use in the city only. The car can notify the user that distance from the city and make sure he doesn't leave the city. This can also be used for security purposes to make sure the car doesn't leave a particular location in case of getting stolen.

Vehicle to Vehicle Communication

As the data is processed very fast it can help in V2V communication. The driver can communicate with drivers in a certain radius to make decisions in a few matters. Similarly, V2V also can play a major role in autonomous driving as it can communicate with other vehicles and make decisions.

Smart City

With tons of data that can be processed at a particular edge, there are no limits to what a city can do to increase the safety, roads and traffic situations with the help of edge-computing.

X. CONCLUSION

The model has been designed to provide a simple solution to show how edge-computing can be implemented in real-time. The first application was to classify driving behavior of a driver. This can be used in the insurance sector. The data is classified as good or bad driving. If the driving is good, it can be used to reduce the insurance premium. If it is bad, it can be used to increase the insurance premium. The algorithm we used for classifying the data was Ada-boost. The second application is to classify situations in case of accidents or bad roads. The change in speed rate over a definite time is calculated. If the speed rate is less than a certain value the GPS coordinates are stored into file. This application can be used for Smart Cities. It can be used to monitor traffic, condition of roads and also detect accident prone zones. If there is a cluster of vehicles connected to the city's server it can be used to classify locations on roads which are accident prone. The last and final application which was built was a mobile application for the user. It monitored the vehicle's stats. It gave the GPS coordinates of the vehicle's last seen location, Top speed, Average Mileage, Trip Odometer, and the current fuel level. The app was built by directly dumping the file into third-party software called Openasapp. Here the selected cells were identified and showed in the app. The app also has three pages. The first page is just a welcome page whereas the second page stores the records and the final page has the GPS location. For the app all the data was dumped into a single Database file and uploaded.

XI. REFERENCES

[1] Shi-Huang Chen, Jeng-Shyang Pan, and Kaixuan Lu “Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms”.

[2] https://link.springer.com/chapter/10.1007/978-981-13-1056-0_66

[3] https://github.com/vmware-samples/automotive-iot-samples

[4] http://www.motorsforum.com/

[5] Wang, Ruihu. "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review", Physics Procedia, 2012.

[6] https://components101.com/connectors/obd2

[7] https://en.m.wikipedia.org/

[8] ElkeAchtert, Christian Bohm, Hans-Peter Kriegel, Peer Kroger, Arthur Zimek. "On Exploring Complex Relationships of Correlation Clusters", 19th International Conference on Scientific and Statistical Database Management (SSDBM 2007), 2007

[9] https://www.pcworld.com/article/3396158/raspberry-pi-3-b-review.html

[10] https://wiki.edgexfoundry.org/

[11] Miyaji, M., Danno, M., Oguri, K.: Analysis of driver behavior based on traffic incidents for driver monitor
systems. In: Intelligent Vehicles Symposium, IEEE(2008) pp. 930-935.

[12] Liang, J., Cheng, X., Chen, X.: The research of car rear-end warning model based on mas and behavior. In: Power Electronics and Intelligent Transportation System, IEEE (2008), pp. 305-309 .

[13] Kishimoto Y, Oguri K. A modeling method for predicting driving behavior concerning with driver's past movements, IEEE International Conference on Vehicular Electronics and Safety (IEEE ICVES 2008), pp: 132-136.

[14] http://www.codeforge.com/article/239875 (GML AdaBoostMatlab Toolbox Manual)

[15] SAE International. On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual. Pennsylvania, 2003

[16] https://shodhganga.inflibnet.ac.in/bitstream/10603/81908/12/12_chapter_02.pdf

[17] https://learn.sparkfun.com/tutorials/getting-started-with-obd-ii/all

[18] https://gisgeography.com/trilateration-triangulation-gps/

[19] https://freelance.halfacree.co.uk/

[20] https://blogs.vmware.com/opensource/2019/03/28/exploring-open-source-iot-part-2/

[21] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility," IEEE Trans. Electron Devices, vol.ED-11, pp. 34-39, Jan. 1959.

[22] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, and M. Miller, "Rotation, scale, and translation resilient public watermarking for images," IEEE Trans. Image Process., vol. 10, no. 5, pp. 767-782, May 2001.