Mr Keerthivasan L P, 2025, 13:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

# Framework of Cloudcomputing Resource Scheduling Vehicle Fault Diagnosis

**Mr Keerthivasan L P, Assistant Professor Dr Kavitha P**
Department of Computer Applications
Vels Institute of Science Technology and Advanced Studies (VISTAS), Pallavaram, Chennai

**Abstract- Internet of Vehicles (IoVs) provides communication and computing resources, which makes the on-board diagnosis of vehicle faults possible. However, those resources need to be expanded to support the accurate analysis of the on-board diagnosis. Vehicular Cloud Computing (VCC) can solve the pressure of local vehicle processing but will cause an unavoidable delay. Thus, the accuracy and timeliness of on-board diagnosis cannot be guaranteed. To address the issue, we propose a Mobile Edge Caching based Resource Scheduling (MECRS) mechanism for the on-board diagnosis of vehicle faults. According to the urgency of vehicle fault diagnosis, we first design a cloud scheduling algorithm to meet the computation requirements of both the essential business of IoVs and the fault diagnosis. Subsequently, the priority allocation strategy is made for all four types of requests The urgent requests can be processed timely then. Specifically, Theproposed method is a multi-objective optimization method for allocating communication and computing resources for the above requests. We also present a large scale file mobile edge caching alr algorithm where the large scale file is cached at the mobile edge. Offloading the cloud with high popularity takes advantage of it to relieve the pressure of the cloud. Finally, we carry out comprehensive simulations. Results show that the developed mechanism has a high service rate for on board The performances of the other three essential services are not compromised.**

**Keywords- nternet of Vehicles (IoVs), Mobile Edge Caching, Resource Scheduling, Vehicular Cloud Computing (VCC), On-board Diagnosis, Vehicle Fault Detection, Multi-objective Optimization, Edge Computing, Priority Allocation Strategy, Cloud Offloading, Real-time Processing, Intelligent Transportation Systems.**

## I. INTRODUCTION

As more cars are used, a number of unforeseen issues occur while the vehicle is operating at high intensity overload. China, the US, and Japan will be the top three nations in terms of car ownership by 2023. China leads the group with 430 million.Automobiles, 18.21 million new energy vehicles, and 520 million drivers. The rise in car ownership contributes to traffic issues like gridlock, collisions, and pollution of the environment. Long-term load operation in automobiles can lead to a variety of issues. Effective car fault diagnosis is therefore crucial. However, anomalous performance and even security hazards are caused by the delayed diagnosis of vehicle faults Vehicle damage follows, endangering public safety. Consequently, with a prompt and precise car fault diagnosis operation, A linked car can efficiently ensure safety and receive problem notifications in real time. Conventional fault detection technologies, such as knowledge-based, signal-based, and model-based approaches, may be quicker and more effective depending on the expertise of the specialists. Subsequently, as computer technology advanced, three primary techniques emerged: online large data processing, reliability statistics, and signal processing. The advancement of big data technologies has made it possible to diagnose car problems more quickly and effectively. In light of big data technology, a car must process vast amounts of data gathered by sensors using local computer resources . However, more is needed than just the vehicles' local computing capacity.

When data volume rises, car fault diagnosis duties need to be handled quickly. To lessen the processing strain on nearby resources, this issue needs to be resolv edimmediately. This issue is thought to be resolved by vehicular cloud computing (VCC) technologies and vehicle-to-X (V2X) communication made possible by IoVs Those tasks that were not completed in a timely manner could be offloaded to the vehicular cloud using V2X

communication. The computing power and storage capacity of the cloud platform are immense. A car might then obtain the processed diagnosis results that VCC provides by uploading large amounts of data associated with those jobs. IoVs' high mobility causes sporadic connectivity, making it challenging to upload data and download findings . You should ignore requests that the cloud is unable to process quickly. To ensure the promptness and precision of vehicle problem diagnostics, there are two main obstacles.

Cloud computing has been used in numerous studies due to its strong computational and storage capabilities. Research on the relationship between cloud computing and car fault diagnosis is still somewhat limited. It can be difficult to schedule cloud resources to handle requests for vehicle faults.

The majority of studies on cloud resource allocation solely concentrate on one scheduling request and objective. The cloud platform, however, offers a wider range of services than just on-board diagnosis of car faults. Other crucial functions including network operation requests, emergency requests, and largescale file requests must be maintained. Therefore, there is positive research value for a multitasking scheduling method with complete indicators

In order to handle car fault demands, we expect the cloud platform will be able to reserve additional computing resources. Next, we must lessen the resource usage of additional queries on cloud, which could have an impact on their offerings. The research focus is on how to increase the service rate of car problem requests while maintaining the ability to service other requests as usual

A Multi-Task Scheduling Mechanism. Applying cloudcomputing to vehicle fault diagnosis cannot ignore the diversity of tasks on cloud platforms. We divide requests into four groups based on the Quality of Service (QoS) requirements, aiming to target various applications. This not only ensures sufficient processing capability for vehicle fault requests, but also ensures timely response to other tasks

A Priority Allocation Strategy. As a result, there are two categories of car problem diagnosis requests: those that are handled locally for emergence and those that are handled by the cloud for delay tolerance. It is possible to receive a fault diagnosis with great timeliness and precision. Additionally, we establish an emergency factor for the cloud emergency request that keeps its priority. Lastly, the priority policy is used to sort all requests that have been uploaded to the cloud platform.

A Mobile Edge Caching Algorithm. Optimizing resource allocation in the cloud to optimize system reward and decrease the volume of large-scale files downloaded from the cloud is an issue based on the allocation method. Next, we suggest a multi-objective optimization technique to effectively identify suboptimal alternatives, increasing the service rate while maintaining low complexity.

A comprehensive simulation. is carried out, which verifies our findings from both the service and user perspectives. According to simulation results, the service interval for car onboard diagnostic queries was extended by 100% while maintaining the functionality of other network-based services. We can even ensure a 50% service rate even in the case of a high vehicle density. Additionally, the ideal number of virtual machine configurations was provided for the specified simulation size, which has useful reference value in real life.

## II. LITERATURE SURVEY

**Title:** Framework Of Cloud Computing Resources Scheduling For Vehicle Fault Diagnosis Authors: W Gu, H Xu, L Zhu –
**Year:** 2024
**Description**: Vehicle defects can be diagnosed on-board thanks to the communication and computation capabilities offered by the Internet of Vehicles (IoVs). To enable the precise examination of the on-board diagnosis, those resources must be increased. The strain of local vehicle processing can be relieved by vehicular cloud computing (VCC),

although there will inevitably be a delay. Therefore, it is impossible to guarantee the precision and promptness of on-board diagnosis. We suggest a Mobile Edge Caching-based Resource to solve the problem.

**Keywords:**
•Vehicle cloud computing
•Resource scheduling
•Mobile edge caching
•Fault diagnosis

## III. RELATED WORK

Vehicle fault diagnosis has long been a priority of industry and research since driving safety is closely linked to the preservation of life and property. Numerous sophisticated algorithms are currently available for fault diagnosis. In for example, a new defect diagnosis method based on enhanced symplectic geometry mode de composition (SGMD) and optimized support vector machine (SVM) is introduced, demonstrating the method's efficacy and resilience in diagnosing faults in rotating machinery. Aojia et al. create a tracking controller that is fault-tolerant and a fault detector. To address the negative consequences of delay, they suggest a delay-dependent stability criterion. In the authors implemented output control for a roddriven vehicle and presented an active fault-tolerant control method for an underwater remotely operated vehicle. A fault tree analysis (FTA)-based fault detection method for electric vehicle charging devices is developed in literature and is capable of precisely identifying and promptly resolving charging device defects. Furthermore, For onboard applications in EVs, Paper suggests a soft SC fault diagnosis technique based on the extended Kalman filter (EKF). By modifying a gain matrix in response to real-time observed voltages, the EKF in the suggested method determines the state of charge (SOC) of the defective cell. It is reliable and useful for promptly identifying a soft SC failure.

Vehicle defects can in a variety of forms, though. Thus, there is an urgent need for a vehicle fault diagnosis method for large amounts of problem data.

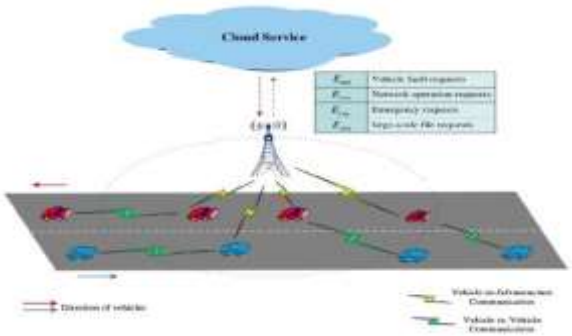
As big data technology develops, researchers suggest efficient detection techniques that can process larger amounts of data. Vehicle fault activities can be processed in real time with the help of online diagnosis. A brand-new intelligent in-car electrical power supply network is suggested in . This study shows that online problems are successfully identified, the power supply process for each device is suitably monitored, and the fault-tolerant approach can provide real-time protection and restoration. Following feature analysis and judgment, Zhang et al used BPNN for categorization and decision-making. A three-layer BP neural network structure was created by Liu et al to achieve the effective fusion of Fixed Detector Data (FDD) and Floating Car Data (FCD). Tian et al introduced a KNN-based bearing fault detection technique that uses spectral kurtosis and cross correlation to extract fault signals. One benefit of online diagnostics is its promptness. However, the local processor will be under stress from processing a lot of problematic services, and it must also handle other IoV services. Barabino et al. create an offline system that uses automatic vehicle position data to diagnose time reliability. Transit management can use this paradigm to conduct precise reliability analysisAn intelligent diagnosis technique for sensor intermittent faults based on a data-driven model was presented in Paper A speed sensor fault detection technique based on a learning-based data-driven approach in induction motor drive systems was proposed in paper . However, depending solely on the offline diagnostic makes it difficult to guarantee punctuality. It is impossible to obtain a multi-objective joint diagnosis in response to the requirement for additional local resources. To guarantee accuracy and punctuality, we seek a multi-objective optimization system. As a result, we combine offline and online techniques to upload and download data using VCC.

We send the local vehicle's time-sensitive demands for online processing. The cloud platform is used to send the processing results back to the car once the time-delay insensitive requests are uploaded there for offline processing.To improve the cloud's service capabilities, numerous cloud computing projects

have been completed. Additionally, the service rate can be raised with sensible resource scheduling. In order to increase resource usage, researchers from the University of Ottawa study the migration of virtual machines in cloud computing. A method for allocating resources that applies maxmin fairness to a variety of resource categories was presented. It was designed to cooperatively optimize time allocations in order to maximize the minimal energy balance among all users. In order to optimize the overall expected benefit of the linked vehicular cloud system, Zheng et al. suggest an efficient computing resource allocation scheme. The method achieves notable performance improvements within a manageable complexity range by representing the optimization problem as an infinite-level semi-Markov decision process. Dominant Resource with Bottlenecked Fairness (DRBF), a novel allocation technique, is proposed by Zhao et al. in their discussion of the equitable distribution of numerous resources. To guarantee that users in the same column receive an allocation in accordance with their fair portions, they separate users into distinct queues depending on their dominant resources. To solve the issue of cloud computing resource distribution and pricing in the mobile blockchain, a contract mechanism is used. and suggested an adverse selection contract to address the issue of information asymmetry. This paper focuses on the VCC architecture, which applies cloud computing to the IoV and then uploads fault tasks to the cloud. The volume of vehicle fault data is enormous. But requests in VCC are diverse, and the performances of other essential services cannot be compromised. Some common mechanisms, such as FCFS (First-Come-First-Served) mechanism, cause untimely service and huge resource waste. The vehicle fault diagnosis mechanism orienting towards multiple requests in VCC is closer to the actual scenario and solves the problem of poor communication and computing resources. Hence, an effective cloud resource allocation mechanism becomes the interest of this paper.

## IV. SYSTEM MODEL

The system model is explained in the parts that follow. We begin by outlining the communication scenario. Next, we go over the system reward function model, vehicle collaborative transmission model, traffic model, and overall optimization goal. Different kinds of requests are sent by vehicles to the central base station, which then transmits them to the cloud platform for processing. The vehicles will receive the processed results. The dotted circles show the base station's effective communication range because of the high dynamic topology of IoV. Direct communication between the base station and vehicles outside its range is not possible. Consequently, the transmission can be used for V2I (Vehicle to Infrastructure) communication range of the base station, and the V2V (Vehicle to Vehicle) communication can be performed outside the transmission range.




We consider a vehicular communication scenario that occurred within a base station and the cloud platform, as shown in Fig.1, including the forward driving and reverse driving vehicles. The user vehicles sends four types of tasks to the cloud platform and forwards them by the base station. The flowchart is shown in Figure 2. It represents the process from the vehicle sending the requests to the cloud platform processing the requests. Besides, the Table 1 is the list of important notations in this

### 1. Traffic Model

Vehicle fault requests are one of four categories into which we divide requests based on Quality of Service (QoS) needs. Eunit (concentrating on the typical driving of the vehicle), network operation requests Large-scale file requests, emergency requests (Eerg), and Econ (which focuses on network stability between the cloud platform and the vehicle) Efficient (concentrate on the driving

pleasure). The vehicles produce the Eunit, Econ, and EF ile. The cloud platform itself generates theEerg. The base station sends the three types of requests to the cloud platform after receiving them. The system reward is then calculated by the cloud platform, with the highest priority going to the largest reward request.

**2. Overall Optimization Objective**

As the optimization goal, we decide on the overall system reward. Additionally, incorporate into the reward function the VCC's processing, bandwidth, energy, and time consumption:

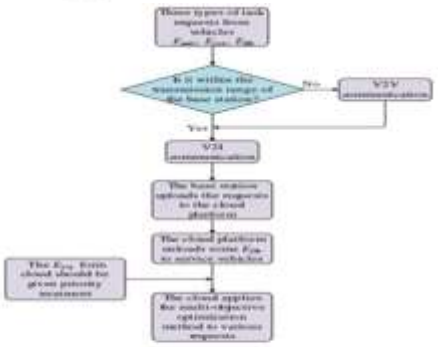
$$\max R(S,O) \qquad \text{(P1)}$$

$$\min C_{file} \qquad \text{(P2)}$$

$$\text{C1}: C_{unit} + C_{con} + C_{erg} + C_{file} < C,$$

$$\text{C2}: C_{unit} > C_{file},$$

$$\text{C3}: \sum_{i=1}^{Kj} Ei \le c \qquad \forall j \in K,$$



| | |
|---|---|
| | Calculation rate of per virtual machine |
| | Occupied bandwidth of per virtual machine |
| | Computation price of per virtual machine |
| | Bandwidth price of per virtual machine |
| | Energy consumption unit price with no cloud |
| | Time consumption unit price with no cloud |
| | The number of the requests served by cloud |
| | Maximum number of virtual machines in cloud |
| | User tolerance time for request $i$ |
| | The playout time required for large-scale request $i$ |
| | Data volume for request $i$ |
| | Popularity degree for request $i$ |
| | Energy consumption by no cloud for request $i$ |
| | Time consumption by no cloud for request $i$ |
| | Time from base station to cloud for request $i$ |
| | Transmission power from station to cloud for request $i$ |
| | Average received power in fading process for request $i$ |
| | Communication radius of per vehicle |
| | Speed of the target vehicle |
| | Speed of and the assisting vehicle |
| | time to maintain the vehicle relative motion |
| | Cache capacity of per vehicle |
| | Transmission rate between target and auxiliary vehicle |
| | Download rate of vehicle |
| | Viewing playout rate of large-scale files |
| | Busy length of the playout buffer when request $i$ arrives |
| | Free length of the playout buffer when request $i$ arrives |
| | Connect duration between user vehicle and cache vehicle |

where the reward function for the system is R(S, O). The greatest cloud computing resource is C. Together, the four tasks' total capacity ought to be less than C. C stands for the vehicle's capacity to store huge files in its cache. Requests for network operations, emergency situations, large-scale files, and vehicle faults, respectively, take up Cunit, Ccon, Cerg, and Cfile computer resources. By C2, we imply that Cunit should be as big as possible compared to Cfile. C3 is a representation of the limited cache capacity per vehicle. The maximum number of large-scale file requests that a vehicle can accommodate is denoted as Kj, where K is the set of cars, K = {1, 2, 3,..., j...}. The goal of this system is to maximize system reward while enabling more computing and communication resources for analyzing and transmitting vehicle faults. However, lowering the Ccon will significantly affect network stability, which will further lead to poor communication quality, while lowering the Cerg will ignore network crises. Additionally, when the Cfile decreased, customers' happiness with the driving experience decreased. However, cloud resources might not always be used in the Ef ile response to large-scale file demands. In order to minimize large-scale file service from the cloud, we therefore provide a mobile edge caching technique in section IV. B, which states that vehicle caching serves the large-scale file requests.

**3. System Reward Function Model**

Equation (2) illustrates how the difference between the gain function E(S, O) and the overhead function P(S, O) can be used to determine the reward function.

R(S, O) = E(S, O) − P(S, O) is the VCC platform's state set. The VCC platform's operation set is O = {0, 1, 2,..., NRU}. The event set is B = {Ra, Rl}. Ra stands for "request arrivals." Ra stands for "request leaves."

$$E(S,O) = \sum_{i=1}^{N_Q} e_i(S,O),$$

where the gain of one request handled by the cloud platform is denoted by ei(S, O). The advantages of each request should be combined, as this paper examined the system. Additionally, the primary reason of ei(S, O) is time and energy usage due to the cloud platform's strong resources and data storage capacity, as demonstrated by the following:

$$e_i(S,O) = \begin{cases} 0 & o_p = 0 \\ [\beta_c(E_{vi} - P_i\delta_i) + \beta_t(T_{vi} - \delta_i)] & o_p \neq 0. \end{cases}$$

The gain is zero when op = 0 since the car does not benefit from the cloud service.

The car benefits from the cloud service when op 6=0. As a result, it eliminates the massive energy use (EVI) and time consumption (Tvi) associated with standard internet downloading. However, the cloud platform procedure adds P δi and δi consumption. Furthermore, overhead function P (S, O) reflects the processing consumption of cloud platforms. The system overhead function for the requests, P (S, O), can be written as follows: It includes energy consumption, time consumption, and energy consumption due to channel fading.

$$P(S,O) = C(S,O)\tau(S,O) + \sum_{i=1}^{N_Q} \beta_c(P_i - \Omega_i) \cdot \delta_i,$$

where the overhead per unit of time is denoted by C(S, O). The time it takes for the cloud to serve the request is τ (S, O).

$$\sum_{i=1}^{N_Q} \beta_c(P_i - \Omega_i) \cdot \delta_i$$

Is the overhead brought on by fading channels. Furthermore, because the number of virtual machines allotted by the cloud platform determines C(S, O),

$$C(S,O) = \sum_{i=1}^{N_{RU}} i \cdot n_i \cdot (ru_{Bw} \cdot ru_{Com}^{P} + ru_{Bw}^{P})$$

We allot sufficient virtual machine resources to requests that can be handled in the user's tolerance period, therefore each request's processing time is

**Vehicle Collaborative Transmission Mode**

Vehicles can receive services outside of the transmission range thanks to the V2V communication method, which is an addition to the V2I communication method. It lessens the strain on computing and communication services. As a result, V2V turns into a strong tool for enhancing cloud services in VCC.

Here, we concentrate on two automobiles communicating with one another. Through multi-hop transmission, many cars can communicate with each other via V2V. Vehicles A and B are the target and assisting vehicles, respectively, if vehicle

A uses vehicle-to-vehicle (V2V) transmission to send a request from vehicle B. V2V communication can be separated into three scenarios based on the vehicle traffic scene: relative driving, opposite driving, and same direction driving between the target vehicle and the aiding vehicle. According to the IEEE 802.11p agreement, the vehicle communication equipment transmission radius is [50m, 500m], and the V2V transmission bandwidth is 100kB/s.

$$T_{v2v} = \begin{cases} \frac{2r_v}{v_{v1}+v_{v2}} & v_{v1} \cdot v_{v2} \leq 0 \\ \frac{2r_v}{|v_{v1}-v_{v2}|} & v_{v1} \cdot v_{v2} > 0, v_{v1} \neq v_{v2} \\ \min\{\frac{E_i}{\lambda}, T_{chg}\} & v_{v1} = v_{v2}, \end{cases}$$

The communication time between the vehicles when they are driving relative to one another, including opposite driving, is shown in the first segment. Additionally, the greatest communication distance between two vehicles is known as the communication range tangent. The communication duration for two cars moving in the same direction but not stationary is expressed in the second segment, and the third segment displays the communication duration for the vehicles that are comparatively stationary.

The two cars can now continue communicating with one another until the request is fulfilled or the relative motion varies. Furthermore, this model can be used to download huge files from the cloud to automobiles.

## V. RESOURCE ALLOCATION FOR VCC

We suggest a Resource Scheduling (MECRS) technique based on Mobile Edge Caching to address the objective function optimization challenge. These three sections make up its content.

**1. Priority Allocation Strategy**

We upload some car defect queries to the cloud platform since we don't have enough computer power locally. The cloud platform ranks the four categories of requests in order of user tolerance time, data volume, and request popularity.

The three primary components of this approach are as follows:

**Local computing**

We can select either local computing or cloud computing for the Eunit based on the relationship between the service tolerance time Tt and the time threshold Ts. It falls under the emergency vehicle fault request when Tt < Ts. To guarantee prompt processing, the requests should now be handled locally in the car.

This request is not time-sensitive when Tt > Ts. In this instance, the services ought to be uploaded to the cloud for processing.

**Emergency response**

All services must use computing resources, with the exception of the local computing component. Because of their urgency, Cloud should respond to Eerg's request first. For that, we set an emergency factor ζ. ζ = 1 ensures preferential processing when Eerg occurs.

$$\zeta = \begin{cases} 1 & E_{erg} \text{ occurs} \\ 0 & \text{no } E_{erg} \text{ occurs.} \end{cases}$$

**Priority computing**

All requests submitted to the cloud must have access to sufficient cloud computing resources. We will lose out on some high reward requests if we solely adhere to the first come, first served premise because vehicle nodes move quickly. As a result, we create priority rules. The following is the priority equation:

$$P_{ri} = (\omega_1/T_{ti} + \omega_2/E_i) \times N_{ri}$$
$$s.t. \qquad \omega_1 + \omega 2 = 1,$$

where ω1 and ω2 stand for the weights of data volume and user tolerance time, respectively. Furthermore, various requests require different amounts of resources. For instance, the computation-intensive files need processing speed and virtual machine memory, while the communicationintensive files concentrate on the requirement for network bandwidth resources. Therefore, ω1 and ω2 are utilized to modify how much emphasis is placed on communication or computation resources. Under the same circumstances, it is evident that the processing priority increases with the shorter user tolerance time, the lower request data amount, and the higher request popularity. When the cloud platform receives several similar requests (such the precise automotive part defect, downloading the same enormous file, etc.), this request is very common. The request popularity degree Nri is the most crucial of the three priority factors as, when receiving the request, the cloud verifies that there are precise requests before deciding on a scheduling strategy. It is possible to acquire further identical requests from V2V if the most popular request is chosen and fulfilled.

**2. Mobile Edge Caching Algorithm**

We categorize the cloud requests into four groups, as was previously described. Our goal is to promptly respond to requests for car fault diagnosis while using the least amount of network and processing power possible. We anticipate that more cloud resources will be available to handle the Eunit in addition to the locally processed requests. The cloud platform, however, is unable to target a single request type. How can we be sure that the response to the other three kinds of inquiries is the issue to be resolved? Lowering the Ccon will significantly affect network stability, which will further lead to poor communication quality, while lowering the Cerg will ignore network crises. The cloud must handle these two kinds of requests. In order to reduce the service of large-scale files from the cloud, we therefore suggest a mobile edge caching technique that is targeted at efficiency.

$$\min C_{file}$$
$$s.t. \sum_{i=1}^{K_j} E_i \leq c \qquad \forall j \in K.$$

Vehicles are separated into two categories: service vehicles and requesting vehicles. Each vehicle can, of course, be a server or a requester. According to

certain guidelines, we pre-cache huge files onto the service vehicle. Large-scale file services from the cloud can be reduced by using the V2V transmission technique, which allows the requesting vehicle to download large-scale files straight from the service vehicle.

This approach can guarantee Ef ile's service rate, lessen the cloud platform's processing load, and free up additional processing power for Ef ile. Since caching material at the edge of mobile networks is a promising solution to the data tsunami , we have chosen to use the edge caching method to serve largescale file requests in the resource allocation mechanism. It is believed that there will be little temporal overlap when many vehicles have the same storage contents. This indicates that the same content is being stored in cars. Until the transmission distance causes it to be halted, the user car just needs to issue a request to a service vehicle. This paper queues based on M/D/1 in the resource buffer pool. E[rp], the file playback rate, is the service rate. The arrival rate is λp, and the transmissive huge file data size is E[Y].

$$\lambda_p = \lambda \cdot x_i,$$

$$E[Y] = E[D] \cdot E[r_h],$$

where λ denotes the communication speed between the target vehicle and the auxiliary vehicle, or the task arrival rate, E[D] denotes the transmission time between user vehicles and service vehicles, E[rh] denotes the download rate of vehicles, and xi denotes the quantity of identical files downloaded simultaneously from the collaborative vehicle. It is evident that the queue's service rate:

$$\rho = \lambda_p \cdot \frac{E[Y]}{E[r_p]} = \lambda \cdot x_i \cdot E[D] \cdot \frac{E[r_h]}{E[r_p]},$$

where E[rp] is the watching playout rate of large-scale files and ρ is the queue's longterm usage. Pi is the queue's free probability at the time of the large-scale file request i.

$$P_i = \frac{E[I_i]}{E[B_i] + E[I_i]} = 1 - \rho,$$

where the busy length of the playout buffer upon arrival of request I is represented by E[Bi], and the free length of the playout buffer is represented by E[Ii].

The transfer time is prolonged because our model handles huge files. Next, keep in mind that Cf ile is equal to the total number of bytes that were downloaded from the cloud. We own that:
From the above derivation, we can get the following formula:

$$C_{file} = E_i \cdot (1 - \lambda \cdot E[D] \cdot \frac{E[r_h]}{E[r_p]})$$

The calculation above shows that, once the other factors are established, the Cf file mostly relies on the file's attribute, which is the Ei Nevertheless, we prioritize largescale file requests, which are unavoidable for massive Ei. As a result, we investigate document popularity, another file attribute. The demand for documents is referred to as popularity. Many individuals can download such large-scale files due to their high popularity. We suggest a greedy algorithm based on popularity in section

## V. GLOBAL OPTIMIZATION METHOD

Optimizing resource allocation in the cloud to optimize system reward and decrease the volume of large-scale files downloaded from the cloud is an issue that arises based on the allocation technique. Next, we suggest a multiobjective optimization technique to effectively identify suboptimal alternatives, increasing the service rate while maintaining low complexity.

P1 is receiving the highest system reward, as demonstrated by algorithm 1. P2 is reducing the amount of processing power used by huge files. P1 and P2 are both quite computationally complex. Therefore, we suggest a greedy two-phase resource allocation approach to effectively identify low-

complexity sub-optimal solutions for Problems 1 and 2. As illustrated in Algorithm 1, P1 and P2 are solved in Phase 1 (Algorithm 2) and Phase 2 (Algorithm 3), respectively.

Algorithm 1 is broken down into Algorithms 2 and 3. Algorithm 2 uses greedy algorithms to distribute virtual machine resources across different request kinds and organizes requests according to priority and system reward functions. Depending on the file popularity,

Algorithm 3 avariciously caches the user's vehicle's substantial file requirements into the service vehicle. Bubble sorting, the algorithm's sorting technique, has a lower temporal complexity than fast sorting and other techniques. The Greedy algorithm's simplicity and low temporal complexity are its merits. The Greedy method, which has an O(n 2) complexity, is improved upon in this article. In order for each optimization result to reduce the problem to a smaller sub-problem, this research focuses on efficiently allocating the available cloud computing resources. Furthermore, if the requests are handled via the FCFS technique, computer resources cannot be used efficiently due to the cloud platform's limited bandwidth, computational load, and varying tolerance times, data volumes, and popularity for distinct requests.

In order to make resource allocation more thorough and rational, this work enhances the request priority while choosing the best solution of sub-problems, given the greedy algorithm's simplicity and low time complexity.The algorithm breaks down the resource optimization challenge for cloud platforms into a few sub-selection issues. Every sub-selection simultaneously maximizes the present advantage, resulting in a local optimal solution. The choice is made once more to find the best local solution for the new subproblem whenever the cloud platform releases fresh resources.

**Algorithm 1:** Two-Phase Greedy Resource Allocation Algorithm 1)

**Algorithm 2:** Priority-based Dynamic Greedy

Algorithm
**Input :** 1 [num_request]
**Output** : Sorted I [max_num_request]
**Initialization :**

**Begin**
  Initialize parameters, $K = \{1, 2, 3, \ldots, j \ldots\}$;
  **For** $j \in K$ to $num\_request$
    Calculate corresponding $R(S,O)$ , $P_{ri}$ ;
  **End For**
  **For** $i = 1$ to $num\_request - 1$
    **If** $R(S,O)_i < R(S,O)_{i+1}$ **Then**
      Exchange $I[i]$ and $I[i+1]$ ;
    **End If**
    **If** $R(S,O)_i == R(S,O)_{i+1}$ **Then**
      **If** $P_{ri} < P_{r(i+1)}$ **Then**
        Exchange $I[i]$ and $I[i+1]$
      **End If**
    **End If**
  **End For**
Allocate VMs in the sequence of $I$ $[max\_num\_request]$
**End**

**Algorithm 3** : Popularity-based Caching Greedy Algorithm
**Input**: 1 [f ile_request], c
**Output**: Sorted I [f ile_max_popular]
**Initialization**:

**Begin**
  Initialize parameters, ;
  **For** $j \in K$ to $file\_request$
    **If** $N_{r[j]} < N_{r[j+1]}$ **Then**
      Exchange $I[j]$ and $I[j+1]$ ;
    **End If**
  **End For**
Vehicle cache in the sequence of $I$ $[file\_max\_popular]$
  **For** $j = 1$ to $file\_request$
    **while** $\sum_1^j E_{vi[i]} + E_{vi[most\_popular]} < c$
      $j \leftarrow j + 1$
  **End For**
**End**

In the real scenario, the vehicle would continuously reach the base station communication range and send requests, as described by the repeated process. Consequently, the cloud platform's processing and bandwidth resources can be efficiently optimized to maximize the system reward by employing a prioritybased dynamic greedy algorithm.

In particular, for every request that the cloud platform receives, the algorithm first determines the system rewards. Sort by prize level after that. An overall ordered task queue will arise if all of the requests have the same reward and are ranked from high to low in terms of priority. Then, beginning at the top of the task queue, each attempt is made to assign enough virtual machines

to satisfy the amount of bandwidth and computation demand. If the optimization measure outperforms other requests, the allocation is finished. If not, the task queue's subsequent request might be chosen. Algorithm 2 below illustrates the concept of the algorithm.

During the waiting phase, the cloud platform gathers requests, and during the decision period, it makes decisions. The cloud platform consistently serves highpriority and large-reward requests, and the pseudocode depicts a single resource allocation strategy of the VCC during the decision time.The procedure is repeated during the subsequent decision time after the service requests have been gathered. A dynamic scheduling system for cloud platform resources is created since the time separation between neighboring periods is limited. is the collection of cars, and since the duration of both periods is brief, a dynamic cloud platform resource scheduling selection procedure is formed. Additionally, I [max_num_request] is the arranged request set, K is the set of cars, and I [num_request] denotes the set of requests gathered by the cloud platform while awaiting a decision.

In Algorithm 3, we provide a caching greedy algorithm based on popularity to solve the P2. The most popular material in each service vehicle is replicated by this algorithm. Next, the Large-scale files are avariciously crammed into the vehicle's remaining storage, arranged from high to low popularity. In this manner, the service vehicle efficiently offloads the massive file requests. Consequently, the strain on cloud resources to handle numerous requests can be reduced. As a result, more computer power is available for Ef ile, increasing the Ef ile service rate. Furthermore, Evi[i] represents the file size, Nr[i] indicates the efficiency popularity level, and c is the vehicle's cache capacity for huge files.

## VI. SIMULATION AND ANALYSIS

We perform extensive simulations in this section. Four service rates of successful requests are used as metrics to assess the effectiveness of the suggested approach. Both the service side and the user side are used to display MECRS. Prior to comparing the rates under various request proportions, we first show how the four requests service rates vary with vehicle density. Lastly, MECRS optimality is demonstrated by how the service rate varies with cloud virtual machine density. We model the communication scenarios depicted in Figure 1, in which M cars are outside the base station's range and N vehicles are inside the coverage area of a single cellular base station. Vehicles are separated into relative and opposite driving, and the road has a single lane in both directions. Three different kinds of requests are sent by the user cars to the cloud platform, where the base station transmits them. The cloud platform itself is the source of the additional Eerg requests.

**A:Simulation Setup**

| Parameters | Value |
|---|---|
| Calculation rate of per virtual machine | 3000 MIPS |
| Occupied bandwidth of per virtual machine | 40 Mbps |
| Computation price of per virtual machine | 1.1 yuan/hour |
| Bandwidth price of per virtual machine | 0.8 yuan/hour |
| Energy consumption unit price with no cloud | 1.2 yuan/hour |
| Time consumption unit price with no cloud | 0.9 yuan/hour |
| The number of virtual machines | [0, 500] |
| The number of vehicles | [0, 400] |
| Transmission power from base station to cloud | 100 W |
| Average received power in the fading process | 20 W |
| Data volume processed by per virtual machine | 6000 bytes |

MATLAB is used to carry out the simulation analysis task. To model the requests for cars to the cloud, we create a simulation tool. We placed infrastructure at 5,000 meters in the middle of the 10,000-meter-long road. There should be fifty meters between each car. Usually, the vehicle accelerates in a scattered manner. Additionally, when there is less than the safe distance between two cars, real-time speed adjustment is implemented.

Regarding the network environment's specifications, we use the IEEE 802.11p protocol, which has a V2V transmission bandwidth of 100 kB/s and an onboard communication equipment transmission radius of [50, 500 m]. In this case, we'll

assume that big files are ranked from 1 to 10, where 1 represents the least popular and 10 represents the most popular. Next, we assume that the cloud platform has not yet reached saturation and that there are always computing resources available. Table 2 displays the simulation parameters. The cost per virtual machine is the same as the cost of public cloud services offered by Amazon.

There are two sections to the simulation experiment. This study illustrates the effectiveness of the MECRS mechanism from the viewpoints of the user and the server using four request success service rate evaluation criteria. The percentage of requests that are successfully answered relative to the total number of requests is how we define the service rate. The trajectory of successful service ratios for four distinct services under various mechanisms as vehicle density rose is first compared in the user-side experiment analysis.

Next, we show the variations in service ratios for four distinct requests in varying amounts. Lastly, we confirm the efficacy of the cloud platform-assisted car defect diagnosis processing using the actual data that was gathered.

As the vehicle density rose, we compared the virtual machine occupancy under various computing resource techniques in the server-side experimental analysis.

To prevent resource waste, the ideal number of virtual machines is offered.

**2. Compare Mechanism**

We take into account and contrast the results using the allocation policies listed below:

- **MECRS**: In order to properly utilize local resources, the approach suggested in this research separates local processing from cloud processing. Next, an optimal resource scheduling strategy with several objectives is used.
- **Random:** The car caches huge files at random and uploads all fault requests to the cloud for processing.
- **No-Caching:** The cloud is used to process all car fault queries; the vehicle edge caching technique is not used.
- **FCFS:** The cloud follows the principle of first come, first served when responding to all queries.

**3. Analysis for User Side**

Every second, we force every car to send a request. Vehicles beyond the effective communication range can use V2V to send data over a number of hops. The quantity of cloud virtual machines (VMs)

200 machines), with a 5:2:2:5 ratio of four requests. Our assumptions are that Eerg is proportional to Econ and Eunit is proportional to Ef ile. Eunit and Ef ile are sent more regularly, of course. This paper's goal is to optimize the system reward function, which is mostly established by the overhead and gain functions in Equation (2). Furthermore, this research makes the assumption that channel fading loss is not taken into account. Figure 3 shows that although Econ and Erg requests are fulfilled promptly, the first three methods have the potential to prolong the time it takes for Eunit to receive a 100% service. But if the number of vehicles increases, under the Random and No-

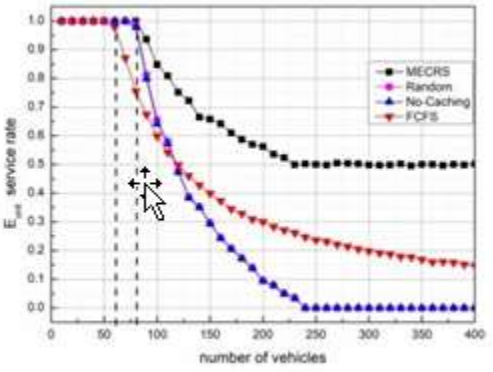
**Caching Mechanisms:**

Because category Eerg and Econ have a higher priority, Eunit does not have enough virtual machines. Because of this, the vehicle fault service rate is even lower than that of the FCFS mechanism (Fig. 3(d) shows the same explanation for the lower Ef EF service rate of the NoCaching technique). To address this issue, we then provide a popularity-based caching greedy algorithm and a prioritybased approach.

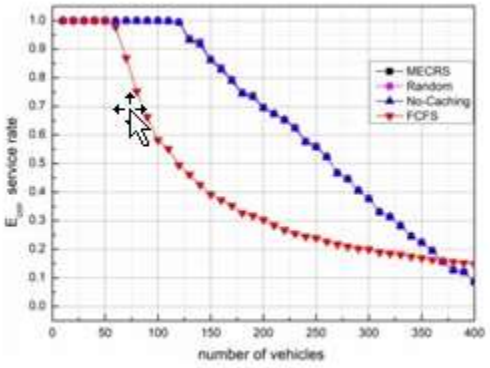
When compared to alternative methods, Fig. 3(a) demonstrates how well the MECRS mechanism can increase the service rate of Ef ile. Sequentially, we can provide a 50% service rate even in the case of a high vehicle density. The emergency factor we choose allows Eerg to be reacted to preferentially in Fig. 3(b)(c). The Econ has a higher service rate to guarantee network security and stability under the 100 percent Eerg service rate premise. The MECRS

mechanism enhances the Efile service rate, as seen in Fig. 3(d). There is a minimum service rate for the number of cars, N=120. This is because the cloud platform uses a lot of resources for other requests with a higher priority, and the cached file capacity is limited by the number of vehicles.

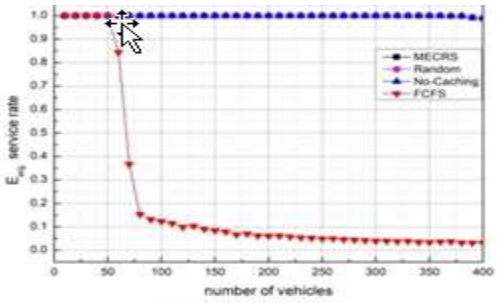
As a result, the lowest value is displayed. The overall buffer capacity rises with vehicle density, which causes the service rate to grow before leveling off. The cloud resources are now the limiting factor since the service rate for users retrieving huge files from the cache cars has reached its maximum.The service rates of Eunit and Ef ile at various ratios are displayed in Fig. 4. Both are in the same proportion in the first. The Efficient Weight Ratio comes in second. Eunit weight ratio comes in third



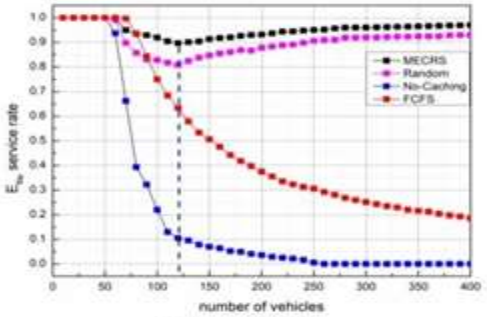

(a) $E_{unit}$ service rate.




(b) $E_{com}$ service rate.

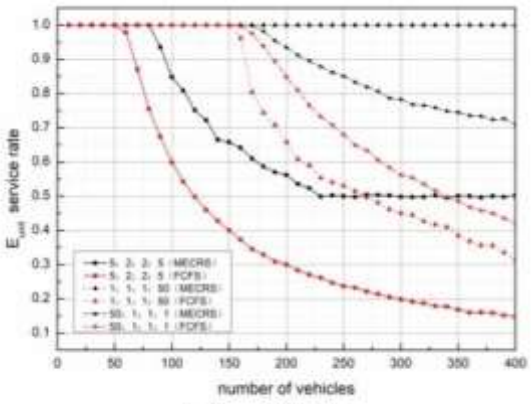



(c) $E_{org}$ service rate.

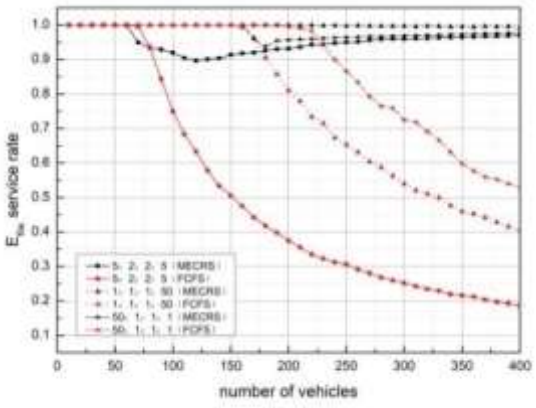



(d) $E_{file}$ service rate.

FIGURE 3 :
Service rate of four requests vs.number of vehicles, VMs = 200,proportion of four types is 5:2:2:5.




(a) $E_{unit}$ service rate.




(b) $E_{file}$ service rate.

FIGUIRE 4:
Service rate in different proportion vs number of vehicles, VMs=200.

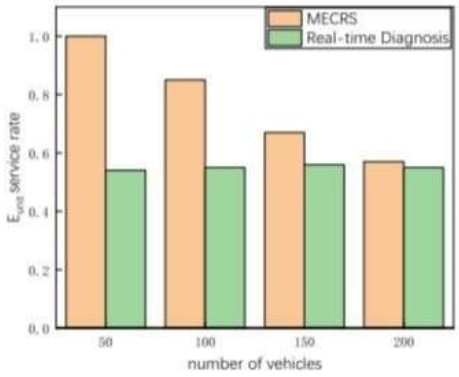



FIGURE 5 : Service rate under different mechanisms

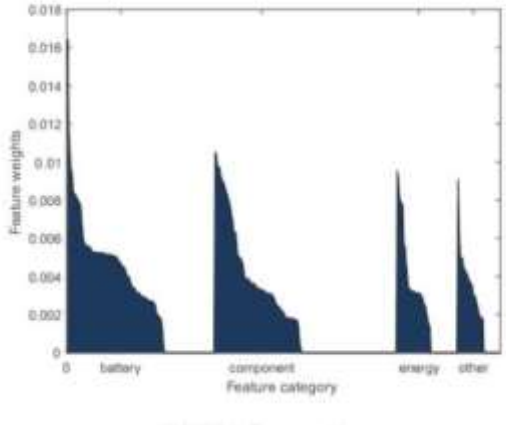



FIGURE 6 : Feature weights

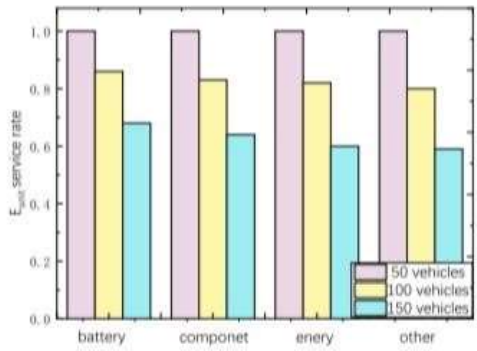
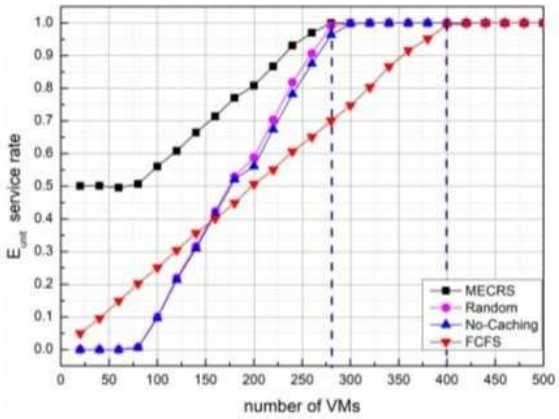


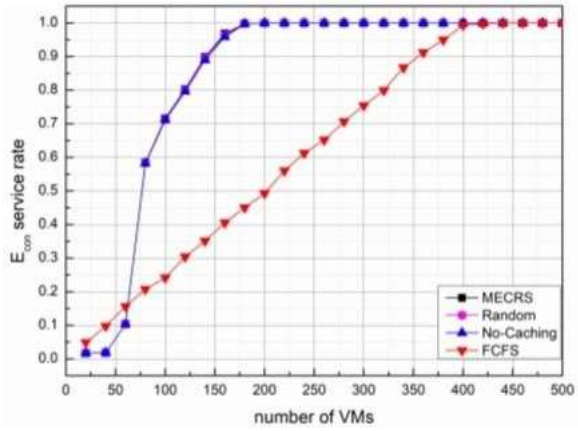

FIGURE 7 : Service rate ofvechicle features
Because of the decrease in Eunit, the local processing capacity is enough under the 1: 1: 1: 50 ratio and may achieve a 100% service rate. Additionally, with the boost in efficiency, all of the car's cache files have been used up, allowing the service rate to hit 99 percent. The reason for the remaining 1% is that none of the nearby cars have such a file. A service failure will occur if this file, which needs to be downloaded from the cloud, cannot be successfully delivered in the allotted time. Additionally, as the request density has increased, the Eunit service rate and the successful service rate have also increased under the 50: 1: 1: 1 ratio. In the meantime, the service rate improves when the efficiency declines and there is less competition with enough cache space. In conclusion, for different request ratios, the suggested mechanism can raise the service request rate. The majority of problem data is gathered and processed online by conventional car fault diagnosis algorithms. This can guarantee that tasks are completed on time, but it will result in the local processor in cars having insufficient processing power.

Particularly in light of the numerous other kinds of jobs that need local processing in cars, including autonomous driving, road section alerts, etc. Thus, a fault task offloading system is proposed in this study, which retains some of the vehicle failure duties locally and transfers others to the cloud platform for processing. Section IV goes into further detail. A vehiclemounted defect diagnosis system with less data and minimal computational complexity is proposed by V.

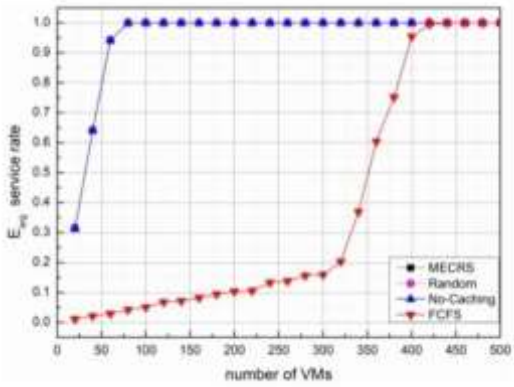
A.Literature storage, in order to monitor the status of the vehicle in real time. It confirms that the suggested diagnosis method is accurate, sophisticated, and has enough storage space. Electronic vehicle management provided 466688 samples for this study, of which 177261 are defective. In conclusion, the actual data capacity is 6.52 GB, and there are 539 original characteristics. Some data errors and omissions may inevitably occur during the dataset collection procedure. As a result, preprocessing the data is required prior to simulation verification. Effective real data yields more valuable and enlightening experimental outcomes. The three primary phases are missing value processing, data integration, and the removal of superfluous attributes.v



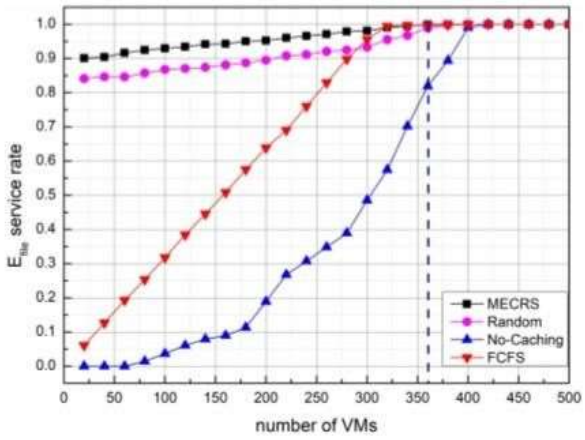

**(a)E UNIT service rate:**




**(b)E CON service rate:**




**(c)E ERG service rate:**




**(d)E FILE service rate**

Certain fault tasks with high latency requirements may not be processed in a timely manner if the local computing power is inadequate or heavily occupied by other processes. The vehicle has now driven outside of the V2I transmission range and is unable to transmit via V2V, resulting in resource waste. The MECRS method can increase the service rate of Ef ile when local resources are insufficient, as demonstrated by the simulation results in Fig. 5, assuming that only 20% of the vehicle's CPU resources are used.

Additionally, as the vehicle density rises, it eventually falls. Furthermore, vehicle density has no effect on the real-time diagnosing mechanism. Each feature's weight is displayed in Fig. 6. As we can see, features pertaining to batteries carry the largest weights, whereas features pertaining to vehicle components rank second in importance. The energy aspects come in third. The term "other features" refers to allofthe remaining characteristics. To validate the mechanism suggested in this paper, we used these four feature categories. Figure 7 shows that all four categories of failure data were successfully addressed. A 100% response rate can be attained at low vehicle densities.

## 4. Analysis for Service Side

Every one second, we force every car to send requests. There are 200 vehicles, and the percentage of four requests is 5:2:2:5.

Figure 8 shows that the service rates of four requests improve with increasing virtual machine density.

High priority Econ and Eerg take up the majority of virtual machines under the Random and No-Caching procedures when virtual machine resources are limited, which lowers the efficient service rate even further compared to the FCFS mechanism. But when the number of virtual machines rises to 160, it gets much better. We successfully raised the Eunit service rate under the Econ and Eerg's prompt responses. On the one hand, we guarantee that all requests be fulfilled, provided that the virtual machines (280– 400) are not overloaded. On the other hand, vehicle local processing can also ensure a 50% service rate when the number of virtual machines (20–80) is insufficient. Because the other three requests in Fig. 8(d) have a higher priority than large-scale file requests, the efficient service rate under the no-caching mechanism is the lowest. Therefore, in order to address the issue of poor file service rate, we employ the Popularitybased edge caching approach. The efficiency service rate is clearly increased by the suggested MRCRS mechanism, and the other two requests are likewise promptly handled.

Furthermore, it is evident that 360 virtual computers is the ideal number when all needs are fully fulfilled. Nonetheless, the first three categories of requests have achieved a 100% service rate at 260 units. The efficient service rate does not rise much between 260 and 360. Therefore, from an economic standpoint, there should be roughly 260 virtual machines on the cloud platform, which has real-world use.

In this project, a secure cloud storage system was designed and implemented to address the growing concerns surrounding data privacy, integrity, and user confidentiality in modern digital environments. By incorporating advanced encryption techniques such as AES for data encryption and RSA for key management, along with SHA-256 hashing for integrity verification, the system ensures that user data remains confidential and tamper-proof throughout its lifecycle.

The integration of secure user authentication mechanisms and access control features further enhances the trustworthiness of the platform, minimizing risks related to unauthorized access and potential data breaches. Leveraging lightweight and scalable technologies like Python, Flask, and SQLite, the system offers a modular, user-friendly, and efficient solution suitable for small to medium-scale deployments.

Future work may include scaling the system for distributed cloud environments, integrating biometric authentication, and ensuring compliance with broader data privacy regulations such as GDPR and CCPA.

## VII. CONCLUSION AND FUTURE WORD

In order to diagnose on-board vehicle faults, we suggested a computational resource allocation technique in this article. For each of the four request categories, we first created a priority allocation plan based on the diagnosis service's tolerance time.

Then, in order to optimize the VCC system's long-term reward, we developed a priority allocation mechanism taking into account the QoS features of requests. Furthermore, a widely used mobile edge caching strategy was put forth to relieve the strain on the cloud by offloading massive file requests. In order to distribute computational and communication resources, the multi-objective optimization resource scheduling approach was put out. Ultimately, the simulation results showed that the Eunit 100% serviced interval was extended while maintaining the promptness of the other requests, and we could ensure a 50% service rate even in the case of a high vehicle density. It can increase the service rate in the event of low resources when compared to the real-time diagnosis mechanism. This will greatly lessen the impact that vehicle fault has on traffic. Additionally, 260 was determined to be the ideal number of virtual machine configurations for the specified simulation scale from an economic standpoint; this figure has real-world application. Vehicle fault diagnosis research is highly significant and can successfully prevent a

number of traffic issues, including trip plans, traffic accidents, and traffic congestion. A vast knowledge base about car faults can be created by employing knowledge graph technology to associate and mine the gathered data. Furthermore, car defect diagnosis issues can be resolved more successfully because to cloud computing's strong processing and storage capabilities, opening up new avenues for future research.

## REFERENCES

1. Defect Diagnosis of an Autonomous Vehicle With an Improved SVM Algorithm Subject to Unbalanced Datasets," by Q. Shi and H. Zhang, appeared in IEEE Transactions on Industrial Electronics, vol. 68, no. 7, July 2021, pp. 6248-6256.
2. IEEE Transactions on Instrumentation and Measurement, vol. 71, pp. 1–10, 2022, Art no. 3504410, "A Learning-Based Method for Speed Sensor Fault Diagnosis of Induction Motor Drive Systems," by Y.
3. Xia, Y. Xu, B. Gou, and Q. Deng.
4. Real-Time Fault Diagnosis for EVs With Multilabel Feature Selection and Sliding Window Control," by L. Zhu, Y. Zhou, R. Jia, W. Gu, T. H. Luan, and M. Li, was published in the IEEE Internet of Things Journal on October 1, 2022, in volume 9, issue 19, pages 18346–18359.
5. IEEE Transactions on Power Electronics, vol. 38, no. 3, pp. 2861-2865, March 2023; K. Zhang, B. Gou, W. Xiong, and X. Feng, "An Online Diagnosis Method for Sensor Intermittent Fault Based on Data-Driven Model," pp.
6. IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 128–138, January– March 2023; J. Wang, J. Li, Z. Gao, Z. Han, C. Qiu, and X. Wang, "Resource Management and Pricing for Cloud Computing Based Mobile Blockchain With Pooling,"