클라우드와 자동차 엣지 간 환경 패리티를 통해 소프트웨어 정의 차량 개발을 가속화하는 방법

기리쉬 쉬라삿(Girish Shirasat)
Arm 오토모티브 부문 소프트웨어 전략 및 아키텍처 디렉터
스테파노 마르자니(Stefano Marzani)
AWS 자율주행차 부문 수석 스페셜리스트 솔루션 아키텍트

arm

aws

백서

비전

2025년, 주요 자동차 공급업체의 소프트웨어 엔지니어인 주디스(Judith)는 재택근무를 하며 자동 크루즈 컨트롤 기능에 대한 고객의 피드백을 처리하고 있다. 그녀는 클라우드 상의 작업 공간 내에서 문제 동작과 관련된 방대한 데이터 세트에 액세스하고 이를 제어 알고리즘 조정에 활용한다. 차량 다이내믹 시뮬레이터와 함께 SiL(software-in-the-loop) 테스트벤치를 사용해 새로운 성능을 평가하고 검증한 다음 패키지를 배포할 준비가 되었음을 표시한다. 한편, 수천 마일 떨어진 다른 대륙의 디자인 회사에서 일하고 있는 마크(Mark)는 다음 주에 자동차 출시가 예정된 새로운 지역에 현지화된 UI 소프트웨어의 마무리 작업을 하고 있다. 그는 OEM 업체가 제공하는 클라우드 패널에서 수정 작업을 하며 가상 에이전트를 활용하는 대규모 병렬 테스트 제품군을 실행한다. 테스트를 통과하면 새로운 UI 패키지를 배포 준비 완료로 표시한다. 데브옵스(DevOps) 검증 엔지니어인 케이(Kay)는 OEM 본사에서 근무하고 있다. 그는 두 개의 새로운 소프트웨어 패키지를 보고 최종 HiL(hardware-in-the-loop) 테스트 제품군을 실행하여 다가오는 주간 릴리스에서 전체 생산 차량에 배포될 콘텐츠를 완전히 검증하고 인증한다.

소개

자동차 산업이 소프트웨어 정의 시대를 맞이함에 따라, 많은 OEM 업체들이 추구하는 비전은 위에 묘사된 것과 같이 민첩성과 유연성을 갖춘 소프트웨어를 개발함으로써 품질이나 안전성에 대한 타협 없이 자동차에 기능을 점진적으로 제공할 수 있도록 하는 것이다.

기능 안전 개념과 실시간 실행 측면에서 자동차 고유의 특성을 보존할 수 있는 클라우드 네이티브 접근법[1]은 최신 디지털 서비스와 사용자 친화적인 오토모티브 애플리케이션을 통해 이러한 소프트웨어 중심 에코시스템을 형성하는 핵심 요소이다. 혁신적이고 효율적인 워크플로우는 더 많은 개발자가 개발 프로세스에 참여할 수 있게 한다. 또한 자동차 업체는 개발 시간을 단축하고, 현대 소비자의 기대에 부응하기 위해 빠르게 진화하고 기능을 업데이트하는데 필요한 민첩성을 확보할 수 있다.

오토모티브 클라우드 네이티브 개발 파이프라인을 구현하는 첫 번째 핵심 기술 요소는 클라우드 환경과 대상이 되는 임베디드 자동차 엣지 플랫폼 간에 환경 패리티(environmental parity)를 달성하여 워크로드를 배포할 수 있게 하는 것이다. 정보 아키텍처(IA) 전문가 케빈 호프만(Kevin Hoffman)은 다음과 같이 설명한다[2]:

*"**환경 패리티**에 엄격함과 규칙을 적용하는 이유는 애플리케이션이 **어디에서나 작동**할 것이라는 확신을 팀과 조직 전체에 주기 위해서다."*

환경 패리티를 달성하면 개발자가 보유한 임베디드 하드웨어에 구애받지 않고 클라우드 내에서 개발 및 검증할 수 있으며, 차량의 가치사슬 전체에 걸쳐 솔루션의 시장 출시 시간을 획기적으로 단축함으로써 애플리케이션 이식성을 보장하고 미래에 대비한 개발 워크플로우를 만들 수 있다.

환경 패리티를 달성하고 증가하는 자동차 소프트웨어에 대한 수요를 지원하기 위해서는[3], 차량 내부에 적절하게 설계된 멀티 코어 임베디드 시스템이 필요하다. 본 백서에서는 특히 클라우드 내 실행 환경과 자동차 엣지 간 환경 패리티를 달성하는 데 초점을 맞춰 오토모티브 시스템 개발에 대한 클라우드 네이티브 접근법을 구현하는 방법을 살펴보고, 이 접근법이 소프트웨어 정의 차량의 시장 출시 기간을 단축하는 데 미치는 영향을 살펴본다.

소프트웨어 정의 차량(SDV)이란?

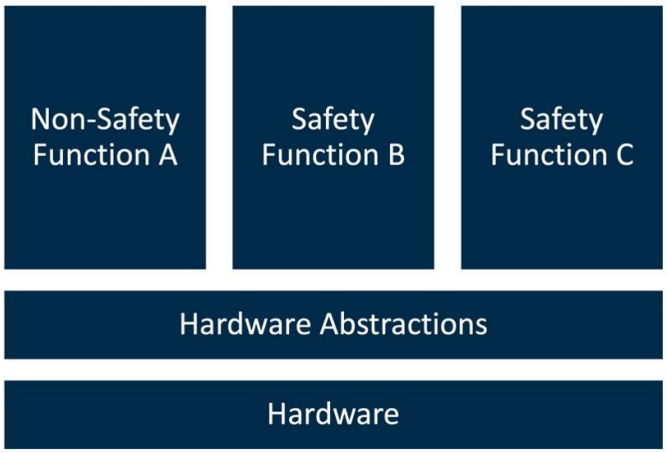


그림 1 - 추상화된 하드웨어 위에 탑재된 자동차 기능

소프트웨어 정의 차량에서 '소프트웨어 정의'라 함은, 개별의 물리적 ECU 제어 장치 또는 이와 유사한 장치로 구현되는 것과 달리, 공유 또는 중앙 집중식 컴퓨팅에서 실행되는 소프트웨어 기능과 서비스로서 자동차 기능의 특성화와 구현으로 설명될 수 있다. 또한, '소프트웨어 정의'는 제조 전후를 포함한 전체 오토모티브 시스템 개발 수명 주기 동안 이러한 소프트웨어 정의 기능을 민첩하게 개발하고 배포할 수 있는 능력을 내포한다.
그림 1에서 볼 수 있듯이, 이상적인 환경에서 이러한 소프트웨어 서비스는 하드웨어와 공급업체에 구애받지 않으며 특정 기능을 가능하게 하는 데이터 제공(센서), 데이터 처리(애플리케이션 로직), 데이터 소비(액추에이터) 서비스로 구성된다. 이 정도 수준의 유연성은 OEM뿐만 아니라 순수 소프트웨어 기업도 특정 애플리케이션 도메인에 대한 새로운 기능을 만들 수 있게 한다. 판매 후 새로운 기능을 추가하는 방식은 적어도 10년 넘게 스마트폰과 같은 분야에서 널리 행해져 왔으며, 이제는 자동차를 포함한 전통적인 임베디드 시장에서도 유사한 접근법을 도입하려는 수요가 크게 증가하고 있다.

SDV는 다음과 같은 특징을 가진 시스템으로 요약할 수 있다.

- 커넥티비티 기능이 제공되는 차량으로, '빅 루프' 관점에서 OTA(Over-the-Air)로 지속적으로 데이터를 전송하고 소프트웨어 업데이트 수신 가능[5].
- 기반 하드웨어에서 추상화된 소프트웨어.
- 자동차 기능이 소프트웨어를 통해 제공되며, 차량 수명 주기 동안 업그레이드 및 관리 가능.
- 자동차 개발 운영(DevOps) 관점에서, 클라우드에서 자동차 엣지에 이르기까지 하드웨어 플랫폼 전반에 걸쳐 클라우드 네이티브 설계 패러다임을 채택.
- 오토모티브 애플리케이션의 혼재된 중요도 관리를 통해 다양한 수준에서 워크로드 장애 대비[6].

SDV에서의 클라우드 네이티브

'소프트웨어 정의'는 새로운 개념이 아니다. 위에서 예시로 언급한 스마트폰 외에도, 소프트웨어 정의 네트워킹, 소프트웨어 정의 스토리지, 소프트웨어 정의 컴퓨팅 설계 패러다임을 통해 소프트웨어 정의는 통신과 데이터센터에 이미 적용되고 있었으며, 엔터프라이즈 애플리케이션 도메인에서도 널리 사용되고 있다. 이러한 도메인에서 클라우드 네이티브 개념은 풍부한 설계 패턴과 에코시스템 툴을 통해 소프트웨어 정의 시스템을 구현하는 데 성공적으로 사용되어 왔다[12]. 따라서 오토모티브 분야에 클라우드 네이티브 개념을 적용하고, 이를 중심으로 개발된 방대한 기술·상업 에코시스템을 활용해 혁신의 효과와 속도를 높이려는 노력은 어찌 보면 당연한 것으로 보인다.

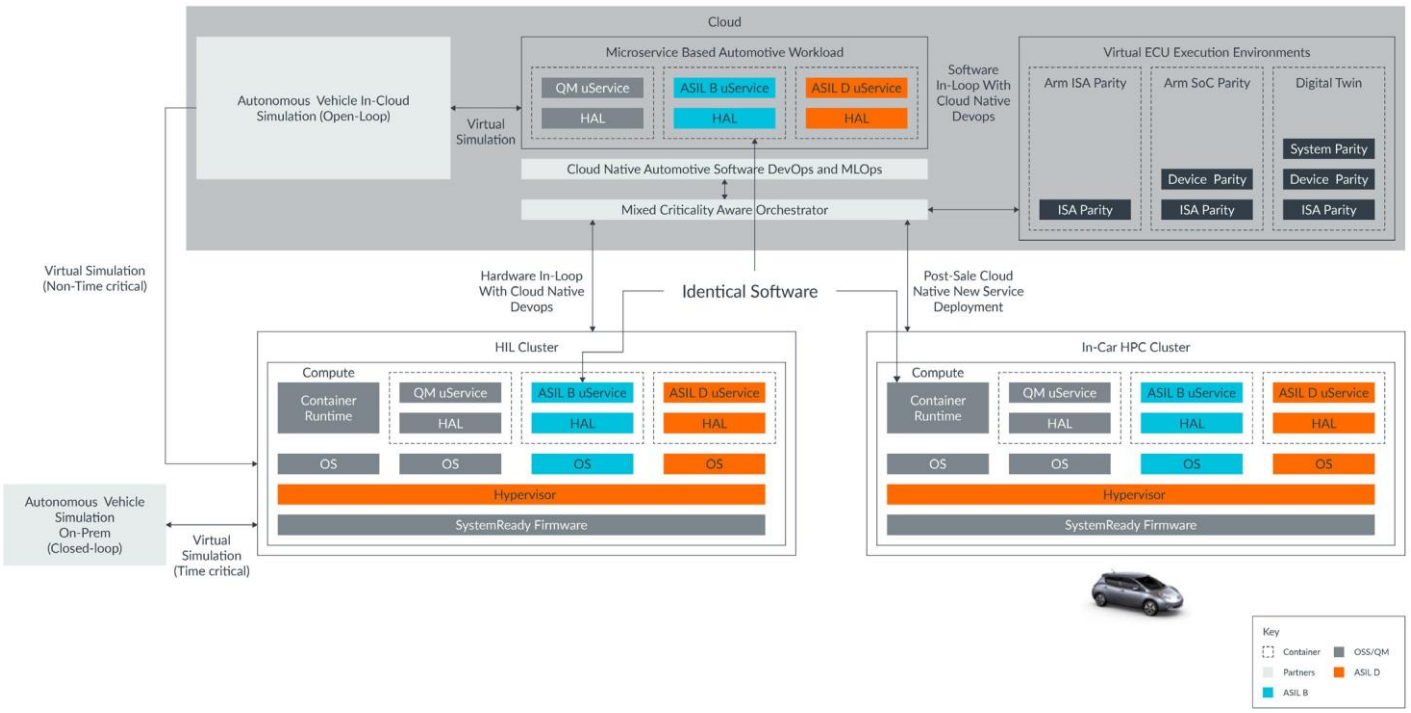


그림 2 - 오토모티브 시스템 개발에 적용되는 클라우드 네이티브

*그림 2*를 보면 클라우드와 자동차 엣지 간의 환경 패리티를 가정하여, 개발자가 카페에서 노트북을 통해 클라우드에 접속하고, 적응형 크루즈 컨트롤이나 측선 제어와 같은 인포테인먼트 또는 첨단 운전자 보조 시스템(ADAS)과 애플리케이션을 개발하고, 코드를 커밋한 후, 물리적인 차량 대상 시스템을 자세하게 나타내는 가상 실행 환경에서 클라우드 내 설계 및 통합 주기를 트리거 하는 모습을 상상할 수 있다. 클라우드와 자동차 엣지 간 달성할 수 있는 환경 패리티 수준과 시뮬레이션 속도는 개발자의 효율성에 직접적인 영향을 미친다.

이제 오토모티브 시스템 소프트웨어의 복잡성이 보잉 787 드림라이너[7] 항공기 수준을 넘어서기 시작하면서, 오토모티브 개발자는 서비스 지향 아키텍처/마이크로 서비스 기반의 검증된 소프트웨어 설계 패턴을 사용하는 것을 선호할 것이다[11, 12]. 이 패턴에서는 컨테이너가 워크로드 이동성과 복잡성을 확보하기 위한 기초 기술을 구성한다. 오토모티브 애플리케이션 개발은 컨테이너를 일상적으로 활용하던 일반적인 기업이나 스마트폰 부문에서는 고려되지 않던 특수한 어려움을 수반한다. 오토모티브 워크로드는 본질적으로 중요도가 혼재될 가능성이 있다. 안전과 실시간 요구사항에 따라, 워크로드를 형성하는 마이크로서비스가 해당 안전 분해(decomposition) 외에도 특정한 공간적, 시간적 요구사항에 의해 제한되기 때문이다. 애플리케이션에 따라, 이러한 마이크로서비스 중 일부는 품질 관리(QM)를 받거나 ISO26262 규격에 정의된 ASIL-B/Avcv-D 무결성 수준을 달성하는 등의 조건을 요구할 수 있다. 이는 클라우드 기반 툴링 프레임워크의 일부로 통합될 안전 인증 컴파일러와 툴에 대한 요구사항으로 이어진다.

또한, 오토모티브 애플리케이션이 배포되는 하드웨어는 전통적으로 매우 분산되어 있고(최신 자동차 한 대에는 100개 이상의 전자제어장치(ECU)가 탑재될 수 있음), 본질적으로 종류가 매우 다양하므로 하드웨어에서 소프트웨어를 분리하기가 어렵다. 마지막으로, 엔터프라이즈급 플랫폼 인식을 기반으로 최적의 하드웨어 노드에 마이크로서비스를 배포할 수 있는 오케스트레이터를 포함하는 기존의 클라우드 네이티브 인프라는 이처럼 고유한 임베디드 자동차 하드웨어 시스템의 기능을 이해할 수 있어야 하며 최적의 방식으로 마이크로서비스를 배포할 수 있도록 확장되어야 한다.

이러한 모든 요구사항과 문제에 대해 적절한 솔루션을 찾았을 때, 개발자는 확장성과 탄력성과 같은 클라우드의 고유한 이점을 활용하여 클라우드에서 시뮬레이션을 안전하게 실행할 수 있다. 예를 들어 데브옵스(DevOps) 인프라의 일부로서, 클라우드에서 실행되는 시뮬레이터를 사용하여 다양한 시뮬레이션 운영 설계 도메인(ODD)을 실행하고, 방대한 시뮬레이션 데이터를 테스트 대상 소프트웨어에 입력하고 신속하게 출력을 검증함으로써 완전한 SiL 검증을 실행할 수 있을 것으로 기대된다. 이렇게 방대한 실제 또는 시뮬레이션 환경 세트는 수천 개의 시나리오를 동시에 검증하도록 확장될 수 있으며, 수천 개의 코어에서 병렬 실행을 시작할 수 있다. 이는 HiL(hardware-in-the-loop) 장비의 임베디드 시스템에 의존해서는 달성할 수 없는 규모다.

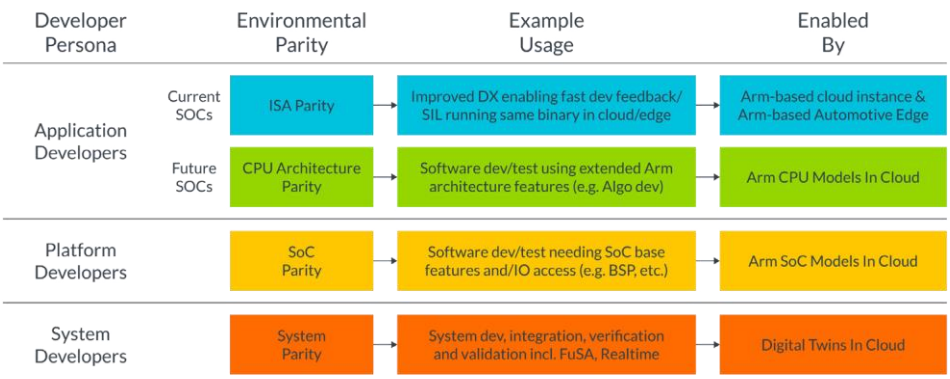
고객을 대상으로 한 예비 조사에 따르면, 환경 패리티를 달성할 시 현재 HiL 장비에서 실행되는 테스트의 약 70%를 클라우드 기반 SiL 환경으로 이동함으로써 이러한 클라우드 확장성을 활용할 수 있을 것으로 예상된다.

여전히 일부 기능(예: 입출력 테스트)은 임베디드 시스템의 물리적인 특성으로 인해 하드웨어에서 항상 검증되어야 하지만, 클라우드에서 자동차 엣지로 이동함에 따라, 시스템 개발 관점에서 볼 때 클라우드에서 검증된 동일한 소프트웨어를 클라우드 네이티브 오케스트레이션 기술을 통해 물리적 ECU에 배포함으로써, 출시 후 도로에서 운행되고 있는 차량에 대한 배포로 확장하기 전에 실험실 내 HiL 검증을 수행할 수 있을 것으로 기대된다.

오토모티브에서 클라우드 네이티브 구현을 위한 '클라우드 투 자동차 엣지' 환경 패리티

클라우드 네이티브 오토모티브 시스템 개발을 위한 핵심적인 인프라 요구사항 중 하나는 개발, 통합, 검증 목적을 위한 다양한 가상 시스템 실행 환경에 대한 요구사항으로, 물리적 환경과 매우 유사한 최상의 자동차 엣지 환경 패리티를 달성해야 한다. 높은 수준의 환경 패리티를 달성하는 것은 개발자의 피드백 루프에 직접적인 영향을 미치며 개발 효율을 높일 수 있다[8]. 다양한 개발자 관점에서 환경 패리티 측면을 좀 더 자세히 살펴보고 다양한 패리티와 페르소나를 검토하여 클라우드에서 환경 패리티를 확보하는 방법에 대해 알아보고자 한다.

그림 3 - 클라우드와 자동차 엣지 간 환경 패리티



애플리케이션 개발자

오토모티브 개발자의 관점에서 클라우드 기반 실행 환경에 대해 기본적으로 기대하는 것은 클라우드와 자동차 엣지 환경 간에 명령어 집합 구조(ISA)와 CPU 아키텍처 간 패리티를 확보하는 것이다. Arm 기반 오토모티브 컴퓨팅 플랫폼을 예로 들자면, 이제 AWS 그래비톤(Graviton)이 제공하는 새로운 Arm 기반 클라우드 인스턴스에서 클라우드 환경을 호스팅할 수 있게 됐다. 이제 완전한 엔드 투 엔드의 Arm 기반 '클라우드 투 자동차 엣지' 컴퓨팅 연속체를 구현할 수 있게 됐다는 점에서 이는 획기적이다. 이것이 이상적인 개발 환경으로 간주될 수 있는 이유는 다음과 같다.

- 개발자 효율성: 개발자 효율성 전문가 팀 코크란(Tim Cochran)이 [9]에서 확인한 바에 따르면, 개발자 피드백 루프를 축소하는 것은 기업의 순이익에 영향을 미치는 개발 효율 향상을 위해 매우 중요하다. Arm for Arm에서 코드를 개발할 경우 다음과 같은 이점이 있다.
  - 한 아키텍처의 개발 시스템에서 코드를 컴파일하여 다른 아키텍처의 대상에서 실행하는 프로세스인 크로스 컴파일을 피할 수 있다. 크로스 컴파일은 본질적으로 개발-배포-테스트 프로세스에 단계를 추가하여 개발자 피드백 루프를 늘린다.
  - 또한 크로스 컴파일은 소프트웨어에 잠재적인 오류를 초래하는 또 다른 원인이 된다.
  - Arm CPU 모델을 클라우드에서 호스팅함으로써, ISA 패리티가 부족하고 CPU 아키텍처 패리티가 필요한 경우를 지원할 수 있다. 다른 아키텍처에서의 에뮬레이션은 느리고 비효율적일 수 있다. Arm-on-Arm 환경에서는 Arm 하이퍼바이저 확장을 직접 사용할 수 있으므로 모델을 더 빠르게 실행할 수 있다.

- 성능 최적화. 컴파일러가 필요한 모든 최적화를 처리한다는 인식이 있지만, 완벽하게 최적화된 라이브러리나 기능 블록을 생성하기 위해 중요한 세그먼트의 성능을 최대로 끌어 올리려면 여전히 개발자가 아키텍처별 최적화를 직접 수행해야 하는 경우가 있으며, 이는 다양한 파트너와 서드파티 에코시스템 공급업체에서 확인할 수 있다. 이 시나리오에서는 클라우드와 엣지 패리티를 통해 개발에서 배포까지 이러한 최적화의 '원활한' 전환을 보장한다.

플랫폼 개발자

애플리케이션 개발 외에도, 오늘날 차량에 들어가는 또 다른 주요 소프트웨어 계층은 펌웨어, OS/RTOS, 장치 드라이버 및 관련 구성 요소를 포함하는 기본 플랫폼 시스템 소프트웨어이다. 플랫폼 개발자는 CPU 코어를 사용하여 구축된 SoC(System-on-Chip)와 해당 SoC 시스템 아키텍처에 적합한 디바이스 인터페이스를 갖춘 추가 기능을 시뮬레이션하는 가상 환경이 필요하다. 이는 클라우드에서 가상 SoC 모델을 호스팅하고 전체 자동차 소프트웨어 CI/CD 프레임워크의 일부로 통합함으로써 구현할 수 있다. 이 부분에서 관건은 기본 시스템 아키텍처를 표준화하여 기성 OS 이미지를 플랫폼별로 거의 또는 전혀 수정하지 않고 부팅할 수 있도록 하는 것이다. 이는 광범위한 업계 지원과 함께 Arm SystemReady 인증 프로그램이 직접 담당하고 있는 과제이다.

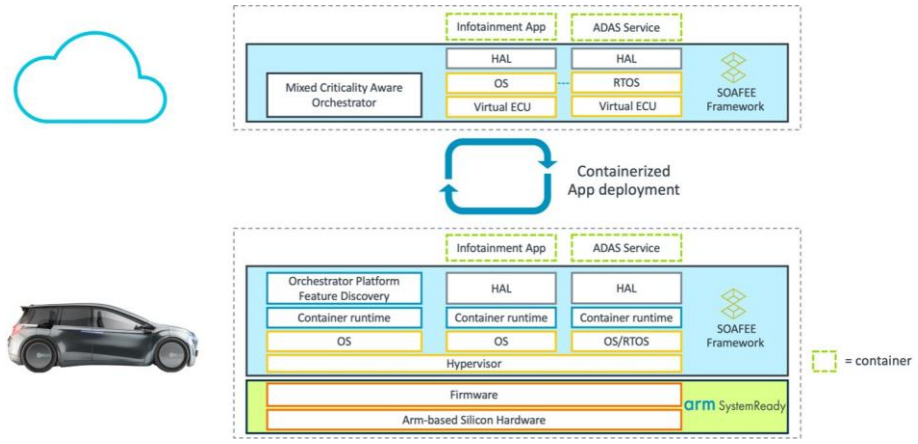
시스템 개발자

본 백서의 앞부분에서 설명한 바와 같이, 오토모티브 시스템에서 실행되는 워크로드는 본질적으로 중요도가 혼재되고 분산되어 있다. 이러한 워크로드를 개발 및 검증하려면, 실시간, 안전, 성능 특성을 포함하여 시스템의 기능적 측면과 비기능적 측면을 시뮬레이션하는 자동차의 디지털 트윈이 클라우드에 호스팅 되어있어야 한다. 또한 디지털 트윈은 클라우드에서 실행될 때, 물리적 환경에서 겪을 수 있는 다양한 운영 설계 도메인을 모델링해야 한다. 이미 오토모티브 디지털 트윈 기술을 개발 중인 주요 에코시스템 파트너들은 환경 패리티 격차를 해소하는 데 있어 상당한 진전을 보이고 있다.

SOAFEE –오토모티브 부문의 클라우드 네이티브를 촉진하기 위한 업계 주도 이니셔티브

카페에서 접속가능한 ADAS 개발자의 비전을 현실로 만들기 위해서는 가치 사슬 전반의 파트너 에코시스템이 함께 협력해야 한다. Arm은 AWS와 같은 주요 파트너와의 협력을 통해, '임베디드 엣지를 위한 확장 가능한 오픈 아키텍처(Scalable Open Architecture for Embedded Edge)'라는 SOAFEE 이니셔티브를 최근 발표했다. SOAFEE는 오토모티브 애플리케이션의 혼재된 중요도에 맞게 강화된 클라우드 네이티브 아키텍처를 제공할 뿐만 아니라 상업용 및 비상업용 오퍼링을 가능하게 하는 오픈소스 참조 구현도 제공할 것이다. 개방형 표준을 기반으로 SOAFEE 아키텍처를 정의하고 오픈소스 구현을 생성하는 업계 주도의 SIG(Special Interest Group)가 SOAFEE를 주도하고 있다. 이 이니셔티브에 대한 자세한 내용은 SOAFEE 웹사이트에서 확인할 수 있다.

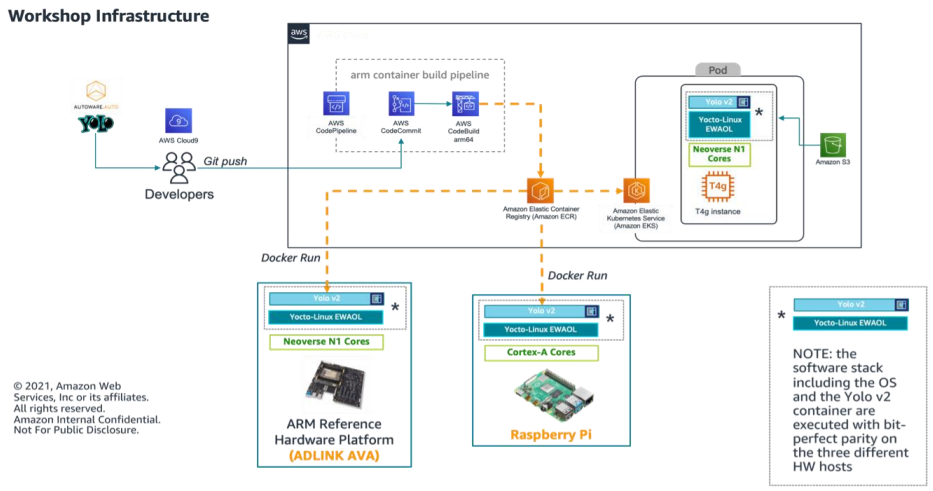
그림 5 - SOAFEE 상위 수준 레퍼런스 아키텍처



Arm DevSummit 워크숍 – 초기 SOAFEE 기반 구현을 사용한 데모

Arm DevSummit 2021[9]의 라이브 워크숍에서 SOAFEE의 첫 번째 참조 구현이 소개됐다.

그림 6 - 클라우드-자동차 엣지 패리티를 달성한 자동차 개발 파이프라인



묘사된 아키텍처와 같이, 이 워크숍에서는 AWS EC2 그래비톤2 인스턴스, Raspberry Pi, AVA 개발자 플랫폼과 같은 대상 컴퓨팅 요소 세트에서 환경 패리티를 달성하고, 컨테이너화된 동일한 워크로드를 실행할 수 있는 새로운 오토모티브 네이티브 소프트웨어 개발 인프라가 소개됐다.

이 아키텍처에서는, AWS 서비스를 사용하여 인식 네트워크인 YOLO의 클라우드 및 임베디드 디바이스에서 대규모로 구축, 컨테이너화, 평가 및 배포를 활성화하는 CI/CD 파이프라인을 생성한다. 이 네트워크는 설계 패러다임을 시연하기 위해 오토모티브 애플리케이션 워크로드에 대한 대용으로 사용된다. 이 워크숍에서 사용된 YOLO 버전은 우분투 리눅스 20.04에서 실행되는 YOLOv2-Tiny이다. 참고로 SOAFEE 아키텍처의 참조 구현인 Arm의 EWAOL(Edge Workload Abstraction and Orchestration Layer) 덕분에 클라우드에서 기본 속성으로 실행되는 임베디드 운영 체제(Yocto-Linux 배포판)가 전체 SUT(system under test) 스택에 처음으로 포함됐다.

그림 7 - SUT(테스트 대상 시스템): OS 부터 풀스택

OS 부터 모든 SUT 구성요소는 모든 대상에 대해 동일한 Aarch64 기본 아키텍처를 활용하는 명령어 집합 패리티(instruction set parity)와 함께 실행되었다. Arm의 EWAOL 은 여러 임베디드 플랫폼에서 애플리케이션의 배포 및 오케스트레이션에 컨테이너를 사용하는 표준 기반 프레임워크를 사용자에게 제공한다. 이 결합된 기능 세트는 다음을 지원한다.

- 개발 중인 소프트웨어 유닛(이 예시에서는 YOLOv2-Tiny 인식 모듈 자체)뿐만 아니라 임베디드 OS부터 전체 임베디드 소프트웨어 스택의 클라우드 실행.
- 클라우드에서 임베디드 엣지까지 SUT의 완벽한 이식성 지원. 더 이상 크로스 컴파일이나 에뮬레이션(그리고 컴파일 오류 또는 성능 저하와 같은 관련 문제)할 필요가 없다.

실제로 이 접근법을 통해, *클라우드에서 임베디드 코드* 작성 및 테스트를 시작하고, 임베디드 개발 워크플로우를 시프트 레프트(shift left) 하며, 클라우드 확장성 기능을 활용해 테스트 범위를 크게 확장할 수 있다(이 예시에서는 여러 SUT 실행을 병렬로 작동하기 위해 AWS Batch를 사용).

결론 및 향후 활동

결론적으로, 소프트웨어 정의 차량(SDV)은 분명히 미래의 얘기가 아니라 이미 일어나고 있는 현재의 주요 트렌드이며, 이는 오토모티브 에코시스템 전반에 걸친 사례에서 찾아볼 수 있다. 앞으로 소프트웨어는 OEM 경쟁사 간의 주요 차별화 요소가 될 것이며, 소프트웨어 개발자는 향후 오토모티브 시스템 개발에서 가장 중요한 페르소나 중 하나가 될 것이다. 이렇게 중대한 변화 속에서, 업계는 기존의 클라우드 네이티브 기술을 활용하여 해결할 수 있고, 해결해야만 하는 과제들이 있다. 그렇기 때문에, SOAFEE와 같은 이니셔티브가 오토모티브 부문의 클라우드 네이티브의 도입과 가속화에 중요한 역할을 한다.

클라우드 네이티브 오토모티브 소프트웨어 개발을 위한 첫 번째 핵심 기술은 오토모티브 소프트웨어 개발자가 클라우드에서 소프트웨어를 개발 및 검증할 수 있도록 클라우드 기반 개발 환경과 자동차 엣지 간 환경 패리티를 달성하는 것이다. 이는 시프트 레프트, 스케일 아웃, 판매 후 배포 모델을 지원함으로써, 오토모티브 시스템 개발의 효율성을 높이고 자동차 산업을 변혁할 수 있는 혁신적인 새로운 비즈니스 모델을 촉진할 것이다. 이러한 패리티의 예로 현재 사용 가능한 Arm 기반 오토모티브 대상 시스템과 AWS 그래비톤 인스턴스가 있으며, 이는 Arm DevSummit 2021[9]에서 엔드 투 엔드로 시연되었다.

마지막으로, 새로운 기술과 개념을 개발하는 데 있어, 개념을 시연하기 위한 코드를 개발하는 것, 그리고 개발자들이 이러한 에코시스템을 혁신하고 키워나갈 수 있는 플랫폼을 제공하는 것이 중요하다. AWS와의 DevSummit 워크숍 시연 외에도, Arm은 최근 오토웨어 파운데이션(Autoware Foundation・AWF)과의 협력을 통해 Open AD Kit[13]를 발표했다. Open AD Kit는 오토웨어를 AWS 그래비톤 인스턴스에서 바로 참조 자율 소프트웨어 스택으로 개발할 수 있도록 하며 클라우드 네이티브 오케스트레이터를 사용해 자동차 엣지 플랫폼에 원활하게 배포할 수 있게 해주는 완전한 엔드 투 엔드 개발 키트를 제공한다.

이렇게 자동차 산업의 혁신 여정이 시작되었다. Arm, AWS와 SOAFEE SIG의 회원들은 오토모티브에서 클라우드 네이티브 구현을 가속화하기 위해 계속 협력해나갈 것이다. 이러한 비전을 실현하고자 하는 열정을 가진 분들은 SOAFEE SIG에 함께할 수 있으며, 자세한 내용은 soafee.io/에서 확인할 수 있다.

참고자료

[1] 기리시 시라삿(Girish Shirasat), 소프트웨어 정의 차량에 대한 클라우드 네이티브 접근법, 2021년 9월

[2] 케빈 호프먼(Kevin Hoffman), 12 팩터(Twelve-Factor) 앱을 넘어서, 2016년 4월, 오라일리 미디어(O'Reilly Media, Inc).

[3] 로버트 N. 샬렛(Robert N. Charette), 소프트웨어가 자동차를 집어삼키는 방법, 2021년 6월

[4] 로버트 데이(Robert Day), 소프트웨어 정의 차량에는 오래 또 멀리 달릴 수 있는 하드웨어가 필요하다, 2021년 6월

[5] 콘스탄틴 길리스(Constantin Gillies), 빅 루프: 인공지능과 머신러닝, 2021년 7월

[6] 앨런 번스(Alan Burns) & 로버트 I. 데이비스(Robert I. Davis), 중요도가 혼재된 시스템 - 검토, 2019년 3월

[7] 정보는 아름답다 – 수백만 줄의 코드

[8] 마틴 파울러(Martin Fowler)의 개발자 효율성에 관한 훌륭한 글

[9] '클라우드 네이티브 오토모티브 소프트웨어 개발 시작하기' 워크숍, 영상은 여기에서, 단계별 튜토리얼은 여기에서 확인할 수 있다.

[10] 클라우드 네이티브 아키텍처로 이동하는 넷플릭스

[11] 컴퓨터 위클리(Computer Weekly) — 컨테이너화가 VW의 자동차 소프트웨어 개발을 돕는 방법

[12] 재규어 랜드로버의 데브옵스 활용

[13] 오토웨어 파운데이션(The Autoware Foundation) — 오토웨어 파운데이션, 클라우드 네이티브 자율주행 개발을 가속화시키는 퀵스타터 키트 출시

[14] Soafee 아키텍처가 중요도가 혼재된 오토모티브 시스템에 클라우드 네이티브를 적용시키는 방법