High-concurrency Solution for Intelligent Connected Vehicle Race Based on Cloud Computing

Shiman Liu, Ziyi Liu, Shuai Zhao, Xin Hu*, Bolin Zhou, Chen Chen, Lingxiang Zhang, Xiaoting Li

Automotive Data of China (Tianjin) Co., Ltd., Tianjin, China

Abstract

Based on the existing high-concurrency solutions, this paper studies the high-concurrency solutions for intelligent connected vehicle race based on cloud computing, and verifies the effectiveness of the scheme by designing the scheduling method of the intelligent networked car race platform and high-concurrency tests. The experimental results show that the designed scheduling method of the intelligent networked car event platform can meet the high concurrency requirements of the event, especially when international communication is affected by the epidemic, ensuring the participation of domestic and foreign teams in the competition and effective technical exchanges. It has a relatively broad application prospect.

Keywords

Cloud computing, intelligent connected automobile events, high concurrency demand, cloud simulation platform

1. Introduction

As a new generation of information technology and transportation integrated development product, intelligent network vehicle is an important content of Chinese scientific and technological innovation support to accelerate the construction of transportation power. At present, virtual simulation is an indispensable link in automobile research and development[1], and its status is constantly improving in the industry background of accelerating the development of autonomous driving [2]. In addition, virtual simulation testing is characterized by high efficiency, strong test repeatability, safety, reliability and low cost, which is the direction and trend of future autonomous vehicle testing[3]. Therefore, based on cloud computing[4], the intelligent driving simulation competition is held, and the intelligent connected vehicle simulation development and verification are taken as the entry point, and the scientific research and research of intelligent connected vehicle are combined with the industrial competition to escort the technological development of intelligent connected vehicle virtual simulation.

Held at present, the related intelligent driving simulation cases are less, part of the event to take the form of the client game, is bad for the contestants from different regions are more, not all teams playing online at the same time, also can't see the real time teams scores ranking situation, there is no guarantee that the game fair, fair and open, Especially when international exchanges are affected by the epidemic, it is not conducive for international teams to participate in competitions and technical exchanges. In addition, they lack experience in dealing with a large number of participants accessing the interface of the competition platform and high concurrency requests. Once there is a high concurrent demand[5], the website may be paralyzed due to the heavy traffic[6], which further leads to the failure of the competition.

In the intelligent connected vehicle event based on cloud computing, due to the large number of participating teams and competition items, it is difficult to meet the high concurrency requirements of multiple teams in the same practice period, especially during the official competition, when all participating teams go online at the same time and run the competition questions at the same time. In addi-

ICCEIC2022@3rd International Conference on Computer Engineering and Intelligent Control
EMAIL: *Xin Hu@catarc.ac.cn (Xin Hu)

tion, in order to meet the fairness, fairness and openness of the competition, it is necessary to ensure that the competition environment of the participants is consistent.Therefore, it is necessary to invent a high concurrency solution of intelligent connected auto race based on cloud computing.This can ensure the success of the intelligent connected car event, promote the technical exchanges among the participating teams, and help the implementation of domestic intelligent connected technology.

Based on the existing high-concurrency solutions, this paper studies the high-concurrency solutions of intelligent connected auto races based on cloud computing. By designing the scheduling method of the intelligent connected vehicle event platform[7] and verifying the effectiveness and reliability of the scheme through highly concurrent testing, it provides a reference for the intelligent connected vehicle event based on cloud computing in the industry.

2. The experiment design

2.1. Design of scheduling method for intelligent Connected car race platform

In this experimental design, teams need to enter the corresponding contest interface by using their account and password on the competition login interface. After confirming the information on the question interface, manually start the question. In this case, the contest server automatically queries the current K8s cluster resource usage and determines whether there are idle resources available. If it determines that there are no idle resources available, the server directly enters the queuing mechanism. If it is determined that there are idle resources available, the server directly starts the virtual engine (UE4 is used in this experiment) and determines whether UE4 is successfully started. If the startup fails, the monitoring page is displayed. If the startup is successful, the server accounts for a period of time and checks whether an algorithm is connected within this period. If no algorithm is added, the server automatically stops the UE4 running process. If an algorithm is connected, the server will directly run the questions until the end of the questions and calculate the results.(see **Figure 1**).

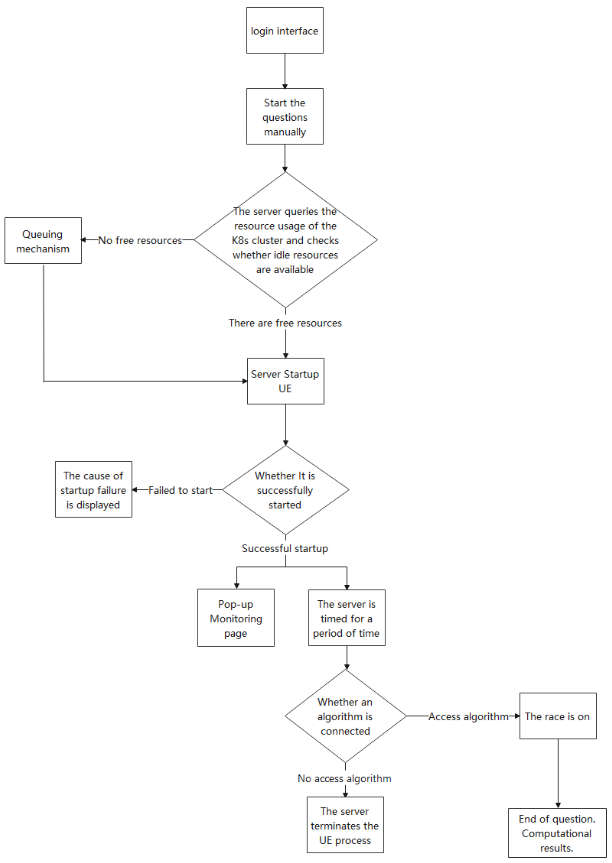

login interface
Start the questions manually
The server queries the resource usage of the K8s cluster and checks whether idle resources are available
No free resources
Queuing mechanism
There are free resources
Server Startup UE
Whether It is successfully started
Failed to start
The cause of startup failure is displayed
Successful startup
Pop-up Monitoring page
The server is timed for a period of time
Whether an algorithm is connected
Access algorithm
The race is on
No access algorithm
The server terminates the UE process
End of question. Computational results.

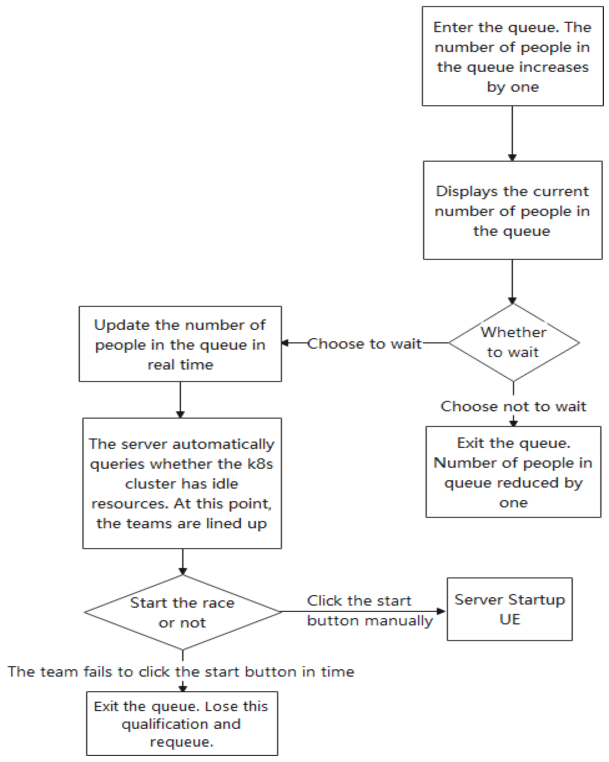
Figure 1 Design of scheduling method for intelligent Connected car race platform

2.1.1. Queuing mechanism design

In this experiment, the queuing mechanism is applied to the case that the server used in the contest determines that there is no idle resource available after automatically querying the current K8s cluster resource usage.

When the server automatically checks and determines that no idle resources are available, the current team will directly enter the queue, and the number of people in the queue will be increased by one. The current queue number is displayed. After that, it is up to the teams to choose whether to wait. If the team chooses not to wait, it will exit the queue directly, and the number of people in the queue will be increased or decreased by one. If you choose to continue waiting, the interface updates the number of people in line in real time.

When the server automatically queries that there are idle resources in the current K8s cluster resources, the team can choose whether to start the contest. If a team fails to manually click the start button in time within the time specified by the system, the team will directly exit the queue, lose the qualification and need to queue up again. If the team manually clicks the start button in time within the time specified by the system, the server will directly start UE4, and the current queue of the team will end **(Figure 2)**.



Enter the queue. The number of people in the queue increases by one
Displays the current number of people in the queue
Whether to wait
Choose to wait
Update the number of people in the queue in real time
Choose not to wait
Exit the queue. Number of people in queue reduced by one
The server automatically queries whether the k8s cluster has idle resources. At this point, the teams are lined up
Start the race or not
Click the start button manually
Server Startup UE
The team fails to click the start button in time
Exit the queue. Lose this qualification and requeue.

Figure 2 Queuing mechanism design

2.1.2 Design of K8s deployment scheme

This experiment adopts the container arrangement scheme of Kubernetes(K8s)[8]. The minimum number of program units can be infinitely extended by K8s technology. If the number of access reaches the load value, the hardware is configured as a new Node by kubectl (command), which can be put into use immediately after the startup, reducing the time of configuring the running environment(Figure 3).

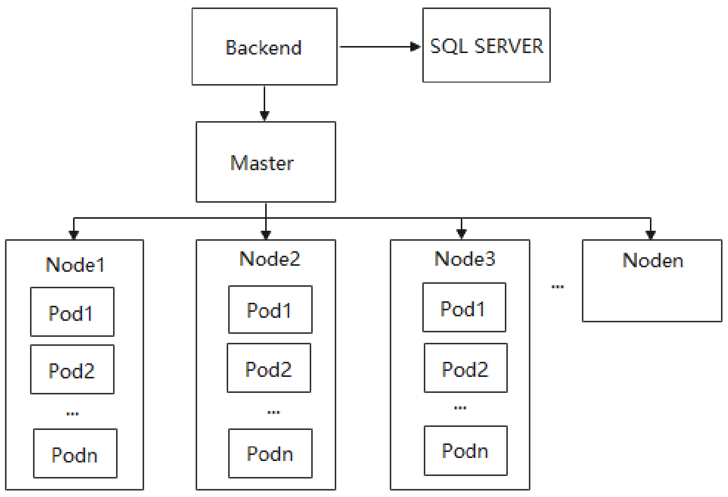


Figure 3 Design of K8s deployment scheme

In addition, K8s could manage multiple management nodes [9], reducing the access load of a single management node. It also greatly reduces the possibility of system outages. In addition, it facilitates node scaling and expansion management, achieves high concurrency, all containers, supports Web interface access, and automatically adjusts hardware resources based on requirements. Each task runs in Docker container, isolated from each other, and does not affect each other to maximize the use of physical hardware resources. At the same time, it ensures the convenience of deployment and expansion, and ensures the requirement of high concurrency for hundreds of people to go online simultaneously.

2.1.3. Design of algorithm access scheme

In this experiment, the algorithm access scheme design is applicable to the server accounting time for a period of time after UE4 is successfully started. And judge whether there is an algorithm access in this period of time.

In this experiment, the algorithm invokes the login interface for authentication. At the same time, verify that the user name and password are valid, and verify that the user has a running question. If the preceding authentication fails, the interface returns the failure cause, and the algorithm connection process ends. If the above verification is successful, the algorithm will call the Web interface to obtain the access key and the communication address. Then, the Web interface is called to obtain the installation sensor information of the main vehicle. Next, the algorithm invokes the UE4 interface with the secret key to send the start instruction. Next, the algorithm calls UE4 interface to obtain sensor data and master vehicle data, makes decisions, and sends control instructions according to the decisions. Finally, after running the questions, the questions are finished and the scores are calculated**(Figure 4)** .

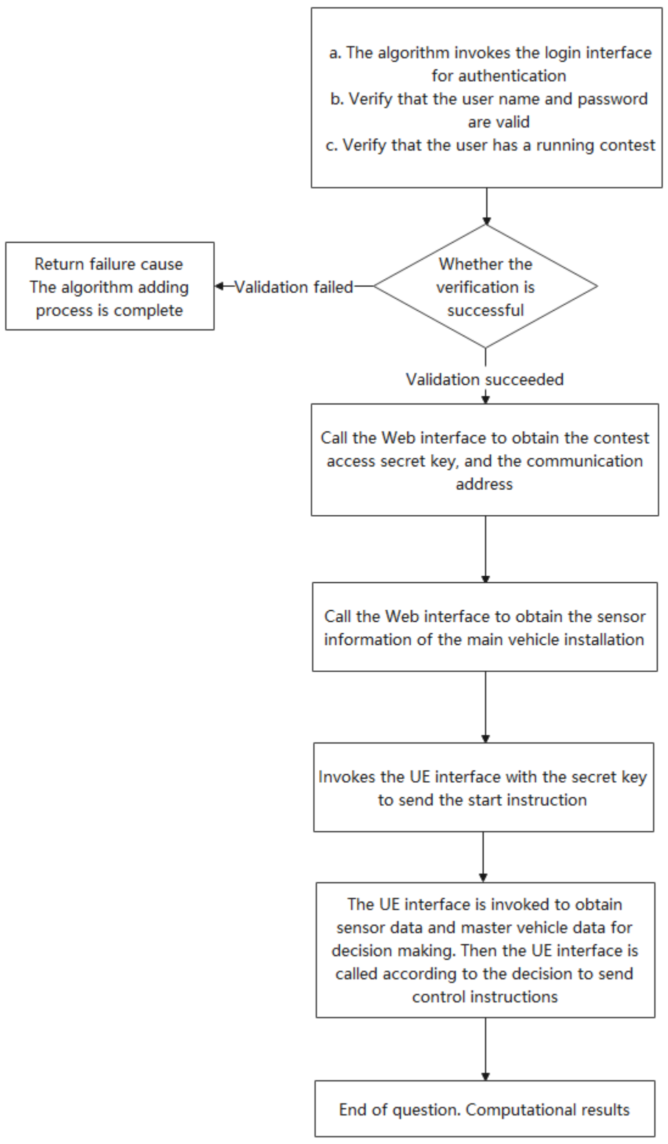

a. The algorithm invokes the login interface for authentication
b. Verify that the user name and password are valid
c. Verify that the user has a running contest
Whether the verification is successful
Validation failed
Return failure cause The algorithm adding process is complete
Validation succeeded
Call the Web interface to obtain the contest access secret key, and the communication address
Call the Web interface to obtain the sensor information of the main vehicle installation
Invokes the UE interface with the secret key to send the start instruction
The UE interface is invoked to obtain sensor data and master vehicle data for decision making. Then the UE interface is called according to the decision to send control instructions
End of question. Computational results

Figure 4 Design of algorithm access scheme

2.2. Test concurrency experimental design

In this experiment, after completing the design of the scheduling method for the intelligent connected car race platform, it is necessary to carry out a high concurrency test on this method, so as to verify that this experiment can finally solve the high concurrency requirements of intelligent connected car race based on cloud computing. Therefore, the high concurrency test is carried out from the following two aspects[10].

2.2.1. Experimental design of single server concurrency test

This experiment first needs to test how many UE4 can run on a single server. According to the requirements of this experimental competition, different kinds of sensors are installed for different kinds of questions. For example, only millimeter wave radar is installed in the decision control class, and only camera is installed in the perception class. Therefore, in this experiment, the decision control and perception questions are respectively run on the same K8s server, and the running conditions of the two different questions are represented by the number of UE4 runs. (as shown in **Table 1**)

Table 1 K8s server parameters in this experiment

Server model	p2v.2xlarge.8
vCPUs|internal storage	64vCPUs | 64GB
CPU	Intel SkyLake 6151 3.0GHz
Base/maximum bandwidth	4/10 Gbit/s
The Intranet receives and sends packets	500000
Features	GPU: 1 * NVIDIA V100 / 1 * 16G
Image	Ubuntu 16.04 server 64bit with Tesla Driver 4 18.67 and Cuda 10.1(40GB)
System disk	4T
Bandwidth	300M/s

2.2.2. Experimental design of single server concurrency test

After determining the number of UE4s running on a single K8s server, a high concurrency test experiment is required based on the overall race high concurrency requirements.（The high concurrency test environment required by the experiment is shown in **Table 2** and **Table 3**） For example, in this experiment, 100 UE4s are run in the same time period. Therefore, this experiment will set up a test environment and carry out high concurrency tests for 100 UE4s running in the same time period from four aspects: time characteristics, resource utilization, capacity and compliance of performance efficiency**(Figure 5)** .

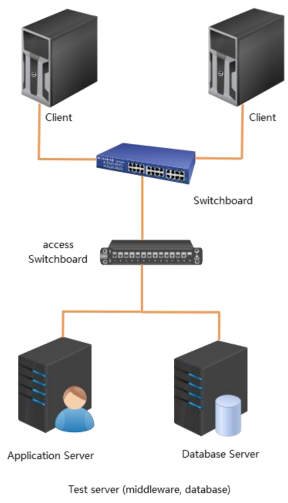


Figure 5 Test environment topology

Table 2 High concurrency test coverage table

Serial number	Test item	Test content
1	Time characteristic	The corresponding time, processing time and throughput when the product performs its functions under specified conditions.
2	Resource utilization	CPU utilization and memory utilization when executing test tasks.
3	Load	Whether the maximum number of concurrent users, communication bandwidth, transaction throughput and other parameters meet the requirements.
4	Compliance with performance efficiency	Whether the product complies with the requirements specification, product description, etc., as well as the performance and efficiency requirements in the standard.

Table 3 High concurrency test environment

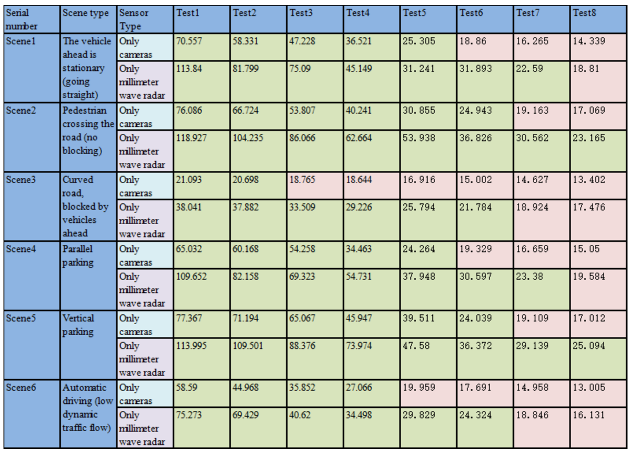
Testing the hardware environment			
Serial number	The name of the hardware	Configuration	Usage
1	The test PC	CPU:30GHZ Internal Storage：176G The hard disk：40G The operating system：windows server 2016 std. Bandwidth：100M/s	Deploying pressure Tools
Testing the Software environment			
Serial number	The name of the Software	Version number	Usage
1	Python	3.6.5rc1	Running the test script
2	The operating sys-tem	windows server 2016 std.	The pressure script is deployed on the Windows server

3. Results analysis

Based on the scheduling method of intelligent connected car race platform, this method is tested with high concurrency in this experiment. The individual server concurrency was first tested to determine the amount of UE4 a single server could run. This was followed by a high concurrency test for large-scale events (this experiment required running 100 UE4 high concurrency tests in the same time period). Finally, after several rounds of testing and code tuning, the experimental results are as follows:

3.1. Single server concurrency test result

According to the different question types of the competition, this experiment selected six types of scenes in the competition: stationary front vehicle (straight line), pedestrian crossing the road (no cover), curved road, obstructed front vehicle, horizontal parking, vertical parking, automatic driving (low dynamic traffic flow). After 8 tests, the test results are shown in **Figure 6**. Marked red indicates that the server runs at a frame rate below 20fps, and marked green indicates that the server runs at a frame rate above 20fps.



Serial number	Scene type	Sensor Type	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8
Scene1	The vehicle ahead is stationary (going straight)	Only cameras	70.557	58.331	47.228	36.521	25. 305	18. 86	16. 265	14. 339
		Only millimeter wave radar	113.84	81.799	75.09	45.149	31. 241	31. 893	22. 59	18. 81
Scene2	Pedestrian crossing the road (no blocking)	Only cameras	76.086	66.724	53.807	40.241	30. 855	24. 943	19. 163	17. 069
		Only millimeter wave radar	118.927	104.235	86.066	62.664	53. 938	36. 826	30. 562	23. 165
Scene3	Curved road, blocked by vehicles ahead	Only cameras	21.093	20.698	18.765	18.644	16. 916	15. 002	14. 627	13. 402
		Only millimeter wave radar	38.041	37.882	33.509	29.226	25. 794	21. 784	18. 924	17. 476
Scene4	Parallel parking	Only cameras	65.032	60.168	54.258	34.463	24. 264	19. 329	16. 659	15. 05
		Only millimeter wave radar	109.652	82.158	69.323	54.731	37. 948	30. 597	23. 38	19. 584
Scene5	Vertical parking	Only cameras	77.367	71.194	65.067	45.947	39. 511	24. 039	19. 109	17. 012
		Only millimeter wave radar	113.995	109.501	88.376	73.974	47. 58	36. 372	29. 139	25. 094
Scene6	Automatic driving (low dynamic traffic flow)	Only cameras	58.59	44.968	35.852	27.066	19. 959	17. 691	14. 958	13. 005
		Only millimeter wave radar	75.273	69.429	40.62	34.498	29. 829	24. 324	18. 846	16. 131

Figure 6 Test the concurrency of a single server in different scenarios

In order to further guarantee the smooth operation of a single K8s server, this experiment further averages the number of UE4 running in the above different scenarios. In addition, if the frame rate of a single server is above 24fps, it is qualified. Therefore, after calculation, the number of UE4 questions in decision control category and running perception category can be run by a single server is 6 and 5 respectively (as shown in **Table 4**).

Table 4 Test the concurrency of a single server in different scenarios

Type of Competition Question	Quantity (unit)	Frame rate （fps）	Amount of bandwidth （Mbps）
Decision control questions (only millimeter wave radar is installed)	1	94.95	21.8
	2	80.83	43.6
	3	65.49	65.4
	4	50.04	87.2
	5	37.72	108.8
	6	30.29	130.4
	7	23.90	152
	8	20.04	173.6
Perceptual questions (only cameras are installed)	Quantity (unit)	Frame rate （fps）	Amount of bandwidth （Mbps）
	1	61.45	43.2
	2	53.68	86.4
	3	45.82	129.9
	4	33.81	172.8
	5	26.13	216
	6	19.97	259.2
	7	16.79	302.4
	8	14.97	345.6

Because the event needs to run both decision control and perception questions at the same time, this experiment finally determines that the number of UE4 that can be run by a single K8s server is 5.

3.2.High concurrency test results applied to large events

According to the requirement that a single K8s server can run 5 UE4, 20 K8s servers are selected for the high concurrency test of 100 concurrency. Then the monitoring data is randomly selected and the maximum value is recorded. After several rounds of testing and code tuning, the monitoring data is extracted as follows:

This highly concurrent test involves a total of 5 interfaces (system login, running the contest, obtaining the data of the main vehicle, obtaining the monitoring data, and the algorithm). In addition, 100 concurrent tests can be started in both the running contest and the data acquisition history, so the concurrency test passes (as shown in **Table 5**).

Table 5 Result of 100 Concurrent tests started in the same period

Test project	Test scene	Expected indicators	Actual indicators	Test result
1	System Login	100 concurrent quantity	100 concurrent quantity	Pass
2	Running questions	100 concurrent quantity	100 concurrent quantity	Pass
3	Obtain ego vehicle data	100 concurrent quantity	100 concurrent quantity	Pass
4	Obtain monitoring data	100 concurrent quantity	100 concurrent quantity	Pass
5	Algorithm	frame rate>30fps	28<frame rate<30fps	Pass

4. Conclusion

In this paper, the high concurrency solution of intelligent connected auto race based on cloud computing is studied. By designing the scheduling method of the intelligent connected car race platform, and through high concurrency test, the validity and reliability of the scheme are verified. The competition is further ensured through the cloud server, which guarantees the high concurrent demand of the intelligent connected car event based on cloud computing for competitors from different regions to go online at the same time. At the same time, it also ensures that all teams can see the ranking of each team in real time. In addition, ensure that the competition is fair, impartial and open. Especially when international exchanges are affected by the epidemic, it guarantees the participation of domestic and foreign teams in competitions and effective technical exchanges, and has a broad application prospect.

5. Reference

[1] Wang Run-min, ZHAO Xiang-mo, XU Zhi-gang, WANG Wen-Wei, CHENG Jing-Jun. A kind of automatic driving vehicle in ring virtual simulation test platform design [J/OL]. Automotive technology: 1-7 [2021-08-29]. HTTP: / / https://doi.org/10.19620/j.cnki.1000-3703.20210130.

[2] Lin Fan, Zhang Qiuzhen, Yang Feng. Internet of Things Technology,2020,10(09):65-68.

[3] Wen Long. High Performance Computing Simulation Platform for Intelligent Connected Vehicle [D]. University of Electronic Science and Technology of China,2020.

[4] Li Xue, Wang Fang. Application of Cloud Computing in Building Large-scale Highly Concurrent Websites [J]. Computers and Networks,2021,47(11):40-42.

[5] Li Si-li, Yang Jing-Rong, Gou Qiang. Research and Implementation of High Concurrency Technology for Lightweight Web Server [J]. Computer Technology and Development,2020,30(10):75-78+85.

[6] Zhao Guang, Ju Changjiang, Pan Jingpeng. Design and Practice of Building Resilient and highly Available Architecture for Massive and highly concurrent data Collection [J]. New Industrialization,2020,10(09):58-62.

[7] Yu Lian-he. Fast Construction of Custom Task Scheduling System [J]. Telecommunication Letters,2020(08):30-32.

[8] Tian Yang-feng, WANG Zhen. Research and Analysis of PaaS Architecture and industry typical Products based on K8s [J]. Science and Technology Innovation,2018(06):97-98.]

[9] Jin Ziwei. Design and implementation of Docker distributed container automatic operation and maintenance system based on K8S [D]. Central South University for Nationalities,2018.

[10] Yang Di. Microservice System based on Container Cloud [J]. Telecommunication Science,2018,34(09):169-178.