SPECTRUM OF ENGINEERING SCIENCES
ISSN (E) 3007-3138 (P) 3007-312X VOLUME 4, ISSUE 6, 2026

REAL-TIME VEHICLE TRACKING IN A BIG DATA ENVIRONMENT

Shahabuddin[1], Abdul Haseeb Malik[2], Muhammad Numan[3], Qazi Ejaz Ali[4], Waheed Ur Rehman[5], Najeebullah[*6], Inam Ullah[7]

[1,2,3,4,5,*6,7]*Department of Computer Science, University of Peshawar, 25120, Pakistan*

[*6]najeebullah9172@uop.edu.pk

DOI:https://doi.org/10.5281/zenodo.20773363

Keywords
Big data; big data analytics; participating vehicles (PV); non-participating vehicles (NPV); query vehicle data; Apache Kafka; Apache Storm; real-time tracking; stream processing.

Article History
Received: 23 April 2026
Accepted: 02 June 2026
Published: 20 June 2026

Copyright @Author
Corresponding Author: *
Najeebullah

Abstract
With the development of the smart city and the Internet of Things, all vehicles are connected to each other, which improves the safety and efficiency of road transportation. In an Internet of Vehicles (IoV) environment, all vehicles transmit huge amounts of data in real time. I am very interested in the research fields of IoV and Big Data Analytics, which are still vibrant and rapidly developing. This paper presents a vehicle-tracking system for the Internet of Vehicles based on stream processing which allows for processing of huge amounts of IoV data streams for vehicle tracking in real-time. The system also deals with the tracking of vehicles which are not in the IoV system. Tracking is done in near real time, the aim being to reduce tracking latencies of vehicles. The proposed system is simulated and evaluated, and its performance is shown that the vehicles could be tracked even in a large scale of IoV deployment within milliseconds.

IEER
Institute for Excellence in Education & Research

I. INTRODUCTION

The convergence of Vehicular Ad-hoc Networks (VANETs) with the Internet of Things (IoT), now known as the Internet of Vehicles (IoV), has been receiving much attention from the scientific community over the last decade, turning into a substantial research area. In addition to being a novel frontier for the automotive sector, IoV is also an essential part of the new paradigm of the smart city [1]. Unlike previous vehicular networks that only enabled vehicle-to-vehicle (V2V) communications, IoV enables multi-modal communications between vehicles and infrastructure such as Road Side Units (RSUs), vehicles and humans such as pedestrians, and a variety of sensor-to-sensor and vehicle-to-vehicle communications. This ecosystem generates so much data that it is definitely considered a Big Data [2].

Vehicles today are packed with an increasing number of sensors and Internet-connected devices that can both create and use information all day, every day for the benefit of drivers and passengers. With the increased connectivity, another set of specifications have come up for the different types of communication channels: vehicle-to-infrastructure, vehicle-to-vehicle, vehicle-to-internet and intra-vehicle. These interactions

require secure, reliable and scalable information exchange of these interactions. However, IoV enables every vehicle to generate up to 30 GB of data every day [3] which presents a whole new set of data management challenges compared to earlier VANETs where data volumes were limited to localized safety applications like collision warnings.

The opportunities that IoV opens up are huge. Many applications have improved the driving experience, businesses and society today, such as parking space detection [5], real-time traffic monitoring [6] and location-based services [7, 8]. In 2020, 20% of all vehicles on the road had internet connectivity and the global volume of vehicular data traffic is more than 300,000 exabytes [9]. This stream of data is huge in size and comes in very high speed, and processing such a huge amount of data in meaningful time requires such kind of real-time stream processing infrastructure as explored in this paper.

An aspect that is not sufficiently addressed in the literature is the presence of connected and non-connected vehicles on the same roads. There are some cars that don't have the hardware required for participation in IoV, and others that choose not to participate for privacy reasons. However, in a networked world, the non-participating vehicles (NPVs) are actual issues of security and safety as they are not within the normal tracking coverage [10][11][12]. Roads shared between participating vehicles (PVs) and NPVs require a single tracking scheme that can be used by both types of vehicles.

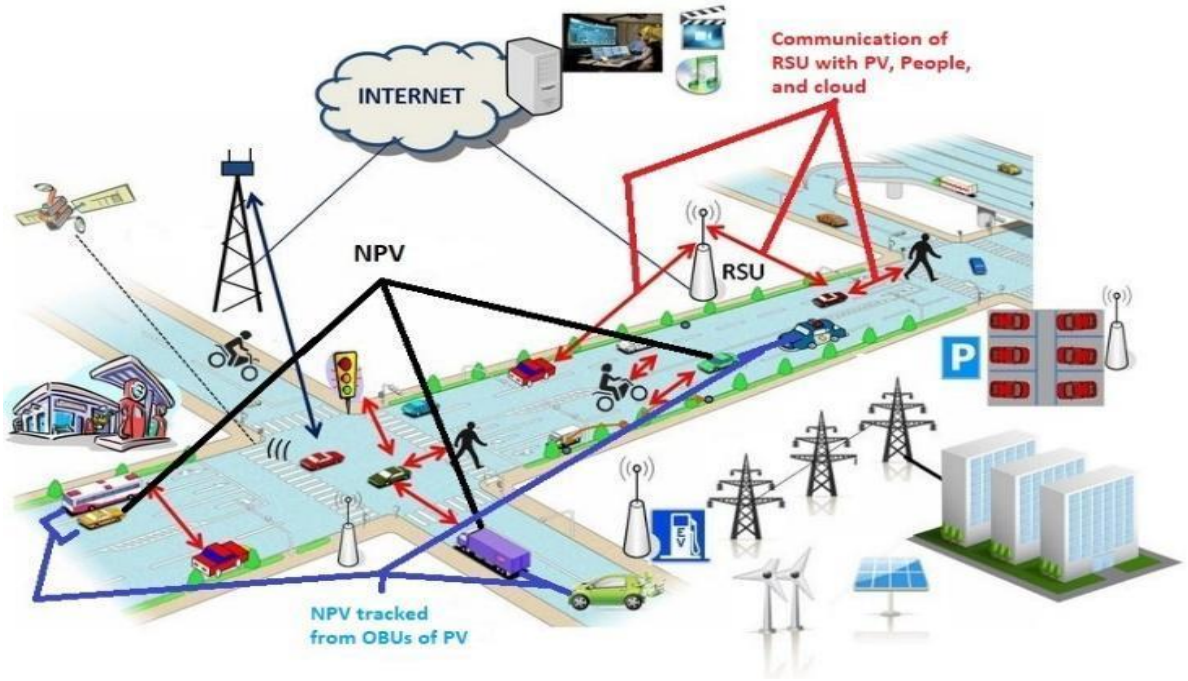

Fig. 1. IoV Environment illustrating the coexistence of participating and non-participating vehicles, RSU infrastructure, and internet connectivity.

Figure 1 shows the context in which this research is situated—the IoV environment. The system presented in this paper resolves the basic challenge of dealing with these huge amounts of real-time generated data, while also maintaining PV and NPV information. It is appropriate for stream processing platforms since these platforms are optimised to deliver high throughput and low latency. This research aims to provide this kind of platform to provide near real time vehicle tracking with guarantees on data access time for both vehicle categories under a common architecture.

The rest of this paper is organized as follows. The literature review, which is described in Section II, determines the gap in the research. In Section III, the presented proposed framework and methodology are presented, which consist of the system design and the major components of the system. Section IV describes the implementation and reports the experimental results. The findings are addressed in Section V in terms of previous research and their implications. The paper is concluded in Section VI and directions for future research are provided.

II. Literature Review

2.1 Vision-Based and Sensor-Fusion Vehicle Detection

Traditional vehicle tracking systems have been based on GPS data, with the difficulty of getting a clear signal in urban areas affecting the accuracy of the tracking. Using information from several sensors covers up the weaknesses of the individual sensors, and a more reliable localization result can be obtained [13]. A early and influential example of this is a dash-cam based detection system, based on an early and effective detector called WaldBoost [14] and a tracker called Tracking Learning Detection (TLD) [15] and achieved real-time detection of vehicles [16]. One method was based on background subtraction which identified abnormal driving behaviour in video streams and also estimated traffic flow and vehicle count [17]. Lately, deep learning has improved detection accuracy even further: deepened YOLO v5 architecture reduced false detection rates when camera images were occluded [43], YOLOv11 claimed good performance in various traffic situations [44] and large-scale deep learning pipeline using computer vision enabled real-time classification of traffic incidents [45].

2.2 Stream Processing Architectures for IoV

Lambda architecture [18] introduced a modern paradigm for big data processing, which was based on the separation of batch and real-time computation layers. Its implementation in a vehicular environment enabled the introduction of time-critical scheduling and deadline-based processing guarantees, which did not fully meet real-time requirements in dynamic tracking scenarios [19]. Subsequently, Basanta-Val et al. [27] showed that principled scheduling strategies can make Apache Storm, one of the most popular stream processing frameworks, more predictable. Zhou et al. [37] used Storm specifically for the access of spatio-temporal vehicle data in an urban sensor network, and found it to be suitable for continuous, high-rate vehicular data streams. One of the concerns in the distributed RSU architecture proposed in this work is network traffic in MapReduce pipelines and Hammoud et al. [30] proposed centre-of-gravity task scheduling to minimize network traffic.

2.3 IoV-Specific Tracking Approaches

To address this, Hou et al. [20] introduced vehicular fog computing that stores computing resources onto vehicles themselves, decreasing the reliance on the central cloud resources. In the harder problem of tracking groups of vehicles whose geometric size and dynamics are unknown, Wang and Ayalew [21] present a probabilistic grouping framework. JUNIPER project [26] explored performance of big data systems in time constrained environments and bottlenecks. The closest precedent to the system proposed here was the Integrated Vehicle Tracking System (IVATS) [24] that distributed video frame data from CCTV and dashboard cameras to processing nodes using Apache Kafka and HBase, and extracted licence plate, position and timestamp attributes from the frames. Plangi et al. [25] developed a real-time localization system based on multi-sensor fusion, which was implemented in smartphones. In recent years, several interesting systems have been proposed, such as the use of blockchain in intrusion detection for vehicle monitoring [46]; accident alert systems based on IoT [47]; self-adaptive traffic signal control based on license plate identification [48]; and integration of vehicle tracking and maintenance management systems [49].

2.4 Big Data Challenges in Streaming Vehicle Environments

The streaming data is unbound, time ordered and has variable rates. With the expansion of the

number of vehicles and sensors, it becomes more challenging to maintain the efficiency of processing [33]. The key to obtaining meaningful information from streaming data is to select the proper data based on the characteristics of the stream [34]. Large amounts of data cannot be effectively handled without sufficient CPU resources [35, 36]. This is complicated by the lack of adequate data cleansing pipelines in restrictive computing resources [37, 38]. To solve these problems, Apache Storm is a distributed solution, based on a parallel topology of processing components, designed for machine learning, data analysis, and continuous computation [39].

2.5 Identified Research Gaps

Searching through the cited material, there are some recurring gaps. First, there is no tracking system available that offers a way to track cars that are not part of the IoV network. Every road has an NPV and a PV; almost all published systems only deal with the connected subset. Second, systems that process vehicle data at scale are rarely able to return end-to-end latency with percentile breakdowns, which is the information needed to determine if a system can satisfy the needs of a particular application class. Third, edge deployment with RSUs where processing is done near the data source without sending raw data to the central cloud is seldom analysed in realistic load. Fourth, previous attempts to implement and test VAT-based visual profiling of NPVs from surrounding PV camera feeds have not been done in a stream processing framework. This paper seeks to tackle all four gaps.

III. Framework and Methodology

The research method is an experimental one that uses prototyping. It was designed, implemented and tested in a simulated vehicular environment in iterations until stable end-to-end operation was achieved. To design, four objectives were set: (1) direct data sharing to track participating vehicles, (2) cooperative observation of non-participating vehicles by surrounding PVs, (3) processing of vehicle data in real-time by a stream processing cluster on RSU nodes, and (4) low-latency query of entire vehicle journey histories. Each objective is described in the following sections, followed by the Evaluation Plan.

IEER
Institute for Excellence in Education & Research

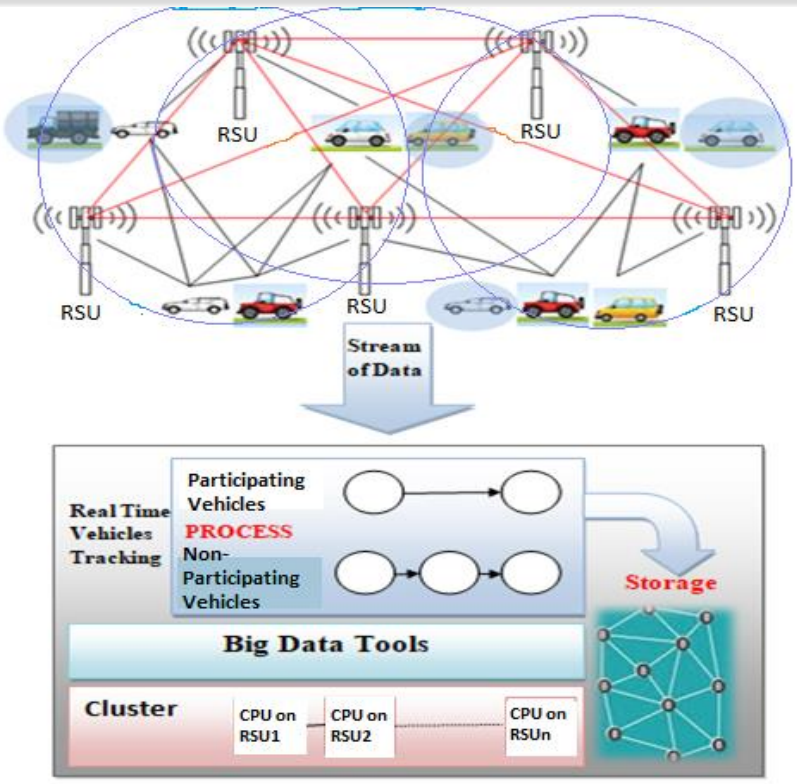

Fig. 2. Proposed architecture for real-time big data processing. PV data and NPV observations flow from RSUs through the stream processing cluster to in-memory storage.

The overall architecture is illustrated in figure 2. Vehicles traveling through RSUs' coverage zones will have data collected. PVs send out their own telemetry, while NPVs are observed passively via the cameras and sensors of nearby PVs. All of the data is pushed into the stream processing cluster and processed in real-time before results are written into RAM for quick query access.

3.1 Tracking of Participating Vehicles

Participating vehicles (PVs) on the IoV network directly feed the data including their location, speed and sensor readings to the closest RSU at fixed time intervals. They also describe the observation of nearby vehicles captured by the vehicles' onboard cameras, front and rear sensors, and LIDAR units. This cooperative sensing role is what renders PVs to be the most important information source for non-participating vehicles in the neighborhood. As each PV approaches the coverage area of an RSU, the RSU records its identifier, its time, and the RSU's physical location and adds this information to a linked list which is stored in RAM. The storage operation is performed in memory data structures and is done within the Storm bolt, which means that it is expected to be in the microsecond range, as demonstrated in the experimental results presented in Section IV.

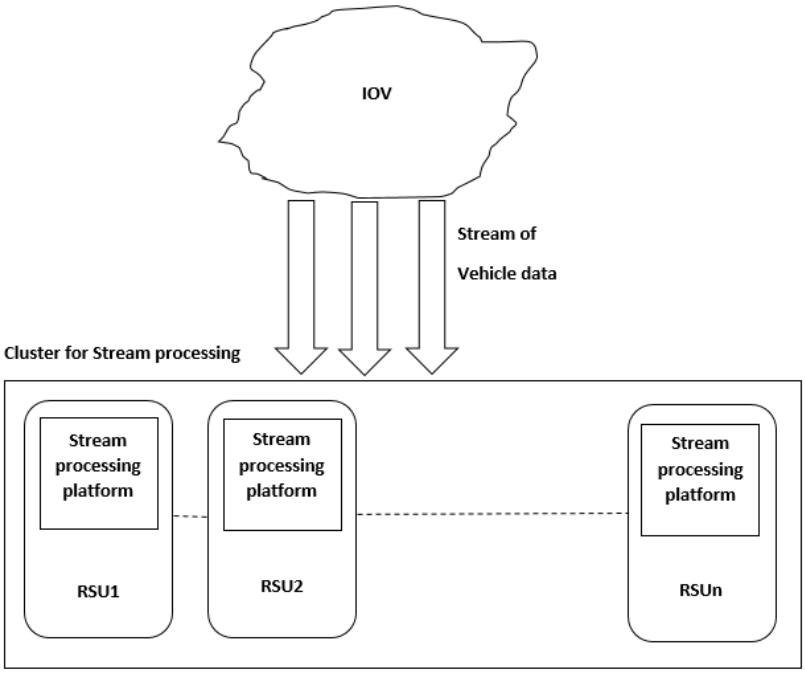

Fig. 3. Computational infrastructure of RSUs. Multiple RSU nodes form a mesh, each hosting a Kafka broker and Storm workers, eliminating single-node bottlenecks.

Figure 3 shows the RSU computational infrastructure. The mesh of n RSU nodes, with each node hosting a Kafka broker and a number of Apache Storm worker processes, removes the single point of failure that would occur if it was a single centre [40][41][42]. The cluster scales horizontally as the number of vehicles increases by the addition of RSU nodes.

3.2 Tracking of Non-Participating Vehicles

NPVs will not generate any traffic on the network themselves. Their presence is only known indirectly, by observation of neighbouring PVs. There are three categories of NPVs that are operated in distinctly different ways within the system:

3.2.1 Vehicles with valid registration plates but no connection (registered but not currently issued to this vehicle) (Vehicle registration plates that are not valid or cannot be matched against the vehicle registration database)

3.2.2 Unregistered vehicles only tracked by a VAT generated shadow profile and flagged as unregistered.

3.2.3 Vehicles in which a fraudulent or mismatched registration plate was detected, through a mismatch between the string captured by the camera and the string assigned to the vehicle by VAT, were flagged up for law enforcement.

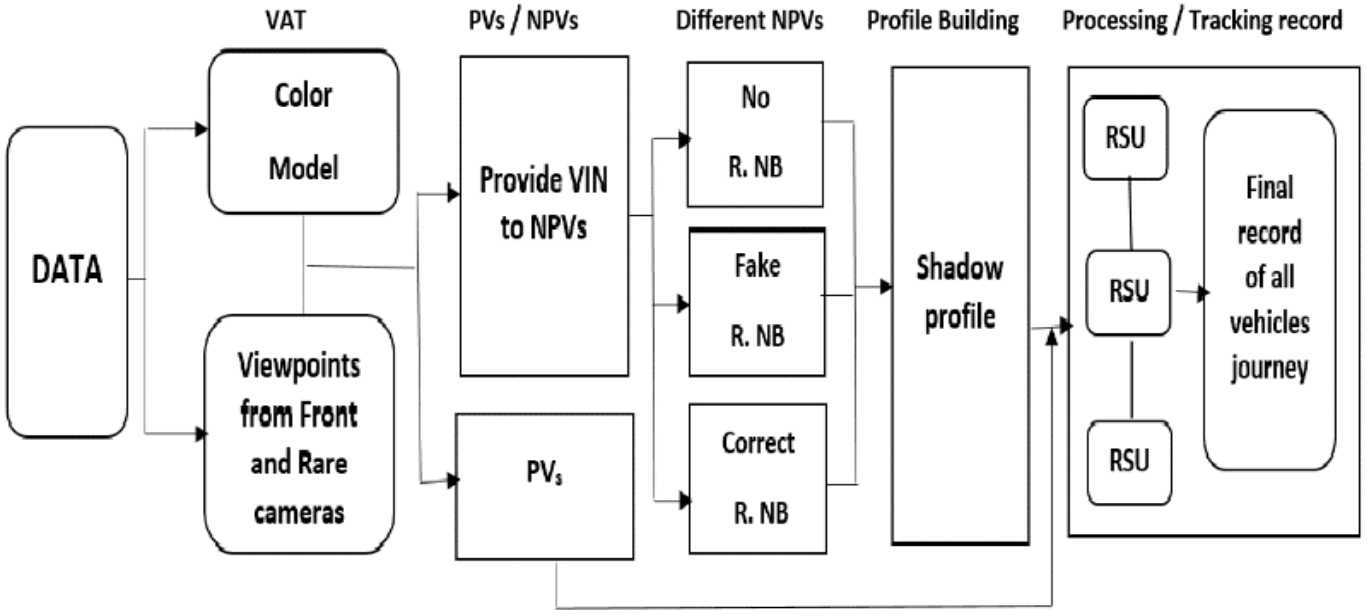

Fig. 4. Architecture for vehicle tracking. The system classifies each detected vehicle as PV or NPV and routes it through the appropriate processing path before storing the journey record.

The decision logic for when a vehicle enters the RSU coverage is shown in Figure 4. Identifiers of the PVs are verified and then put into the linked list quickly. Prior to creating or updating a shadow profile, NPVs are passed through the Vehicle Attribute Transformer. The output is that both categories of vehicles are maintained in the same pipeline and stored in parallel listsone for PVs, and one for NPVsthat get the whole picture of road traffic in the monitored area.

3.3 Vehicle Attribute Transformer (VAT)

The Vehicle Attribute Transformer is the part that transforms the multiple-angles of visual observations of an NPV into a searchable, stable identity. This works as follows. The camera images from the front and rear of neighbouring PVs are received and processed by two transformer modules, one of which extracts colour and vehicle class information, and the other extracts geometric viewpoint information from many angles. The transformer scores each observation by weighing it according to image quality and viewing angle. A Vehicle Identification Number (VIN) is created from the attribute vector's weighted attributes and assigned to an NPV.

Every NPV identity creates a shadow profile that is an in-memory record including the VIN, the attributes observed, and journey history. As more PV observations are received the shadow profile is progressively completed. If two profiles are found to be the same physical vehicle, they are combined into one profile, so that every vehicle that has been observed is mapped to a single shadow profile, even if multiple PVs have observed the same vehicle. This is an important step in ensuring proper journey history and no duplication of journeys.

3.4 Authentication of Vehicles

After a VIN is received or assigned, the system checks the identity of the vehicle in the national vehicle registration database. In the case of PVs, the self-reported registration number will be verified. In case of an NPV having a plate with a VIN, and this VIN does not correspond to the VIN assigned by VAT, the vehicle will be marked as a possible false-registration case. Registration is not required for NPVs that cannot be read on a

plate. Authorised personnel can see in real time suspicious or un-registered vehicles in the monitored area with the dedicated NPV list, which is displayed next to the standard PV list.

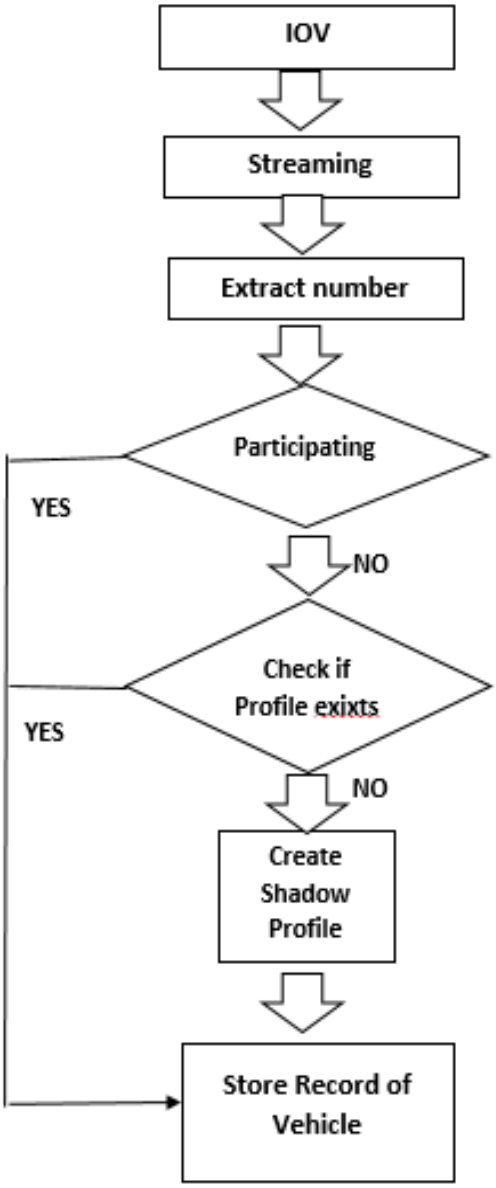

Fig. 5. Flow diagram of vehicle tracking and querying. PV and NPV paths converge at the in-memory store, with shadow profile creation handling new NPV identities.

The complete decision flow is presented in figure 5. Since the stream must enter from IoV, the system checks if the vehicle is a PV or a NPV, and the correct path is taken prior to recording the journey in RAM. If the number or VIN of any vehicle is given as a query, an actual travel history will be returned in real time.

3.5 Querying Vehicle Location

The query interface can be provided with a vehicle number (or VIN) and returns the complete journey history of that vehicle including all the RSUs it passed through, the time when it visited each one of them, and the last recorded location

of the vehicle. The lookup is O(1) amortised - there's no disk access or database seeking. If the vehicle that is being queried doesn't exist in the system, a 'no record found' response is returned immediately not a timeout. This design makes the response time to the query mainly dependent on network round-trip latency and Storm bolt execution time, and not on storage access time, which explains the sub-10 second worst case reported in the results.

3.6 Evaluation Plan

The system was tested in two performance aspects. The first was the time spent in the vehicles table while writing to the vehicle location table, to evaluate the efficiency of the write path: how fast the Storm bolt can add a vehicle record to the in-memory linked list. Times were recorded by using System.The operation occurred at the same time, before and after it, as indicated by nanoTime() calls. Second, for every query submitted to the Kafka topic, the end-to-end query response time was measured from the time the query was submitted until the time it was received by the query client. All measurements were taken after the warmup period to ensure stabilisation of the JIT compilation. The simulation was performed with a large number of vehicles, 50,000+ cars, yielding a statistically significant data set and large data set for both metrics.

IV. Implementation

The proposed system is accomplished using Apache Kafka, Apache Storm, and the ST-SIM vehicular network simulator. A Kafka producer continuously sends the vehicle events to a Kafka topic generated by the simulator. The topic is consumed by a KafkaSpout, and the tuples are passed to a sequence of Storm bolts, where each bolt contains the logic of a classification of the vehicle, processing the VAT and saving the tuple in a linked-list. What you have is a full real-time pipeline where any vehicle number can be queried and you can find its entire travelling history.

The experimental cluster is made up of four RSU compute nodes with each node running a Kafka broker and an Apache Storm supervisor process. The Kafka cluster is composed of 4 brokers, 8 partitions per topic and 2 replication factors to ensure fault tolerance. The topology consisted of a KafkaSpout that was connected to the processing bolts that realized the entire pipeline defined in Section III. ST-SIM was set up to produce a mixed fleet composed of 50,000+ vehicles, including PVs and NPVs, and with realistic urban mobility patterns.

Apache Storm executes on the RSU nodes, and as vehicles arrive in an RSU coverage area, messages are sent to the RSU, which appends a tracking record (with a timestamp) to the linked list. PV and NPV records are stored in different ArrayLists. The current vehicle location, identifier and timestamp are persisted in the Kafka topic. A consumer fetches a topic and fills in the tracking lists into which queries are served. The query function receives a vehicle number and returns the full history of the location in nanoseconds, searching the linked list.

4.1 Vehicle Location Storage Time

The write latency of all vehicles in the 50,000 vehicle simulation is shown in Figure 6. The vehicle index is plotted on the x-axis while the time to insert the location record into the in-memory linked list is plotted on the y-axis, also in nanoseconds. The distribution shape for writes less than 300 microseconds is displayed in the right panel, comprising more than 96% of all writes.

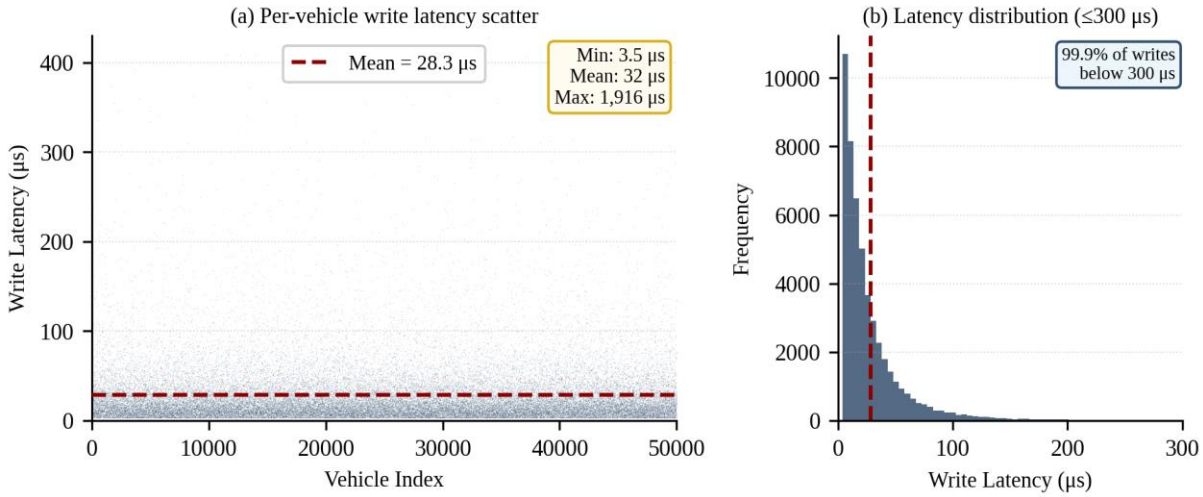

Fig. 6. Vehicle location storage time across the 50,000-vehicle simulation. Left: scatter across vehicle index. Right: distribution histogram for writes ≤300 µs. Mean write time is 32 µs.

Three statistics characterise the write latency distribution:

- Minimum write time: 3.5 microseconds.
- Mean write time: 32 microseconds.
- Maximum write time: 1,916 microseconds (approximately 1.9 milliseconds).

The in-memory linked list approach is clearly effective for large-scale vehicles monitoring tasks, as evidenced by the low average runtime of 32 microseconds. There are occasional surges towards the max, which are due to pauses in the JVM in which garbage is collected and network jitter between Storm worker nodes during transient times. Both are rare and fixed: They are limited to a small minority of insertions, and they do not increase over time; and as can be seen in the absence of any trends over time in the vehicle index, they are not common.

4.2 Query Response Time

Empirical cumulative distribution function (CDF) of end-to-end query response time for all the queries submitted during the simulation is presented in Figure 7. The S-shaped curve, which is typical of a log-normally distributed response time distribution, climbs rapidly then levels off as it gets closer to the maximum observed value in the low-latency range.

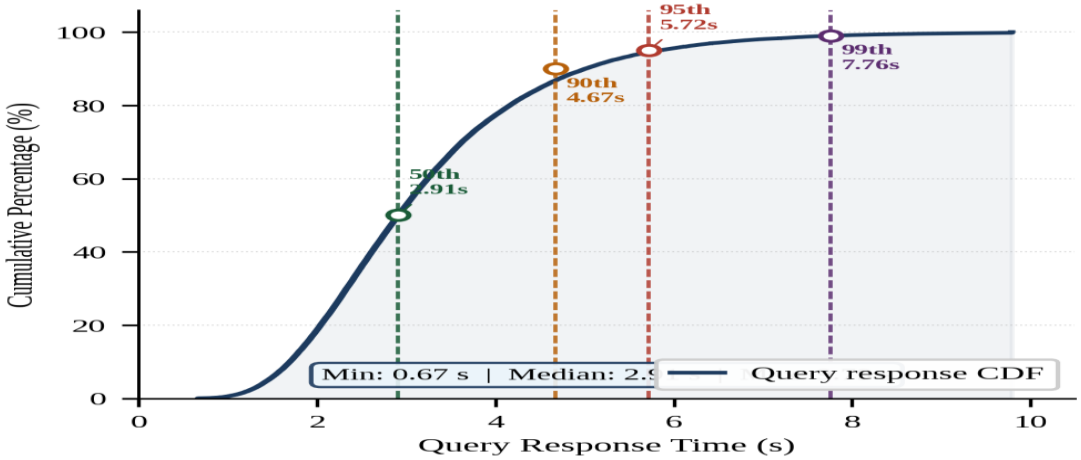

Fig. 7. Cumulative distribution of query response times. Vertical markers indicate the 50th, 90th, 95th, and 99th percentile values.

Table I summarises the key percentile statistics for query response time.

Table I. Query Response Time Statistics (50,000 Vehicles)

Min (ms)	Max (ms)	Median (ms)	90th %ile (ms)	95th %ile (ms)	99th %ile (ms)
670.31	9792.16	2905.38	4671.76	5716.69	7760.00

The results of the latency analysis validate the proposed Kafka–Storm based framework for the vehicle tracking application for large scale, with bounded query response times. The minimum, median and maximum latencies were 0.67 s, 2.91 s and 9.79 s respectively. Additionally, 95% of the complete vehicle query was done within 5.7 s and 99% completed within 7.76 s. The results show that the framework can be used for near real time vehicle monitoring, vehicle history retrieval, smart city analytics and law enforcement applications. The measured response times, however, are not in compliance with the strict latency requirements of safety-critical applications like collision avoidance and autonomous driving, where a latency under 100ms has to be guaranteed. The system proposed is therefore more of a large scale near real time vehicle tracking system than a hard real time safety system.

Under large-scale workloads, the framework guarantees predictable and bounded systems performance: 99% of vehicle queries will be processed within 7.76 seconds, and all observed queries will complete within 10 seconds. This predictability is as crucial as the average latency – operational users in law enforcement and traffic management require the worst case scenario response time and not just the average.

V. Discussion

5.1 Comparison with Related Work

Table II compares the proposed system with the four most closely related published systems with regard to 8 operationally relevant dimensions. Only systems for which functional prototypes and experimental data are reported are included in the selection.

IEER
Institute for Excellence in Education & Research

Table II. Feature Comparison with Related Tracking Systems

Feature	Proposed System	IVATS [24]	Plangi et al. [25]	Zhou et al. [37]
NPV Tracking	✓ Yes	✗ No	✗ No	✗ No
Stream Processing	Kafka + Storm	Kafka + HBase	Smartphone	Apache Storm
In-Memory Storage	✓ RAM	✗ Disk (HBase)	✗ No	Partial
VAT / Shadow Profile	✓ Yes	✗ No	✗ No	✗ No
RSU Edge Deployment	✓ Yes	✗ No	✗ No	Partial
50,000+ Vehicle Scale	✓ Yes	Multi-camera	Small scale	Urban sensors
Sub-ms Write Latency	✓ 32 µs mean			
99th %ile Query (s)	7.76			

IVATS [24] is the most architecturally similar system that has been developed in the past, and is similar in its use of Apache Kafka for ingestion and HBase for storage, but was used to extract vehicle attributes from camera feeds. The important difference lies in how it will be stored. HBase is a distributed column store with disk persistence and fault-tolerant, but with disk I/O

overhead for both writes and reads. The system will replace disk storage with linked lists that are stored in RAM, thus decreasing write latency from the millisecond region to the 10 microsecond range. The downside is that in-memory state is not inherently durable: if a node crashes, the buffered records are lost until Storm replays from the last Kafka offset. Periodic checkpointing to a persistent store could fill the gap for those law-enforcement applications that require full audit trails.

The biggest difference between all the systems compared is the support of NPVs. There are no current systems that offer a means of road tracking of untethered vehicles. The VAT and shadow-profile architecture proposed in this paper is unique to the proposed framework, and fills the gap that has been identified by literature review as the most consistently missing element in IoV tracking research. Partial solutions have been proposed by Plangi et al. [25] and Zhou et al. [37] in covering or scaling but both do not work with non-connected vehicles. It is noted that the sub-32 µs mean write latency and the bounded query time of 7.76 s 99th percentile are not reported by any comparable system and cannot therefore be compared numerically without access to the original experimental environments.

5.2 Significance of the Results

Most practically relevant result is the confirmation that a unified PV and NPV tracking system can be in place and run at almost real-time speed, across a fleet of 50,000+ vehicles on commodity RSU hardware. With a mean write latency of 32 microseconds, RAM-resident stream processing isn't just a nice idea, it's a performance that meets the requirements for continuous, high-frequency ingestion of vehicle data, without needing to rely on disk. More than 96% of the writes are completed within 300 microseconds, allowing the system to handle high volumes of traffic from urban areas.

The query response time distribution is another story altogether, but also important. The system is well in the near real-time tier with a median of 2.91 seconds and a 99th percentile of 7.76 seconds. In order to be operationally acceptable for the target applications such as law enforcement queries, smart city dashboards, and retrospective incident investigations, a response must be delivered within a few seconds. Users' concerns in these environments are not so much with the absolute speed of the system, but with its predictability, that it will always respond within a known upper bound. The 10 second observed ceiling actually offers just that.

It is also confirmed that the gap in NPV tracking which was found in the literature review can be filled without compromising system performance. The amount of time it takes to process a query on the VAT processing path for the non-participating vehicles did not perceptibly change the overall query latency for the entire fleet; the VAT processing path is executed in parallel with the Storm bolt operation, and the result of the VAT operation is stored in the same in-memory structure as the PV records. This can therefore be achieved with a latency cost that can be operated without any trouble by law-enforcement and traffic-management authorities and comprises the entire road population, not only the subset connected to the service.

VI. Conclusion and Future Work

Most practically relevant result is the confirmation that a unified PV and NPV tracking system can be in place and run at almost real-time speed, across a fleet of 50,000+ vehicles on commodity RSU hardware. With a mean write latency of 32 microseconds, RAM-resident stream processing isn't just a nice idea, it's a performance that meets the requirements for continuous, high-frequency ingestion of vehicle data, without needing to rely on disk. More than 96% of the writes are completed within 300 microseconds, allowing the system to handle high volumes of traffic from urban areas.

The query response time distribution is another story altogether, but also important. The system is well in the near real-time tier with a median of 2.91 seconds and a 99th percentile of 7.76 seconds. In order to be operationally acceptable for the target applications such as law enforcement queries, smart city dashboards, and retrospective incident investigations, a response must be

delivered within a few seconds. Users' concerns in these environments are not so much with the absolute speed of the system, but with its predictability, that it will always respond within a known upper bound. The 10 second observed ceiling actually offers just that.

It is also confirmed that the gap in NPV tracking which was found in the literature review can be filled without compromising system performance. The amount of time it takes to process a query on the VAT processing path for the non-participating vehicles did not perceptibly change the overall query latency for the entire fleet; the VAT processing path is executed in parallel with the Storm bolt operation, and the result of the VAT operation is stored in the same in-memory structure as the PV records. This can therefore be achieved with a latency cost that can be operated without any trouble by law-enforcement and traffic-management authorities and comprises the entire road population, not only the subset connected to the service.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, et al., "Internet of vehicles in big data era," IEEE/CAA Journal of Automatica Sinica, vol. 5, pp. 19–35, 2017.

I. A. Abbasi and A. S. Khan, "A review of vehicle to vehicle communication protocols for VANETs in the urban environment," Future Internet, vol. 10, p. 14, 2018.

G. P. Joshi, E. Perumal, K. Shankar, U. Tariq, T. Ahmad, and A. Ibrahim, "Toward blockchain-enabled privacy-preserving data transmission in cluster-based vehicular networks," Electronics, vol. 9, p. 1358, 2020.

M. A. Rahim, M. A. Rahman, M. M. Rahman, A. T. Asyhari, M. Z. A. Bhuiyan, and D. Ramasamy, "Evolution of IoT-enabled connectivity and applications in automotive industry: A review," Vehicular Communications, vol. 27, p. 100285, 2021.

J. Huang, Y. Qian, and R. Q. Hu, "A privacy-preserving scheme for location-based services in the Internet of Vehicles," Journal of Communications and Information Networks, vol. 6, pp. 385–395, 2021.

L. Li, Y. Lin, B. Du, F. Yang, and B. Ran, "Real-time traffic incident detection based on a hybrid deep learning model," Transportmetrica A: Transport Science, vol. 18, pp. 78–98, 2022.

H. Huang, "Location based services," in Springer Handbook of Geographic Information, Springer, 2022, pp. 629–637.

Z. Gui, Y. Sun, L. Yang, D. Peng, F. Li, H. Wu, et al., "LSI-LSTM: An attention-aware LSTM for real-time driving destination prediction," Neurocomputing, vol. 440, pp. 72–88, 2021.

J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in internet of vehicles: A review," IEEE Transactions on Intelligent Transportation Systems, vol. 16, pp. 2339–2352, 2015.

C. Aydin, C. Tarhan, and V. Tecim, "IT based vehicle tracking system for effective management in public organizations," Procedia Economics and Finance, vol. 33, pp. 506–517, 2015.

J. Wu, "An automatic procedure for vehicle tracking with a roadside LiDAR sensor," ITE Journal, vol. 88, pp. 32–37, 2018.

S. Kannimuthu, C. Somesh, P. Mahendhiran, D. Bhanu, and K. Bhuvaneshwari, "Certain investigation on significance of IoT and big data in vehicle tracking system," Indian Journal of Science and Technology, vol. 9, 2016.

IEER
Institute for Excellence in Education & Research

F. Hu and G. Wu, "Distributed error correction of EKF algorithm in multi-sensor fusion localization model," IEEE Access, vol. 8, pp. 93211–93218, 2020.

J. Sochman and J. Matas, "WaldBoostlearning for time constrained sequential detection," in Proc. IEEE CVPR, 2005, pp. 150–156.

Z. Kalal, J. Matas, and K. Mikolajczyk, "PN learning: Bootstrapping binary classifiers by structural constraints," in Proc. IEEE CVPR, 2010, pp. 49–56.

C. Caraffi, T. Vojíř, J. Trefný, J. Šochman, and J. Matas, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," in Proc. IEEE ITSC, 2012, pp. 975–982.

S. Hai-Feng, W. Hui, and W. Dan-Yang, "Vehicle abnormal behavior detection system based on video," in Proc. 5th Int. Symp. Computational Intelligence and Design, 2012, pp. 132–135.

J. Warren and N. Marz, Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Simon and Schuster, 2015.

P. Basanta-Val, N. Fernandez-Garcia, L. Sánchez-Fernández, and J. Arias-Fisteus, "Patterns for distributed real-time stream processing," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 3243–3257, 2017.

X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," IEEE Transactions on Vehicular Technology, vol. 65, pp. 3860–3873, 2016.

Q. Wang and B. Ayalew, "A probabilistic framework for tracking the formation and evolution of multi-vehicle groups," IEEE Transactions on Intelligent Transportation Systems, vol. 19, pp. 560–571, 2017.

K. Fan, W. Jiang, Q. Luo, H. Li, and Y. Yang, "Cloud-based RFID mutual authentication scheme for efficient privacy preserving in IoV," Journal of the Franklin Institute, 2019.

P. Liu, B. Qi, and S. Banerjee, "EdgeEye: An edge service framework for real-time intelligent video analytics," in Proc. 1st Int. Workshop Edge Systems, Analytics and Networking, 2018, pp. 1–6.

S. Jung, Y. Kim, and E. Hwang, "Real-time car tracking system based on surveillance videos," EURASIP Journal on Image and Video Processing, vol. 2018, pp. 1–13, 2018.

S. Plangi, A. Hadachi, A. Lind, and A. Bensrhair, "Real-time vehicles tracking based on mobile multi-sensor fusion," IEEE Sensors Journal, vol. 18, pp. 10077–10084, 2018.

N. C. Audsley, Y. Chan, I. Gray, and A. J. Wellings, "Real-time big data: The JUNIPER approach," 2014.

P. Basanta-Val, N. Fernández-García, A. J. Wellings, and N. C. Audsley, "Improving the predictability of distributed stream processors," Future Generation Computer Systems, vol. 52, pp. 22–36, 2015.

M. Cao, L. Zheng, W. Jia, and X. Liu, "Joint 3D reconstruction and object tracking for traffic video analysis under IoV environment," IEEE Transactions on Intelligent Transportation Systems, 2020.

M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive internet of vehicles," Computer Communications, vol. 120, pp. 58–70, 2018.

M. Hammoud, M. S. Rehman, and M. F. Sakr, "Center-of-gravity reduce task scheduling to lower MapReduce network traffic," in Proc. IEEE Cloud, 2012, pp. 49–58.

N. Ianuale, D. Schiavon, and E. Capobianco, "Smart cities, big data, and communities: Reasoning from the viewpoint of attractors," IEEE Access, vol. 4, pp. 41–47, 2015.

J.-Y. Park and R.-D. Oh, "Efficient sensor stream data processing system using cache for ubiquitous sensor network applications," Journal of Computer Science, vol. 8, p. 333, 2012.

C. Yang, X. Zhang, C. Zhong, C. Liu, J. Pei, K. Ramamohanarao, et al., "A spatiotemporal compression based approach for efficient big data processing on cloud," Journal of Computer and System Sciences, vol. 80, pp. 1563–1583, 2014.

C. S. Liew, T. Y. Wah, J. Shuja, and B. Daghighi, "Mining personal data using smartphones and wearable devices: A survey," Sensors, vol. 15, pp. 4430–4469, 2015.

H. Zhao, L. Yao, Z. Zeng, D. Li, J. Xie, W. Zhu, et al., "An edge streaming data processing framework for autonomous driving," Connection Science, vol. 33, pp. 173–200, 2021.

F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Enabling massive IoT toward 6G: A comprehensive survey," IEEE Internet of Things Journal, vol. 8, pp. 11891–11915, 2021.

L. Zhou, N. Chen, and Z. Chen, "Efficient streaming mass spatio-temporal vehicle data access in urban sensor networks based on Apache Storm," Sensors, vol. 17, p. 815, 2017.

D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, et al., "6G Internet of Things: A comprehensive survey," IEEE Internet of Things Journal, 2021.

P. Karunaratne, S. Karunasekera, and A. Harwood, "Distributed stream clustering using micro-clusters on Apache Storm," Journal of Parallel and Distributed Computing, vol. 108, pp. 74–84, 2017.

L. S. Indrusiak, "End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip," Journal of Systems Architecture, vol. 60, pp. 553–561, 2014.

S. Hesham, J. Rettkowski, D. Goehringer, and M. A. Abd El Ghany, "Survey on real-time networks-on-chip," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 1500–1517, 2016.

P. Basanta-Val and J. S. Anderson, "Using real-time Java in distributed systems: Problems and solutions," in Distributed, Embedded and Real-time Java Systems, Springer, 2012, pp. 23–44.

Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-time vehicle detection based on improved YOLO v5," Sustainability, vol. 14, no. 19, p. 12274, 2022.

M. A. R. Alif, "YOLOv11 for vehicle detection: Advancements, performance, and applications in intelligent transportation systems," arXiv:2410.22898, 2024.

M. I. Basheer Ahmed, R. Zaghdoud, M. S. Ahmed, R. Sendi, S. Alsharif, J. Alabdulkarim, and G. Krishnasamy, "A real-time computer vision based approach to detection and classification of traffic incidents," Big Data and Cognitive Computing, vol. 7, no. 1, p. 22, 2023.

M. S. Peelam, V. Chamola, and B. K. Chaurasia, "Blockchain-enabled intrusion detection systems for real-time vehicle monitoring," Vehicular Communications, p. 100961, 2025.

C. Nandhu, A. Gopi, B. Nithamdhar, C. Laxmiprasanna, D. Srihari, A. Jitendra, and M. Saravanan, "IoT-based vehicle tracking with accident alert system," IJRPETM, vol. 9, no. 2, pp. 486–494, 2026.

M. Ashkanani, A. AlAjmi, A. Alhayyan, Z. Esmael, M. AlBedaiwi, and M. Nadeem, "A self-adaptive traffic signal system integrating real-time vehicle detection and license plate recognition," Inventions, vol. 10, no. 1, p. 14, 2025.

B. Zahri, T. S. Putri, and D. S. Pamungkas, "Integration of vehicle tracking, control and maintenance," Advanced Mechanical and Mechatronic Systems, vol. 2, no. 1, pp. 50–62, 2026.