講演番号	035

Approach to improve the efficiency of software development for Advanced Driver Assistance Systems by using Continuous Integration (CI)

-(Second Report) Scalability and Test Parallelization using Cloud Computing-

Song Ye [1)] **Takuro Yuhara** [1)] **Tomonori Nambu** [1)]

1) Nissan Motor Co., Ltd.
560-2, Okatsukoku, Atsugi-shi, Kanagawa 243-0192, Japan (E-mail: song-ye@mail.nissan.co.jp)

KEY WORDS: **Software and its underlying technologies, CI/CT, DevOps,** cloud computing, SDV

As the automotive industry transitions into the Software-Defined Vehicle (SDV) era, software is becoming increasingly large and complex. To maintain continuous value delivery, frequent feature additions and bug fixes are essential, driving a surge in demand for Continuous Integration (CI) resources. However, traditional on-premises CI environments face significant physical resource constraints. During peak development periods, CI jobs often exceed the available runners, leading to long wait times in the queue. Furthermore, the massive number of test cases required for strict functional safety compliance forces sequential execution, hindering development efficiency.

To overcome these challenges, this paper proposes a scalable, cloud-based CI environment designed to optimize both execution time and costs. While a container orchestration system is efficient for container-based runners, this paper discusses an architecture targeting Virtual Machine based runners. As illustrated in Figure 1, the overall architecture of the proposed CI environment consists of three main components: an auto-scaling system managed by a Runner Manager, a high-speed runner initialization mechanism utilizing filesystem snapshots, and a parallel test execution framework powered by Serverless Functions.

(1) Auto-Scaling System: CI demand fluctuates significantly depending on the time of day and the development cycle. The proposed auto-scaling system dynamically provisions and terminates Virtual Machine (VM) runners based on the actual queue of pending jobs. This ensures that resources are only consumed when needed, eliminating the idle costs associated with fixed on-premises resources while preventing job bottlenecks during peak hours.

(2) Filesystem Snapshots: A major challenge in cloud CI is the time required to download and deploy hundreds of gigabytes of necessary data (e.g., source code, control models, test data) for every newly created runner. By maintaining pre-configured filesystem snapshots that are synchronized daily with the version control system and test databases, the initialization time was drastically reduced.

(3) Test Parallelization via Serverless Functions: To accelerate the Software-In-the-Loop (SIL) testing phase, which involves over 10,000 test cases, the system leverages Serverless Functions. This allows tests to be distributed across 1,000 compute nodes simultaneously without the need for server management.

The implementation of these cloud-based strategies yielded remarkable improvements in a real-world ADAS project. The use of snapshots reduced the runner data preparation time from approximately 400 minutes to an average of just 2 minutes. Consequently, the total runner provisioning time was brought down to about 5 minutes. Moreover, parallelizing the SIL tests compressed the total test execution lead time from 924 minutes (sequential execution) to 126 minutes. This represents a roughly sevenfold increase in speed, even when accounting for the new overheads introduced by data transfer to shared filesystems and artifact aggregation.

Ultimately, this paper demonstrates that intelligently leveraging cloud scalability, snapshots, and serverless architectures can resolve the physical limitations of on-premises environments. These measures significantly reduce wait times and operational costs, maximizing development efficiency and accelerating the CI/CD feedback loop for next-generation SDV development. Future work will focus on establishing a dynamic method for automatically selecting the optimal performance specifications for CI runners to further balance cost and performance.

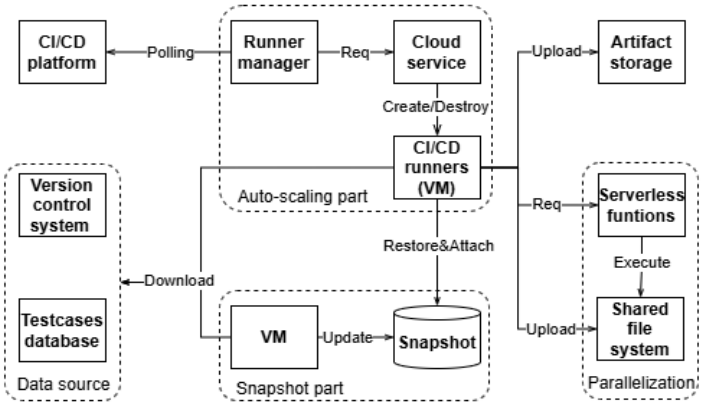

Fig.1 Overall architecture of the proposed CI