DISTRIBUTED CARLA ARCHITECTURE FOR MULTI-AGENT CAV SIMULATION: DESIGN AND PRELIMINARY MULTI-GPU EVALUATION

Vadym Shevtsov[1][0000-0002-5115-4398], Nadiia Babkova[1][0000-0002-2200-7794], Zoia Kochuieva[1][0000-0002-4300-3370], Dmytro Sivykh[1][0000-0002-8585-734X], Ihor Kasianenko[1][0000-0003-1375-3522], Mamontov Anatolii[1][0000-0002-5586-2113], Dănuț-Iulian Stanciu[2][0000-0002-2160-6955]*

[1]National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine,

[2]National Institute of Research and Development in Mechatronics and Measurement Technique, Bucharest, Romania

E-mail(s): vadym.shevtsov@khpi.edu.ua (corresponding author's e-mail), nadiia.babkova@khpi.edu.ua, zoia.kochuieva@khpi.edu.ua, dmytro.sivykh@khpi.edu.ua, danut.stanciu@incdmtm.ro

Abstract - Connected and automated vehicles are a key direction in the development of intelligent transportation systems because they combine autonomous environmental perception, control algorithms, and information exchange between vehicles and road infrastructure. Real-world validation of such systems is costly, difficult to reproduce, and constrained by safety risks; therefore, a substantial part of research is performed through computer simulation. However, high-fidelity simulators such as CARLA exhibit limited scalability in scenarios involving large numbers of traffic agents, sensors, and urban infrastructure objects. This study proposes a distributed architecture for multi-agent simulation of connected and automated vehicles that uses multiple CARLA instances and provides a preliminary performance evaluation on multiple GPUs. The architecture comprises five functional layers: system administration and access control, client access, load balancing and global coordination, client simulation, and distributed execution. Scalability is supported through spatial partitioning of the urban scene, buffer zones, authoritative object ownership, state synchronization, and vehicle migration between simulation instances. SUMO is used to generate and coordinate traffic flows, while a custom urban district is reconstructed from publicly available OpenStreetMap data and further processed in MATLAB RoadRunner, Blender, and CARLA. An experiment on distributing sensor rendering across graphics processors showed that increasing the number of GPUs from 1 to 3 raised performance from 4.40 to 9.70 FPS, reduced the average simulation-step time from 227.08 to 103.14 ms, and yielded an approximate speedup of 2.2. The relative FPS gain was 120.45%, while the average step time decreased by 54.58%. The decrease observed when a fourth GPU was added suggests increasing synchronization and data-transfer overhead, although these components were not measured separately. The results are therefore interpreted as a preliminary evaluation of the sensor-rendering subsystem rather than a complete benchmark of distributed multi-agent scalability.

Keywords: Connected and automated vehicles, CARLA, Multi-agent simulation, Distributed simulation, Intelligent transportation systems, Spatial partitioning, Agent migration, Multi-GPU, scalability

1. Introduction

Urbanization, increasing traffic intensity, and the growing complexity of urban transport infrastructure raise the requirements for safety, network capacity, and adaptability. According to the World Health Organization, road traffic crashes cause approximately 1.19 million deaths worldwide each year [1]. Traffic congestion, uneven flow distribution, and insufficient responsiveness to changing road conditions also lead to economic losses, increased energy consumption, and environmental impacts.

Intelligent transportation systems (ITS) address these challenges by integrating vehicles, roadside infrastructure, sensing devices, wireless communication, and software platforms for real-time data collection and analysis. Directive (EU) 2023/2661 expands the European framework for ITS deployment, including transport-data

dx.doi.org/10.17683/ijomam/issue25/v2.3

availability, digital services, and connected and automated mobility [2].

Automated vehicles perceive their surroundings using cameras, lidars, radars, global navigation satellite systems, and inertial sensors, and then perform localization, trajectory planning, and control. Connected vehicles supplement onboard perception through vehicle-to-vehicle, vehicle-to-infrastructure, and vehicle-to-everything communication. Their combination forms the concept of connected and automated vehicles (CAV), enabling cooperative perception, coordinated maneuver planning, and joint traffic management.

Real-world CAV testing requires repeated execution of normal, critical, and rare situations. Such experiments are expensive, difficult to reproduce under identical initial conditions, and potentially unsafe. Consequently, simulation has become a principal instrument for development, training, verification, and validation.

CARLA is an open urban-driving simulator designed for research and validation of automated-driving systems. It supports three-dimensional urban scenes, vehicle dynamics, weather conditions, and configurable sensor suites [3]. Its use of Unreal Engine provides the visual and physical fidelity required for research in perception, localization, planning, and control.

High fidelity entails substantial computational cost. As the number of vehicles, sensor streams, and urban objects increases, CPU, GPU, memory, and network loads grow accordingly. A single CARLA instance is constrained by the resources of one node and, therefore, becomes insufficient for large multi-agent scenarios.

SUMO is widely used to model large traffic flows through a microscopic representation with comparatively low computational requirements [4]. However, it does not provide photorealistic three-dimensional environments or realistic sensor simulation. A combined CARLA-SUMO workflow is therefore useful: SUMO coordinates the global road network and traffic flows, whereas CARLA provides physical and sensor-level simulation.

Integrated environments such as OpenCDA combine co-simulation, cooperative-driving automation stacks, and scenario management [5, 6]. Nevertheless, integration alone does not solve the scalability problem because larger scenarios increase the amount of synchronized state, inter-agent interactions, and global-time coordination.

The remaining research gap concerns the integration of high-fidelity simulation with distributed execution across multiple CARLA nodes. The key challenges are spatial partitioning, authoritative ownership, state synchronization, vehicle migration across simulation boundaries, and load-aware node selection.

This study proposes a scalable multi-agent CAV simulation architecture based on distributed CARLA instances and provides a preliminary performance evaluation of its multi-GPU sensor-rendering component. The contribution is therefore architectural and methodological; the reported speedup must not be interpreted as a complete end-to-end benchmark of distributed multi-agent execution.

The approach is also relevant to digital twins of urban transport infrastructure. In Ukraine, it could support analysis of traffic changes due to damaged or temporarily unavailable roads, alternative-route testing, evacuation planning, and transport-recovery projects [7, 8]. The experimental scene is described neutrally as a custom urban district reconstructed from publicly available OpenStreetMap data; the method is not tied to a specific geographical territory.

2. Related Work

2.1 High-Fidelity Simulation and CARLA Scalability

CARLA supports configurable urban scenes, vehicles, pedestrians, environmental conditions, and multiple sensor types, including RGB and depth cameras, semantic cameras, lidar, radar, GNSS, and inertial measurement units [3]. These capabilities make it suitable for closed-loop testing of perception and control, but rendering and sensor generation can become dominant components of the simulation step.

The official CARLA multi-GPU mode assigns sensor rendering to secondary servers while a primary server maintains the authoritative state [9]. This vertical acceleration reduces sensor-generation time but differs from horizontal distribution, which assigns world regions and agents to independent CARLA instances and therefore requires explicit ownership, boundary management, synchronization, and migration. The experiment evaluates the former, whereas the proposed architecture is designed for the latter.

2.2 Traffic and Communication Co-Simulation

SUMO represents traffic participants as microscopic agents and is efficient for large networks, but its simplified sensor representation limits perception-oriented experiments [4]. CARLA-SUMO co-simulation combines global traffic coordination with a high-fidelity subset of actors and sensors; it consequently requires consistent coordinate systems, identifiers, routes, control states, and time steps.

Network effects require an additional layer. OMNeT++ supports discrete-event communication and distributed simulation [10], while Artery-C provides the Cellular V2X protocol and application models [11]. These tools reproduce packet loss,

delay, bandwidth restrictions, and interference. The present architecture exposes logical V2X interfaces, but network-level behavior is planned for integration.

2.3 Integrated and Multi-Agent Environments

OpenCDA provides a modular framework for cooperative-driving automation, including co-simulation, perception, localization, planning, control, scenario management, and benchmarking [5, 6]. OpenCDA-ROS extends this concept to simulation-to-real deployment by linking cooperative driving workflows to the Robot Operating System [12]. These platforms emphasize cooperative algorithms and reusable research pipelines rather than spatial partitioning of a high-fidelity world.

SMARTS supports scalable multi-agent reinforcement-learning experiments and diverse interactive behaviors [13]. MetaDrive emphasizes procedurally generated and imported driving scenarios for generalizable learning [14], while Waymax provides accelerator-oriented, data-driven, closed-loop simulation for large-scale planning research [15]. Their scalability is achieved through representations and execution models different from CARLA's photorealistic rendering pipeline.

Distributed automated driving simulation has also been investigated using general multi-node execution platforms [16] and newer digital-twin-oriented multi-agent systems [17]. Across these approaches, a recurring trade-off is visible: systems that maximize physical and visual detail usually process fewer actors per node, whereas systems optimized for large populations often simplify sensing or vehicle dynamics.

Related research in distributed industrial automation has also demonstrated the use of cloud-based processing to reconstruct and represent the behavior of distributed cyber-physical systems. Wu et al. proposed a cloud-based data-driven framework in which distributed automation behavior is reconstructed from operational data and represented through interconnected function-block models [18]. Although their study addresses industrial automation rather than transportation, it illustrates the broader role of centralized computational services in coordinating and representing distributed cyber-physical components.

Spatial partitioning reduces per-node workload but is complicated by mobile actors crossing region boundaries. A vehicle must not be advanced by two authoritative nodes, and the target must have sufficient advance state to avoid discontinuities. The proposed design uses buffer regions, a prepared non-authoritative replica, and ownership transfer only after target confirmation.

Figure 1 summarizes the integrated ecosystem and coordination layer. Table 1 contrasts representative platforms and shows that explicit spatial ownership and migration remain distinct from conventional CARLA multi-GPU rendering.

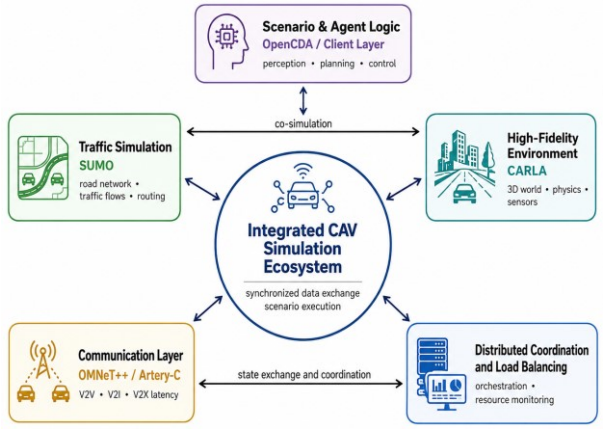

Figure 1: Integrated ecosystem for connected and automated vehicle simulation

Table 1. Comparative characteristics of representative CAV simulation platforms

Platform	Primary scope	Sensor fidelity	V2X support	Distribution / scalability mechanism
CARLA	High-fidelity 3D driving	High	External / logical modules	Primary-secondary multi-GPU rendering; normally node-limited
SUMO	Microscopic traffic flow	None	External modules	Large traffic networks; limited 3D sensing
OMNeT++ / Artery-C	Communication networks	None	Detailed V2X	Discrete-event and parallel network simulation
OpenCDA	Cooperative driving automation	Via CARLA	Integrated logical V2X	CARLA-SUMO co-simulation and reusable CDA modules
SMARTS	Interactive multi-agent learning	Simplified	Agent-level interfaces	Scalable behavior and reinforcement-learning execution
Proposed architecture	Distributed high-fidelity CAV simulation	High through CARLA	Logical interface; network model external	Spatial partitioning, authoritative ownership, migration, and load-aware node selection

3. Materials and Methods

3.1 General Methodology and Scope

The methodology combines high-fidelity physical and sensor simulation in CARLA, traffic coordination in SUMO, and workload distribution across multiple nodes. Each CARLA instance is associated with a defined spatial region. The architecture was designed to preserve a single simulation timeline and unambiguous object ownership while allowing neighboring nodes to prepare replicas before migration.

The evaluation has two distinct scopes. First, the architecture is analyzed at the design and implementation level, including node registration, load-aware selection, partitioning, synchronization, and migration. Second, a preliminary multi-GPU experiment quantifies sensor-rendering performance. Because end-to-end migration delay, network traffic, and resource utilization for the 25-, 50-, and 100-vehicle scenarios were not recorded, these scenarios are treated as functional stress designs rather than quantitative scalability benchmarks.

3.2 Five-Layer Architecture

System Administration and Access Control Layer. Registers computing nodes, monitors availability, manages configurations, and records infrastructure events.

Client Access Layer. Receives scenario parameters, actor and sensor configurations, control commands, and result requests without permanently binding a client to a specific CARLA server.

Load Balancing, Routing and Global Coordination Layer. Selects nodes, evaluates CPU/RAM/GPU utilization, communicates with SUMO, routes requests, and coordinates inter-zone transitions.

Client Simulation Layer. Contains perception, localization, planning, control, and logical V2X functions. A full discrete-event network model was not included in the reported experiment.

Distributed Execution Layer. Contains multiple CARLA instances responsible for physics, rendering, synchronization, replica management, and vehicle migration.

The five-layer organization is illustrated in Figure 2. The layered separation limits direct dependencies between clients and execution nodes, while the coordination layer maintains the global view required for load-aware routing and migration.

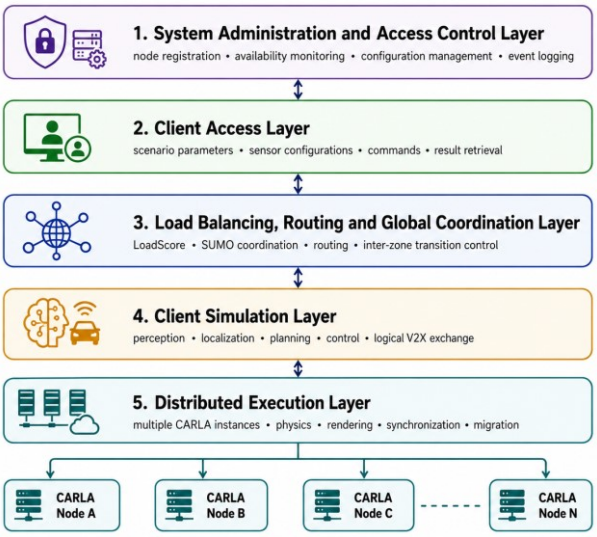

Figure 2: Five-layer distributed simulation architecture

3.3 Spatial Partitioning, Buffer Zones, and Ownership

The simulated territory is divided into zones, each served by a dedicated CARLA instance. The baseline implementation uses static partitioning, with boundaries defined before the experiment. Static decomposition simplifies neighbor discovery and transfer rules, but can lead to load imbalance when vehicles cluster near a complex intersection or within a single region.

Buffer regions are created between neighboring zones. A vehicle may be known to both nodes inside the overlap, but only one node remains authoritative. The authoritative node advances physics, applies control commands, updates coordinates, and publishes the primary sensor state. The neighboring node may maintain a non-authoritative replica but must not independently advance the same object.

The partitioning and ownership model is shown in Figure 3. The design uses pre-transfer replication rather than instantaneous object recreation at the geometric boundary, thereby reducing the risk of missing actors or discontinuous state after migration.

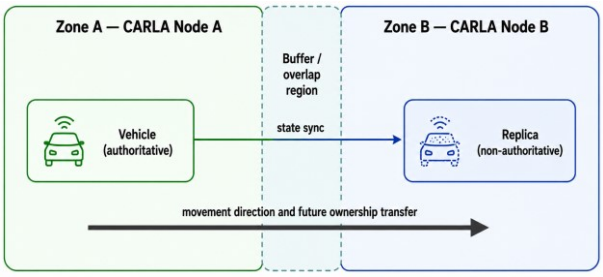

Figure 3: Spatial partitioning, buffer zones, and authoritative ownership

3.4 Synchronization and Vehicle Migration

Migration is triggered when a vehicle leaves its current zone and enters a neighboring region. The transferred state includes a global identifier, position, orientation, linear and angular velocities, acceleration, route, control parameters, vehicle type, sensor configuration, and agent-logic state.

The current node detects boundary approach and sends the state to the target node. The target creates a prepared non-authoritative representation in the overlap region. After the actor crosses the control boundary, the global coordinator switches ownership. The previous node removes its authoritative actor only after a successful confirmation from the target. All instances follow a common simulation clock so that the transferred state corresponds to the same logical step.

Figure 4 summarizes the migration workflow. Table 2 lists the minimum state groups that must be included in the transfer message.

The table describes the implemented conceptual data contract; it does not imply that every CARLA sensor payload is transmitted during ownership transfer.

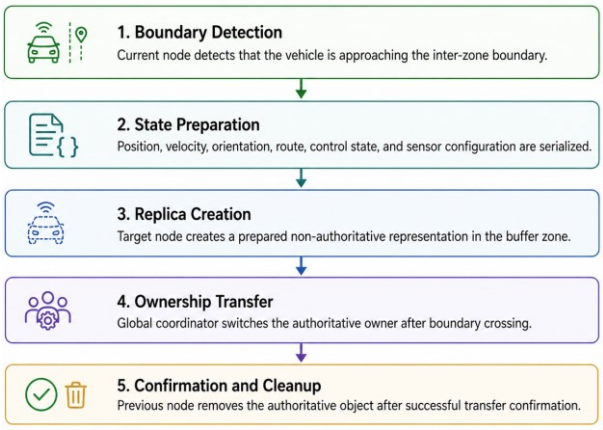

Figure 4: Vehicle synchronization and migration workflow

Table 2. Minimum state groups transferred during vehicle migration

State group	Representative fields	Purpose
Identity and ownership	Global vehicle ID, source zone, target zone, current owner	Prevents duplicate authoritative actors and supports routing
Kinematics	Position, orientation, linear/angular velocity, acceleration	Preserves motion continuity at the boundary
Control and route	Throttle, brake, steering, target route, lane context	Maintains behavior and planned movement
Actor configuration	Vehicle blueprint, dimensions, and active sensor configuration	Creates a compatible target representation
Synchronization metadata	Simulation step, timestamp, transfer status, acknowledgment	Coordinates creation, ownership switch, and cleanup

3.5 Load-Aware Node Selection

Node selection is implemented as a service-oriented procedure. A FastAPI-based coordination service maintains a server-state structure containing each node's IP address, client port, CARLA port, availability flag, CPU utilization, RAM utilization, and GPU utilization. A client request may include a preferred server, but the final assignment is made using the current state of the distributed infrastructure.

Before the integrated score is calculated, nodes that are unavailable or overloaded are removed from the candidate set. The configured exclusion thresholds are CPU utilization above 85%, RAM utilization above 90%, or GPU utilization above 95%. These thresholds prevent a node with one critically loaded resource from accepting new work merely because its other resources remain available.

For each remaining node i, CPU, RAM, and GPU utilization are expressed as normalized fractions in the interval [0, 1]. The integrated workload score is defined as follows:

$$\text{LoadScore}_i = 0.6 \cdot \text{CPU}_i + 0.2 \cdot \text{RAM}_i + 0.2 \cdot \text{GPU}_i. \quad (1)$$

The coefficients 0.6, 0.2, and 0.2 are empirically selected rather than statistically optimized. CPU receives the highest weight because it supports simulation logic, actor management, synchronization, and networking. RAM and GPU remain included because memory pressure and rendering saturation can destabilize sensor-intensive scenarios.

The selected server is the admissible node with the minimum workload score:

$$\text{Server}^* = \arg\min_i \text{LoadScore}_i. \quad (2)$$

If the preferred server remains admissible and its score is acceptable, the system may retain the user selection. Otherwise, the request is redirected to a

less-loaded node. The current implementation applies this mechanism to connection routing and monitoring; it does not claim to fully adaptively move entire spatial zones during execution.

3.6 Custom Urban Environment

The custom urban district was reconstructed from publicly available OpenStreetMap data [21]. Its original geographical name is omitted, and the scene is not presented as a Ukrainian map. The preparation method is geographically independent and can be repeated for another district after equivalent road and asset data are prepared.

The workflow includes road-network extraction, correction of geometry, lanes, and junctions in MATLAB RoadRunner [20], creation or adaptation of three-dimensional assets in Blender, and export to CARLA. OpenDRIVE provides the logical road-network representation [19]. Because the multi-GPU benchmark was intended to isolate rendering distribution, it used the standard Town10HD_Opt map rather than the custom scene.

The current publication does not report the custom map's total road length, number of junctions, or asset count because these characteristics were not retained in the archived experimental record. Consequently, the map is used to demonstrate the portability of the preparation workflow rather than to support a quantitative cross-map comparison.

3.7 Implementation Details

The coordinator maintains shared node availability and resource state, resolves client requests, and returns the selected CARLA connection parameters. Global vehicle identifiers are preserved across zones. During transfer, the target creates a replica and acknowledges readiness before the central ownership record is changed; only then may the source remove its authoritative actor. If readiness is not confirmed, the source remains authoritative. Timeout frequency and recovery delay were not measured and should be assessed through future fault-injection tests.

3.8 Experimental Configuration and Measurement Procedure

The experimental configuration is summarized in Table 3. Exact CPU and GPU model identifiers were not retained in the archived experiment record. This omission prevents the results from being interpreted as a hardware-independent benchmark and is explicitly addressed as a threat to reproducibility.

Table 3. Experimental software and scenario configuration

Component	Configuration
Operating system	Windows 11
Main simulator	CARLA 0.9.16
Traffic simulator	SUMO 1.27.1
Programming language	Python 3.12
Benchmark map	Town10HD_Opt
Execution mode	Synchronous
Vehicle and sensors	1 vehicle; 10 sensors
Warm-up / measured steps	40 / 150
GPU configurations	1, 2, 3, and 4 GPUs
Hardware identifiers	Model-specific CPU/GPU identifiers not retained

The primary server maintained the environment state and simulation logic without graphical rendering. Secondary servers were assigned to separate GPUs and generated sensor data. A separate CARLA configuration was started for each tested GPU count to prevent measurements from being mixed across configurations.

Forty warm-up steps were executed before data collection to reduce startup effects. The subsequent 150 synchronous steps were used to calculate average tick-plus-sensor time, standard deviation across measured steps, sensor waiting time, and FPS. Standard deviation, therefore, characterizes within-run step variability, not variability across independent runs.

The archived record specifies ten sensors but does not preserve their exact types, image resolutions, sensor ticks, or placement. These omitted settings can materially affect rendering costs; the reported results must therefore be interpreted as a preliminary comparison within a single fixed but incompletely documented sensor configuration.

The principal performance indicators are defined as:

$$FPS = 1000 / T(tick+sensor), \quad (3)$$

$$Speedup_n = T_1GPU / T_nGPU. \quad (4)$$

where Ttick+sensor is the mean duration of one synchronous simulation step, including the wait for required sensor data, measured in milliseconds; T1GPU is the baseline duration for one GPU; and TnGPU is the duration for n GPUs.

3.9 Functional Multi-Agent Scenario Design

Three traffic-density scenarios were defined to exercise progressively more demanding architectural conditions. They were not accompanied by complete performance logs and are therefore presented as scenario designs rather than measured scalability results. Table 4 identifies their intended stress factors.

Table 4. Functional stress factors in the multi-agent scenarios

Scenario	Vehicles	Primary stress factor	Mechanisms assessed
Low-density	25	Local interactions and isolated boundary crossings	Object assignment and basic synchronization
Medium-density	50	Higher state-update volume and uneven spatial density	Ownership and load monitoring
High-density	100	Concurrent boundary approaches and transfer requests	Migration coordination and transition control

4. Results

4.1 Multi-GPU Performance

The measured performance metrics are presented in Table 5 and visualized in Figure 5. These results quantify the sensor-rendering configuration; they do not directly measure migration or spatially distributed execution.

Table 5. Measured performance metrics for different GPU configurations

GPUs	FPS	Tick + sensor, ms	SD, ms	Sensor wait, ms	Speedup
1	4.40	227.08	12.07	223.56	1.00
2	8.15	122.63	8.72	119.46	1.85
3	9.70	103.14	11.01	99.88	2.20
4	8.65	115.55	15.71	111.94	1.97

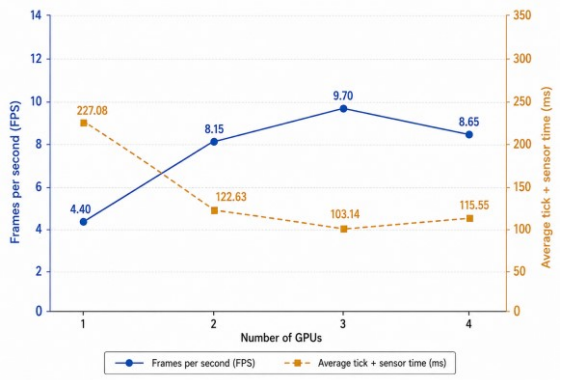

Figure 5: FPS and average tick-plus-sensor time versus the number of GPUs

Moving from one to two GPUs increased performance from 4.40 to 8.15 FPS, an improvement of approximately 85.23%. The average step time decreased from 227.08 to 122.63 ms, a 46.00% reduction, yielding a speedup of 1.85.

The best performance was achieved with three GPUs: 9.70 FPS, an average step time of 103.14 ms, and a speedup of 2.20. The relative FPS increase was 120.45%, while step time decreased by 54.58%. Sensor waiting time dropped from 223.56 to 99.88 ms.

Adding a fourth GPU did not provide further improvement. FPS declined to 8.65, average step time increased to 115.55 ms, and standard deviation rose to 15.71 ms. The reduction suggests that coordination and transfer overhead increasingly offset the benefit of another renderer, although the experiment did not isolate individual overhead components.

4.2 Parallel Efficiency and Execution Stability

Table 6 provides additional indicators derived from the measured values. Parallel efficiency remained high for two GPUs at 92.50%, decreased to 73.33% for three GPUs, and fell to 49.25% for four GPUs. The three-GPU configuration therefore maximized absolute FPS but did not maintain proportional scaling.

Table 6. Derived scalability and stability indicators

GPUs	Parallel efficiency, %	Coefficient of variation, %	Sensor-wait share, %	Non-sensor time, ms
1	100.00	5.32	98.45	3.52
2	92.50	7.11	97.41	3.17
3	73.33	10.67	96.84	3.26
4	49.25	13.60	96.88	3.61

The coefficient of variation increased from 5.32% for one GPU to 13.60% for four GPUs. This trend indicates greater step-time variability as additional secondary servers were introduced. It is consistent with a system in which the synchronous step must wait for the slowest participating rendering process.

Sensor waiting represented 96.84-98.45% of total step time in every configuration, while the residual non-sensor component remained between 3.17 and 3.61 ms. The experiment was consequently dominated by sensor generation rather than by the primary server's non-rendering simulation work. Together with the primary-secondary execution architecture, this observation suggests that further bottleneck analysis should primarily focus on the sensor-generation and synchronous-wait path This explains both the substantial gain from distributing sensors and the limited relevance of the result to workloads dominated by vehicle logic or network communication.

4.3 Functional Interpretation of the Multi-Agent Scenarios

The 25-vehicle scenario targets basic assignment, synchronization, and isolated boundary crossings; the 50-vehicle scenario increases state-update volume and spatial imbalance; and the 100-vehicle scenario stresses concurrent migration and global coordination.

These are structured functional test designs, not quantitative evidence of linear, sublinear, or real-time scalability, because resource, network-traffic, and migration-latency series were not retained.

5. Discussion

Performance did not increase monotonically with GPU count: three GPUs produced the best absolute result, whereas the fourth reduced FPS and parallel efficiency. The workload was rendering-dominated, and the synchronous step waited for all required sensor streams. Additional state distribution, scheduling, communication, or synchronization overhead may therefore have offset the marginal benefit of the fourth renderer. However, the experiment did not separately measure network latency, per-machine CPU utilization, synchronization or locking delays, or library-level execution time; therefore, the specific cause of the performance decrease cannot be identified from the available data.

The CARLA multi-GPU mechanism must be distinguished from the proposed distributed architecture. Multi-GPU accelerates sensor rendering; spatial distribution also divides vehicles, scene regions, physics, replica management, and migration. The 2.2-times speedup consequently applies only to the reported sensor-intensive benchmark.

SUMO provides efficient global traffic coordination, while CARLA supplies physical and sensor fidelity. OpenCDA, SMARTS, MetaDrive, and Waymax address cooperative automation or scalable learning through different representations. A direct comparison would require a common road network, behavior model, sensor load, hardware platform, and protocol.

Static partitioning simplifies ownership but can create an imbalance. The LoadScore heuristic mitigates server-selection imbalance, yet its weights are empirical, and zones are not dynamically reshaped. Future work should combine measured resource use with predicted vehicle density, sensor count, and migration rate.

For Ukraine, the architecture could support resilience-oriented digital twins for alternative routing, infrastructure closure, evacuation analysis, and reconstruction planning [7, 8]. These applications remain prospective because the benchmark used Town10HD_Opt rather than a Ukrainian map.

6. Threats to Validity

Internal validity is limited by the absence of independent repeated runs. Standard deviation describes 150 consecutive steps per run, per configuration, and cannot replace run-level confidence intervals.

Construct validity is limited because the experiment measures sensor rendering rather than complete distributed execution. Migration latency, ownership-transfer failures, network traffic, CPU/RAM utilization, and load-balancer response time were not recorded; therefore, the metrics do not support end-to-end scalability claims.

External validity and reproducibility are constrained by a single optimized map, a single vehicle, an incompletely documented ten-sensor configuration, and missing CPU/GPU identifiers. The custom map was not benchmarked. Future validation should use several maps and retain complete hardware, software, sensor, random-seed, and network manifests.

7. Conclusions

This study addressed the design of a distributed architecture for high-fidelity connected and automated vehicle simulation and provided a preliminary multi-GPU evaluation of a sensor-intensive CARLA workload. The architecture separates system administration, client access, global coordination, client simulation, and distributed execution. Its core mechanisms are spatial partitioning, buffer regions, a single

authoritative owner for every vehicle, state synchronization, and confirmed migration between neighboring CARLA instances.

Load-aware connection routing is based on a shared server-state registry and a weighted CPU/RAM/GPU score. Nodes exceeding CPU, RAM, or GPU thresholds are excluded before selection. The method supports practical infrastructure management but currently uses empirically chosen weights and does not dynamically repartition spatial zones.

The multi-GPU experiment showed that the three-GPU configuration provided the best absolute performance. FPS increased from 4.40 to 9.70, average tick-plus-sensor time decreased from 227.08 to 103.14 ms, and speedup reached approximately 2.20. The relative FPS gain was 120.45%, while step time decreased by 54.58%. Parallel efficiency declined from 92.50% for two GPUs to 73.33% for three GPUs and 49.25% for four GPUs. The coefficient of variation increased with the number of GPUs, indicating less stable synchronous step duration.

Sensor waiting accounted for more than 96% of step time across all configurations, while non-sensor time remained close to 3-4 ms. The benchmark therefore demonstrates the value of distributing sensor rendering, but it does not quantify end-to-end migration, communication, or multi-agent scalability. The scenarios with 25, 50, and 100 vehicles should be viewed as structured functional test designs pending complete resource and timing measurements.

The architecture can support future urban digital twins, including resilience analysis, alternative routing, temporary infrastructure closures, evacuation planning, and transport network recovery. Its transfer to Ukrainian cities requires preparation of corresponding OpenStreetMap, OpenDRIVE, RoadRunner, and three-dimensional scene data. Because the method is geographically independent, the same processing pipeline can be applied across districts, provided that road topology, junction logic, and relevant urban assets are validated prior to simulation.

Overall, the study establishes a technically explicit basis for separating sensor-rendering acceleration from horizontal simulation distribution. This distinction is essential when reporting scalability because gains obtained from additional rendering devices cannot be assumed to represent improvements in migration, coordination, or multi-agent throughput.

The main limitations are incomplete hardware and sensor records, a single optimized benchmark map, no independent repeated runs, no full network-level V2X model, no real-vehicle validation, and no quantitative migration measurements. These limitations limit hardware-independent comparison and preclude claims of real-time end-to-end scalability. Future work should introduce reproducible configuration manifests, multiple independent trials, direct logging of migration delay and failure rate, resource and network monitoring, dynamic partitioning, adaptive load-score calibration, V2X co-simulation, and validation on several custom urban maps.

References

[1] World Health Organization. Global status report on road safety 2023. Geneva: World Health Organization, 2023.

[2] European Parliament and Council of the European Union. Directive (EU) 2023/2661 of 22 November 2023 amending Directive 2010/40/EU on the framework for the deployment of Intelligent Transport Systems. Official Journal of the European Union, 2023.

[3] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. CARLA: An open urban driving simulator. Proceedings of the 1st Annual Conference on Robot Learning, PMLR, vol. 78, 2017, pp. 1-16.

[4] Alvarez Lopez, P., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y.-P., Hilbrich, R., Lucken, L., Rummel, J., Wagner, P., and Wiessner, E. Microscopic traffic simulation using SUMO. 2018 21st International Conference on Intelligent Transportation Systems, 2018, pp. 2575-2582. DOI: 10.1109/ITSC.2018.8569938.

[5] Xu, R., Guo, Y., Han, X., Xia, X., Xiang, H., and Ma, J. OpenCDA: An open cooperative driving automation framework integrated with co-simulation. 2021 IEEE International Intelligent Transportation Systems Conference, 2021, pp. 1155-1162. DOI: 10.1109/ITSC48978.2021.9564825.

[6] Xu, R., Xiang, H., Han, X., Xia, X., Meng, Z., Chen, C.-J., and Ma, J. The OpenCDA open-source ecosystem for cooperative driving automation research. IEEE Transactions on Intelligent Vehicles, vol. 8, no. 4, 2023, pp. 2698-2711. DOI: 10.1109/TIV.2023.3244948.

[7] Cabinet of Ministers of Ukraine. National Transport Strategy of Ukraine for the period up to 2030 and operational action plan for 2025-2027. Resolution No. 1550, 27 December 2024.

[8] Cabinet of Ministers of Ukraine. Certain issues of implementing the project Restoration of Critical Logistics Infrastructure and Network Connectivity (RELINC). Resolution No. 661, 4 June 2025.

[9] CARLA Simulator Team. Multi-GPU simulation. CARLA documentation. Available at: https://carla.readthedocs.io/en/latest/adv_multigpu/ (accessed July 2026).

[10] Varga, A., and Hornig, R. An Overview of the OMNeT++ Simulation Environment. Proceedings of the 1st International ICST Conference on

Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools), 2008. DOI: 10.4108/ICST.SIMUTOOLS2008.3027.

[11] Hegde, A., and Festag, A. Artery-C: An OMNeT++ based discrete event simulation framework for cellular V2X. Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2020. DOI: 10.1145/3416010.3423240.

[12] Zheng, Z., Xu, R., Xiang, H., Xia, X., Han, X., and Ma, J. OpenCDA-ROS: Enabling seamless integration of simulation and real-world cooperative driving automation. IEEE Transactions on Intelligent Vehicles, vol. 8, no. 7, 2023, pp. 3775-3780. DOI: 10.1109/TIV.2023.3298528.

[13] Zhou, M., Luo, J., Villella, J., et al. SMARTS: An open-source scalable multi-agent reinforcement learning training school for autonomous driving. Proceedings of the 2020 Conference on Robot Learning, PMLR, vol. 155, 2021, pp. 264-285.

[14] Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., and Zhou, B. MetaDrive: Composing diverse driving scenarios for generalizable reinforcement learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 3, 2023, pp. 3461-3475. DOI: 10.1109/TPAMI.2022.3190471.

[15] Gulino, C., Fu, J., Luo, W., et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. Advances in Neural Information Processing Systems, vol. 36, 2023, pp. 7730-7742.

[16] Tang, J., Liu, S., Wang, C., and Liu, C. Distributed Simulation Platform for Autonomous Driving. In: Internet of Vehicles. Technologies and Services for Smart Cities, Lecture Notes in Computer Science, 2017, pp. 190–200. DOI: 10.1007/978-3-319-72329-7_17.

[17] Huang, Z., Sheng, Z., Wan, Z., Qu, Y., Luo, Y., Wang, B., Li, P., Chen, Y.-J., Chen, J., Long, K., Meng, J., Leng, Y., and Chen, S. Sky-Drive: A Distributed Multi-Agent Simulation Platform for Socially-Aware and Human-AI Collaborative Future Transportation. Journal of Intelligent and Connected Vehicles, vol. 8, no. 4, 2025, article 9210070. DOI: 10.26599/JICV.2026.9210070.

[18] Wu, X., Yu, C., Zhang, L., Zhang, H., and Dai, W. Cloud-Based Data-Driven Behavior Model Recovery for Distributed Automation Systems. EAI Endorsed Transactions on Digital Transformation of Industrial Processes, vol. 1, no. 1, 2025. DOI: 10.4108/dtip.8591.

[19] ASAM e.V. ASAM OpenDRIVE Base Standard, version 1.8.1, 2024. Available at: https://www.asam.net/standards/detail/opendrive/ (accessed July 2026).

[20] MathWorks. RoadRunner documentation. Available at: https://www.mathworks.com/help/roadrunner/ (accessed July 2026).

[21] OpenStreetMap contributors. OpenStreetMap. Available at: https://www.openstreetmap.org/ (accessed July 2026).