车控操作系统架构
研究报告

汽标委智能网联汽车分标委
资源管理与信息服务标准工作组

2021 年 7 月

目　录

前　言

智能网联汽车是解决汽车社会交通安全、道路拥堵、能源消耗、污染排放等问题的重要手段，也是构建智慧出行服务产业生态的核心要素和推进交通强国、数字中国、智慧城市的重要载体。

车载智能计算基础平台是智能网联汽车核心新型增量零部件。车控操作系统是智能网联汽车的基础软件部分，运行于智能网联汽车车载智能计算基础平台。智能网联汽车的复杂性，需要通过车控操作系统软件架构和接口的标准化实现产业链的分工协同，提高开发效率，保障软件平台的安全可信，减少对接成本，构建统一生态。

在此衷心感谢参加研究报告编写的各单位、组织及个人。

组织指导：汽标委智能网联汽车分标委

牵头单位：国汽(北京)智能网联汽车研究院有限公司、华为技术有限公司

参与单位：国汽智控（北京）科技有限公司、中国汽车技术研究中心有限公司、北京地平线机器人技术研发有限公司、电子科技大学、中国第一汽车股份有限公司、一汽解放汽车有限公司、上汽大众汽车有限公司、北汽福田汽车股份有限公司、惠州市德赛西威汽车电子股份有限公司、上汽通用五菱汽车股份有限公司、长城汽车股份有限公司、东风汽车有限公司东风日产乘用车公司、上海汽车集团股份有限公司零束软件分公司、大众汽车（中国）投资有限公司、中汽研软件测评（天津）有限公司、北京百度智行科技有限公司、江淮汽车集团股份有限公司、长安汽车软件科技有限公司、清华大学、东风汽车集

团有限公司技术中心、东风商用车有限公司、国家 ITS 中心智能驾驶研究院、北京汽车股份有限公司、阿里巴巴（中国）有限公司、高通无线通信技术(中国)有限公司、中国汽车工程研究院股份有限公司、中兴通讯股份有限公司、中国软件评测中心（工业和信息化部软件与集成电路促进中心)、上海机动车检测认证技术研究中心有限公司、斑马网络科技有限公司、上海瓶钵信息科技有限公司

参与人员：田思波、丛炜、潘晏涛、褚文博、周铮、程智锋、吴含冰、罗蕾、石庆鹏、张晓谦、孔涛、孟宪刚、郑岩、石景文、余得水、钱国平、伍宇志、罗覃月、周思儒、邹德英、孙华、郑四发、樊胜利、吴丹丹、唐波、周明珂、彭琪惠、徐耀宗、彭伟、冯锦文、吴琼、朱乾勇、黄懿、桂绍靖、江浩、刘珺、王琳、刘红军、李俨、凌晓峰、徐立锋、郭盈、李玉坷、高长胜、金一华、刘时珍、王锦、张路、刘晨曦、程唐平、田俊涛、马涛、王圭、许梦枚、林智桂、王萌、金天、侯昕田、丁钊、李毓强、郑方、殷苏辰、陈书平、贾元辉、秦文婷、刘大鹏、崔云峰、周波、刘海威、满志勇

1 术语定义及缩略语

1.1 术语与定义

下列术语与定义适用于本文件。

1.1.1 车载智能计算基础平台 intelligent vehicle base computing platform

支撑智能网联汽车驾驶自动化功能实现的软硬件一体化平台，包括芯片、模组、接口等硬件以及系统软件、功能软件等软件，以适应传统电子控制单元向异构高性能处理器转变的趋势。

1.1.2 车用操作系统 vehicle operating system

运行于车内的系统程序集合，以实现管理硬件资源、隐藏内部逻辑提供软件平台、提供用户程序与系统交互接口、为上层应用提供基础服务等功能，包含车控操作系统和车载操作系统。

1.1.3 车控操作系统 vehicle-controlled operating system

运行于车载智能计算基础平台异构硬件之上，支撑智能网联汽车驾驶自动化功能实现和安全可靠运行的软件集合。

1.1.4 系统软件 system software

车控操作系统中支撑驾驶自动化功能实现的复杂大规模嵌入式系统运行环境，分为安全车控系统软件和智能驾驶系统软件。

1.1.5 功能软件 function software

车控操作系统中根据面向服务的架构设计理念，通过提取智能驾驶核心共性需求，形成智能驾驶各共性服务功能模块，高效实现驾驶自动化功能开发的软件模块。

1.1.6 域控制器 domain controller unit

根据汽车电子部件功能将整车划分为动力总成、智能座舱和智能驾驶等几个域，集中控制域内原本归属各个 ECU 的大部分功能，以取代传统的分布式架构。

1.1.7 驾驶自动化功能 driving automation feature

驾驶自动化系统在特定的设计运行范围内执行动态驾驶任务的能力。

1.1.8 实时安全域 realtime safety domain

车控操作系统中对应安全车控操作系统的系统软件部分，主要有实时操作系统、硬件抽象层、基础软件、运行时环境和实时域功能服务组成。通过其上运行的各种检测任务针对获取到的各种数据进行监测判断以确定当前车辆是否处于安全状态，并在必要的时候，通过其上运行的安全控制应用实现对车辆的安全控制。

1.1.9 云控技术 cloud integration control technology, CICT

车路云一体化融合控制技术的简称，是车路云一体化融合控制系统通过车路融合感知、融合决策与控制，实现车辆行驶和交通运行安全与效率综合提升的技术，以下简称“云控”。

2.2 缩略语

下列缩略语适用于本文件。

ADAS 高级驾驶辅助系统 Advanced Driving Assistance System

AI 人工智能 Artificial Intelligence

API 应用程序编程接口 Application Programming Interface

ARA AUTOSAR 自适应应用运行环境 AUTOSAR Runtime for Adaptive Applications

ASIL 汽车安全集成等级 Automotive Safety Integration Level

AUTOSAR 汽车开放系统架构 AUTmotive Open System Architecture

CA 条件自动驾驶 Conditional Driving Automation

CNN 卷积神经网络 Convolutional Neural Network

CPU 中央处理器 Central Processing Unit

DDS 数据分发服务 Data Distribution Service

ECU 电子控制单元 Electronic Control Unit

EEA 电子电气架构 Electrical/Electronic Architecture

FA 全自动驾驶 Full Driving Automation

FIFO 先进先出 First Input First Output

FOTA 固件升级 Firmware Over-The-Air

GPL GNU 通用公共许可证 GNU General Public License

HA 高级自动驾驶 High Driving Automation

HMI 人机接口 Human Machine Interface

IDE 集成开发环境 Integrated Development Environment

IDL 接口定义语言 Interface Definition Language

IPC 工业控制计算机 Industrial PC

JasPar 日本汽车软件平台架构组织 Japan Automotive Software Platform Architecture

MCU 微控制单元 Microcontroller Unit

OEM 原始设备制造商 Original Equipment Manufacturer

OSEK 汽车电子开放式系统及接口规范 Open Systems and the corresponding interfaces for automotive Electronics

OTA 空中下载 Over the Air

POSIX 可移植操作系统接口 Portable Operating System Interface of UNIX

QoS 服务质量 Quality of Service

VDX 汽车分布式执行标准 Vehicle Distributed Executive

ROS 机器人操作系统 Robot Operating System

RTOS 实时操作系统 Real Time Operating System

SLAM 同步定位与建图 Simultaneous Localization and Mapping

SOA 面向服务的架构 Service-Oriented Architecture

TSN 时间敏感网络 Time Sensitive Network

V2X 车联网通信 Vehicle to Everything

VSLAM 视觉同步定位与地图构建 Visual SLAM

2 车控操作系统研究背景

汽车产业是国民经济的重要支柱产业，带动庞大的制造业上下游，代表国家工业水平，是“制造强国”重大战略部署的重要支撑和融合载体。当前，世界汽车产业正在经历一场以“电动化、智能化、网联化和共享化”为特征的“新四化”技术革命和行业变革，随着新一代能源技术变革、信息技术、人工智能、大数据等与汽车“新四化”的加速融合，以汽车为代表的运载工具正在成为全球技术变革和科技创

新的竞争制高点。发展智能网联汽车是解决汽车社会交通安全、道路拥堵、能源消耗、污染排放等问题的重要手段，也是构建智慧出行服务产业生态的核心要素和推进交通强国、数字中国、智慧城市的重要载体。

2020 年 11 月国务院印发《新能源汽车产业发展规划（2021－2035 年）》部署了 5 项战略任务，其中构建新型产业生态作为其中一项，鼓励以生态主导型企业为龙头，加快车用操作系统开发应用。同年发布的《智能网联汽车技术路线图 2.0》支撑构建了中国智能网联汽车产业技术发展体系，为产业技术发展指明了方向，对于中国智能网联汽车产业动态与趋势，出现的技术新特征与趋势进行研判。

车载智能计算基础平台是智能网联汽车最核心的新型增量零部件,是智能网联汽车产业互联网下的最基础平台，是兼顾市场亟需与国家监管的“软硬”一体的高科技产品。2019 年 5 月，在工信部指导下，赛迪研究院、装备中心和国汽智联等联合业内优势单位发布《车载智能计算基础平台参考架构 1.0》白皮书，涵盖网联、云控、数据通信、地图、信息安全等中国特色，初步形成车控操作系统中国共识。车控操作系统是运行在异构分布硬件架构上的实时安全平台软件，提供整车及部件感知、规划、控制等功能框架并向上支撑智能网联驾驶生态的软件集合，是车载智能计算基础平台安全、实时和高效运行的重要基础和核心支撑。

2019 年 10 月，汽标委发布了《车用操作系统标准体系》，规范了车用操作系统定义，划分了车用操作系统边界，明确了车用操作系统

分类（如下图 1 所示），构建了车用操作系统标准体系。《车用操作系统标准体系》的发布，为车控操作系统标准化工作的开展提供了指导方向。

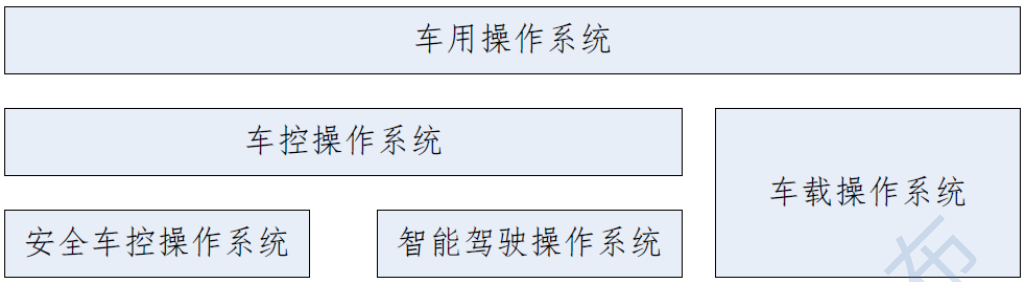

图 1 车用操作系统分类

车控操作系统分为安全车控操作系统和智能驾驶操作系统，其中，安全车控操作系统主要面向经典车辆控制领域，如动力系统、底盘系统和车身系统等，该类操作系统对实时性和安全性要求极高，生态发展已趋于成熟。智能驾驶操作系统主要面向智能驾驶领域，应用于智能驾驶域控制器，该类操作系统对安全性和可靠性要求较高，同时对性能和运算能力的要求也较高。该类操作系统目前在全世界范围内都处于研究发展的初期，生态尚未完备。

车载操作系统主要面向信息娱乐和智能座舱，主要应用于车机中控系统，对于安全性和可靠性的要求处于中等水平，该类操作系统发展迅速，依托于该类操作系统的生态也处于迅速发展时期。

本研究报告的涉及范围主要是车控操作系统。

2.1 车控操作系统需求分析

汽车电子电气架构（EEA，Electrical/Electronic Architecture）是集合了汽车的电子电气系统原理设计、中央电器盒设计、连接器设计、电子电气分配系统等设计为一体的整车电子电气解决方案。汽车电子

电气架构具体是在功能需求、设计要求和标准法规等特定约束下，通过对功能、性能、成本和装配等各方面进行分析，将动力总成、智能驾驶系统、信息娱乐系统等信息转化为实际的电源分配物理布局、信号网络、数据网络、诊断、电源管理等电子电气解决方案。

随着汽车智能化、网联化趋势的发展，汽车电子底层硬件不再是由单一芯片提供简单的逻辑计算，而是需要复杂的多核芯片提供更为复杂控制逻辑以及强大的算力支持。软件也不再是基于某一固定硬件开发，而是要具备可移植、可迭代和可扩展等特性。汽车智能化与网联化发展趋势共同推动了汽车电子电气架构的变革，一方面是车内网络拓扑的优化和实时、高速网络的启用，另一方面是电子控制单元（ECU）的功能进一步集成到域控制器甚至车载计算机。汽车电子电气架构的演进趋势为从分布式架构到域集中式架构最后到中央集中式架构转变，如图 2 所示。

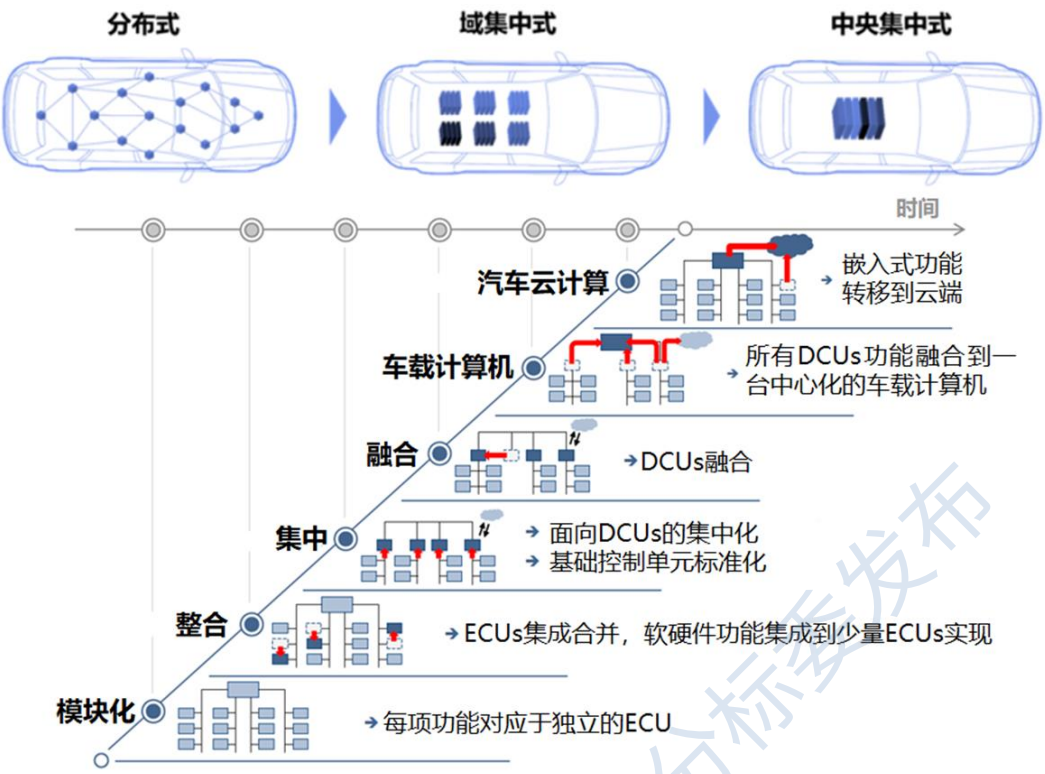

图 2 汽车电子电气架构演进趋势（来源于博世）

当前，集中式电子电气架构的主要类型有三域 EEA（如图 3 所示）和 Zonal EEA（如图 4 所示）。本研究报告主要是针对三域 EEA 架构类型。其中，三域主要是指车辆控制域、智能驾驶域和智能座舱域，车辆控制域是将原动力域、底盘域和车身域等经典车辆域进行了整合，负责整车控制，实时性安全性要求高；智能驾驶域负责自动驾驶相关感知、规划、决策相关功能的实现；智能座舱域负责 HMI 交互和智能座舱相关功能的实现。对于集中式电子电气架构，智能驾驶系统功能的实现在其中扮演重要地位，相对于分布式电子电气架构，集中式电子电气架构可以提供更为强大的算力，而强大的算力实现不仅需要硬件的支持，更需要和集中式电子电气架构适配的先进的车控操作系统。随着软件定义汽车、数字驱动设计趋势的发展，车控操作系统作为车载智能计算基础平台的核心技术，也将成为未来智能汽车

领域竞争的重要方向。

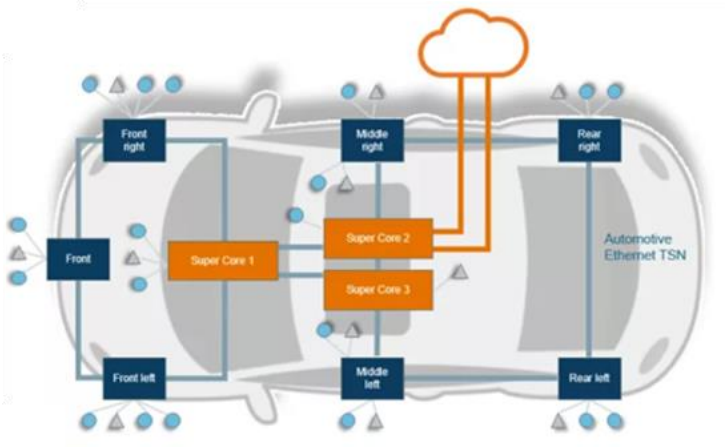

图 3 三域 EEA 示意图（来源于网络）

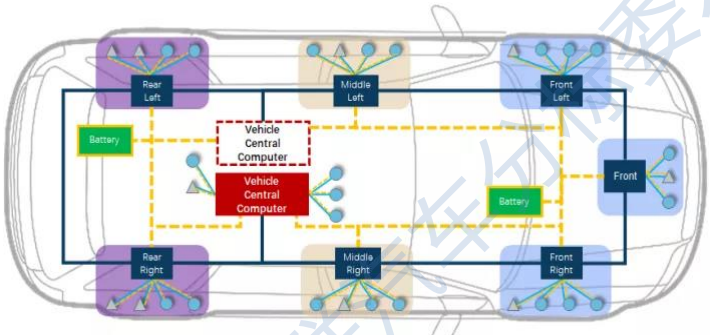

图 4 Zonal EEA 示意图（来源于网络）

2020 年 10 月，《节能与新能源汽车技术路线图 2.0》发布，其中提出了智能计算平台的技术发展路线（参考表 1），进一步从战略层面定位了计算平台及其操作系统在智能驾驶汽车发展中的重要作用。

表 1 计算平台技术发展路线

2025	2030	2035
计算平台支持 CA 级别自动驾驶。硬件平台实现核心模块的集成，系统软件实现全栈、完整化构建能力，功能软件实现框架完整构建能力，自动驾驶操作系统实现自主知识产权的突破，初步建立自主开发生态。		
计算平台支持全面网联和云控协同感知、决策、控制，数据收集与处理，支持 HA、FA 级自动驾驶。硬件平台实现计算平台核心模块异构分布架构，系统软件主要部分实现自主，功能软件在部分重要模块的实现与应用基础上实现自主可控，在自主可控能力之上建立功能软件级生态。		

计算平台具备和交通基础设施（车路云）全方位无缝协同的能力。架构方面实现车辆控制单元高度集成及异构融合方案，算法上支持强人工智能。硬件平台在芯片级集成基础上实现完全自主知识产权，系统软件与功能软件实现全面自主化，引领全行业的标准。计算平台实现定制与行业领先，建立自主可控的开发与应用生态。

车控操作系统包括安全车控操作系统和智能驾驶操作系统。安全车控操作系统主要是实时操作系统（RTOS），主要应用对象是 ECU。ECU 对安全车控操作系统最基本的要求是高实时性，系统需要在规定时间内完成资源分配、任务同步等指定动作。嵌入式实时操作系统具有高可靠性、实时性、交互性以及多路性的优势，系统响应极高，通常在毫秒或者微秒级别，满足了高实时性的要求。目前，主流的安全车控操作系统都兼容 OSEK/VDX 和 Classic AUTOSAR 这两类汽车电子软件标准。其中，Classic 平台基于 OSEK/VDX 标准，定义了安全车控操作系统的技术规范。

随着智能化、网联化技术的发展，智能汽车感知融合、决策规划和控制执行功能带来了更为复杂算法并产生大量的数据，需要更高的计算能力与数据通信能力。基于 OSEK/VDX 和 Classic AUTOSAR 软件架构的安全车控操作系统已经不能满足未来自动驾驶汽车的发展需求，AUTOSAR 组织为面向更复杂的域控制器和中央计算平台的集中式电子电气架构推出 Adaptive AUTOSAR 平台。Adaptive 平台定义采用了基于 POSIX 标准的操作系统，可以为支持 POSIX 标准的操作系统及不同的应用需求提供标准化的平台接口和应用服务，主要是为了适应汽车智能化的发展需求。Adaptive AUTOSAR 尚处于发展初期，其生态建设获得 Tier1、主机厂的普遍认可尚需时日。同时，由于智能网联汽车的区域属性及社会属性增加，在行驶过程中需要通信、地

图、数据平台等国家属性的支撑和安全管理，每个国家都有自己的使用标准规范，因此，Adaptive AUTOSAR 能否满足智能化的需求尚有待验证。

智能汽车的发展需要符合中国标准的车内电子电气架构、通信系统、智能终端、驾驶辅助和自动驾驶系统、云平台等新架构汽车产品标准，因此，在软件定义汽车的趋势下，网联、云控也成为中国智能网联汽车发展的特色，而作为智能网联汽车的核心车控操作系统，更应该满足“中国标准”方案的智能汽车电子电气架构。此外，车控操作系统的架构设计某种程度上决定能否建立良好的应用软件生态，从而建立基于软件、服务的商业模式。

综上所述，鉴于适配于集中式电子电气架构的车控操作系统软件架构尚在演化，以及“中国标准”方案的智能网联汽车的发展需求，研究并制定符合“中国方案”智能网联汽车发展的车控操作系统架构具有迫切需求和重要意义。

2.2 车控操作系统研究现状

2.2.1 安全车控操作系统

安全车控操作系统国外发展较早，目前已经开展了一系列的标准化工作，国内目前主要处于跟随状态。

欧洲在 20 世纪 90 年代发展出用于汽车电子上分布式实时控制系统的开放式系统标准 OSEK/VDX，主要包括 4 部分标准：1）操作系统规范（OS）；2）通信规范；3）网络管理规范；4）OSEK 实现语言。但随着技术、产品、客户需求等的升级，OSEK 标准逐渐不能支

持新的硬件平台。

2003 年，宝马、博世、大陆、戴姆勒、通用、福特、标志雪铁龙、丰田、大众 9 家企业作为核心成员，成立了一个汽车开放系统架构组织（简称 AUTOSAR 组织），致力于建立一个标准化平台，独立于硬件的分层软件架构，制定各种车辆应用接口规范和集成标准，为应用开发提供方法论层面的指导，以减少汽车软件设计的复杂度，提高汽车软件的灵活性和开发效率，以及在不同汽车平台的复用性。AUTOSAR 以 OSEK/VDX 为基础，但涉及的范围更广。

截至目前，AUTOSAR 组织已发布 Classic 和 Adaptive 两个平台规范，分别对应安全控制类和自动驾驶的高性能类。Classic 平台基于 OSEK/VDX 标准，定义了安全车控操作系统的技术规范。Classic AUTOSAR 的软件架构如图 5 所示，其主要特点是面向功能的架构（FOA），采用分层设计，实现应用层、基础软件层和硬件层的解耦。

图 5 Classic AUTOSAR 软件架构

AUTOSAR 标准平台由于采用开放式架构和纵向分层、横向模块化架构，不仅提高了开发效率、降低开发成本，同时保障了车辆的安全性与一致性。AUTOSAR 组织发展至今，得到了越来越多的行业认可，目前已有超过 180 家的车、零部件、软件、电子等领域的成员。AUTOSAR 目前已经成为国际主流的标准软件架构，基于 AUTOSAR 标准平台，拥有完整的汽车软件解决方案的企业主要有 Vector、KPIT、ETAS、DS 以及被大陆收购的 Elektrobit 和被西门子收购的 MentorGraphics。此外，宝马、沃尔沃等汽车厂商都相继推出了基于 AUTOSAR 标准平台的车型。

在日本，日本汽车软件平台架构组织（Japan Automotive Software Platform Architecture，JasPar）成立于 2004 年，旨在联合企业横向定制兼顾汽车软硬件的通信标准、实现车控操作系统的通用化，提高基础软件的再利用率等。JasPar 组织成员包括绝大多数的日系汽车及配套软硬件产品厂商。

我国主机厂及零配件供应商目前主要使用 Classic AUTOSAR 标准进行软件开发。一汽集团、长安集团等主机厂于 2009 年开始利用 Classic AUTOSAR 标准的工具进行 ECU 的设计、开发、验证。同时，上汽集团、一汽集团、长安集团、奇瑞集团等主机厂和部分高校成立了 CASA 联盟，旨在中国推广和发展 AUTOSAR 架构。目前江淮汽车也是主要基于 Classic AUTOSAR 标准进行软件和产品开发。

在产品方面，普华软件是中国电子科技集团的国产操作系统战略平台，并作为牵头单位承担了关于汽车电子操作系统的十一五、十二

五核高基重大专项，所形成的车控操作系统在车身控制模块（BCM）、新能源整车控制器（VCU/HCU）、电子转向系统（EPS）等关键零部件得到量产应用，并已被德国博世的先进辅助驾驶系统（ADAS）量产使用。

东软睿驰发布了 NeuSAR 产品，其基于 AUTOSAR 研发制作，为自主研发自动驾驶系统的 OEM 整车企业及零部件供应商提供的面向下一代汽车通讯和计算架构的系统平台，包含 AUTOSAR Classic、AUTOSAR Adaptive 及系列开发系统工具。

2.2.2 智能驾驶操作系统

智能驾驶操作系统将会成为自动驾驶汽车发展的核心竞争力之一。智能驾驶操作系统发展趋势和特点是纵向分层，以实现层与层之间的解耦，方便快速开发和移植，如图 6 所示。

图 6 智能驾驶操作系统纵向分层示意图

AUTOSAR 组织为应对自动驾驶技术的发展推出了 Adaptive AUTOSAR（AP）架构，如图 7 所示，其主要特点是采用面向服务的架构（SOA），服务可根据应用需求动态加载，可通过配置文件动态加载配置，并可进行单独更新，相对于 Classic AUTOSAR（CP），可

以满足更强大的算力需求，更安全，兼容性好，可进行敏捷开发。

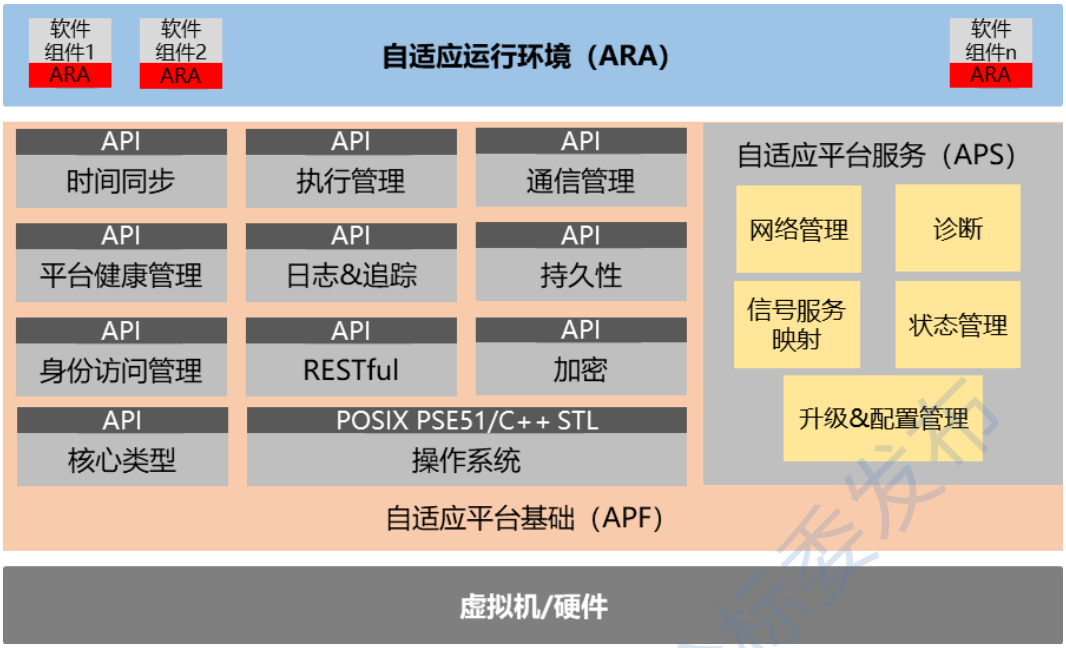

图 7 Adaptive AUTOSAR 架构

Adaptive AUTOSAR 系统主要适应于新的集中式的高性能计算平台，满足车内部件之间的高速通信需求和智能驾驶的高计算能力需求。AP 平台采用了服务化的架构，系统由一系列的服务组成，应用和其他软件模块可以根据需求调用其中的一个或者多个服务，而服务可以是平台提供的，也可以是远程其他部件提供，OEM 可以按照功能设计需求定义自己的服务组合。AP 设计也在 CP 成功的设计方法论的指导下进行开发，兼顾灵活和功能安全的确定性需求，同时可以进行整车功能的统一设计，兼顾安全车控和智能驾驶系统。AP 平台没有设计新的操作系统内核，所有符合 POSIX PSE51 接口的操作系统内核都可以使用，AP 平台重点是在操作系统内核之上的系统服务中间层，主要分为平台基础功能和平台服务功能两部分。平台基础功能更多的是本地服务，采用库的方式进行调用，如生命周期管理、通

信管理、存储管理、诊断功能等；平台服务不限制服务的提供实例的位置，采用互联网常用的服务类似的调用方式，如升级管理、网络管理、状态管理、传感器服务接口等。AP 平台主要的三个支撑和演进方向是：安全（包含信息安全和功能安全），连接（包括车内和车外各种新的通信机制），可升级（包含 OTA，灵活的软件设计和管理等）。AP 平台仍采用传统的标准设计方式，每年一个版本集中进行新的功能发布。

虽然 AUTOSAR AP 在继承 ATUOSAR CP 成功经验的基础上，采用了很多互联网的思路来设计，但是由于汽车软件需求变化较快，仍存在很多问题，如传统的 AUTOSAR 方法论配置繁琐，支持的通信速率低，不支持解耦部署策略和动态升级方案，特别是车车协同、车路协同、车云协同等还一直处于需求讨论阶段，没有支持的时间表。AUTOSAR AP 平台是适应新一代电子电气架构下的集中式计算需求而产生的，但只是整车功能中的一小部分，缺乏从整车电子电气系统视角考虑信息安全、功能安全、通信等需求。

在底层操作系统之上，软件中间件在智能驾驶领域也备受关注。中间件的主要目标是为上层应用提供数据通信、协议对齐、计算调度、模块化封装等常用功能，为应用开发提供标准化、模块化的开发框架，实现模块解耦和代码复用。ROS 作为最早开源的机器人软件中间件，很早就被机器人行业使用，很多知名的机器人开源库，比如基于 quaternion 的坐标转换、3D 点云处理驱动、定位算法 SLAM 等都是开源贡献者基于 ROS 开发的。ROS 的首要设计目标是在机器人研发

领域提高代码复用率。ROS 是一个分布式的进程（也就是“节点”）框架，这些进程被封装在易于被分享和发布的程序包和功能包中。整个智能驾驶系统和机器人系统有很强的相似度，ROS 的开源特性，丰富的开源库和工具链，特别在智能驾驶的研究领域有着较为广泛的应用，很多自动驾驶的原型系统中都能够看到 ROS 的身影，例如 AUTOWARE，百度 Apollo 最初也是使用了 ROS，直至 Apollo 3.5 版本才切换至自研的车载中间件 Cyber RT。

ROS 在发展过程中主要有两个版本，ROS1 和 ROS2，ROS1 的通信依赖中心节点的处理，无法解决单点失败等可靠性问题。为了更好的符合工业级的运行标准，ROS2 最大的改变是，取消 Master 中央节点，实现节点的分布式发现，发布/订阅，请求/响应；底层基于 DDS（数据分发服务）这个工业级的通信中间件通信机制，支持多操作系统，包括 Linux、windows、Mac、RTOS 等。

虽然 ROS2 基于 ROS1 有了很大的改进，但是距离完全车规应用还有很大的距离，有些公司如 APEX.AI 也在对 ROS 进行车规级的改造尝试。

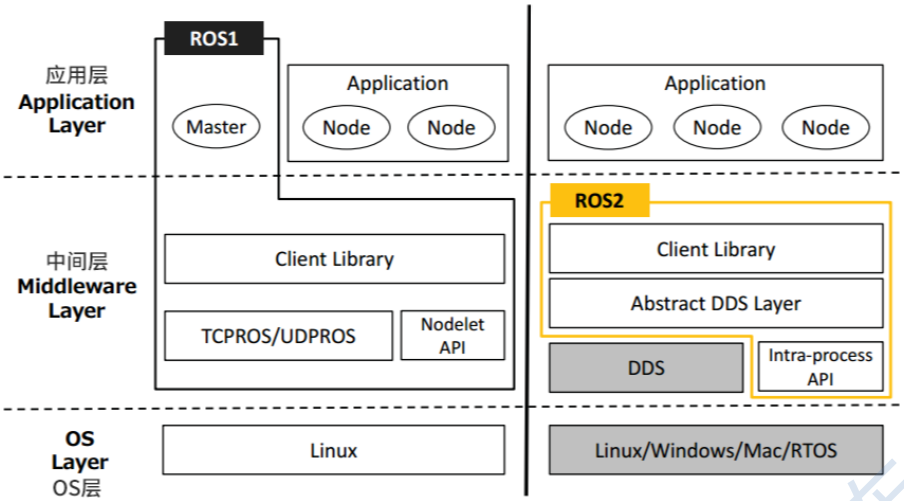

图 8 ROS 架构

目前普遍采用的车控操作系统底层内核主要有 Linux、QNX 和其他 RTOS（如 FreeRTOS、ThreadX、VxWorks 等），三者之间的主要特点对比如表 2 所示。

Linux 最初是作为通用操作系统而设计开发的，但提供了一些实时处理支持，这包括大部分 POSIX 标准中的实时功能，支持多任务、多线程，具有丰富的通信机制等。除此之外，Linux 社区有实时性增强 patch，在 Linux 内核原有 RT 功能上，增加了中断线程化、优先级默认继承等功能。Linux 也提供了符合 POSIX 标准的调度策略，包括 FIFO 调度策略、时间片轮转调度策略和静态优先级抢占式调度策略。另外，Linux 还提供了内存锁定功能，以避免在实时处理中存储页面被换出，同时提供了符合 POSIX 标准的实时信号机制。

QNX 是一种商用的遵从 POSIX 规范的类 Unix 实时操作系统，其主要特点是符合分布式、嵌入式、可规模扩展的硬实时操作系统。QNX 遵循 POSIX.1 (程序接口）和 POSIX.2 (Shell 和工具)、部分遵循 POSIX.1b（实时扩展）。QNX 的微内核结构是它区别于其它操作系统

的显著特点。QNX 的微内核结构，内核独立自处于一个被保护的地址空间；驱动程序、网络协议和应用程序处于程序空间中。

表 2 车用操作系统内核比较

项目指标	Linux	QNX	其他 RTOS
实时性能	需要进行实时性改造	微秒级延时	微秒级延时
开放性	源代码开放	封闭	商用或开放
许可协议	GPL	商用	N/A
费用	无授权费用（商用收费）	Royalty&License	较低或免费
功能安全	ASIL B 有可能	ASIL D	N/A
软件生态	应用生态链完善	汽车领域应用广泛	有限
优势	技术中立，支撑复杂功能	性能强，安全性高	实时性好，启动快
劣势	系统复杂	进程间通信、系统调用开销等	进程间通信、系统调用开销
主要适用范围	智能座舱、信息娱乐、TBOX、ADAS、某些域控制器等	仪表盘、智能座舱、信息娱乐、导航、ADAS、域控制器等	仪表盘、ADAS、整车控制器等

随着自动驾驶技术的快速发展，汽车对软件特别是操作系统的变革需求越来越高，主机厂、Tier1 供应商和自动驾驶软硬件技术方案提供商纷纷投入大量的人力、物力和财力进行车控操作系统的研发，希望在软件定义汽车的时代能够占据一席之地。下面将对目前国内外主流车控操作系统的开发和应用情况进行简单介绍。

（1）特斯拉 Autopilot 自动驾驶软件架构

众所周知，特斯拉是自动驾驶技术和产业化的领跑者，其优势在于以计算平台为核心，自研并领先芯片硬件、操作系统平台软件等。特斯拉自动驾驶软件架构如图 11 所示，主要特点是其操作系统基于单一 Linux 内核，打造了整套自动驾驶的软件方案，分别完成了从感知、决策规划和控制系统解决方案。从现在公开的信息可知，系统基

于 Ubuntu 进行裁剪，对 Linux 内核进行了实时性改造，这个内核也开源在 github 上，深度学习框架基于 PyTorch，实时数据处理基于开源流处理平台 Kafka，拥有 48 个独立的神经网络进行多维度数据处理，并且具备强大的 OTA 升级能力。其 FSD（Full Self-Driving）计算平台硬件集成了智能座舱域和自动驾驶域，操作系统通过 OTA 软件升级，充分利用数据、云计算生态，开创汽车产品价值和服务的新模式。

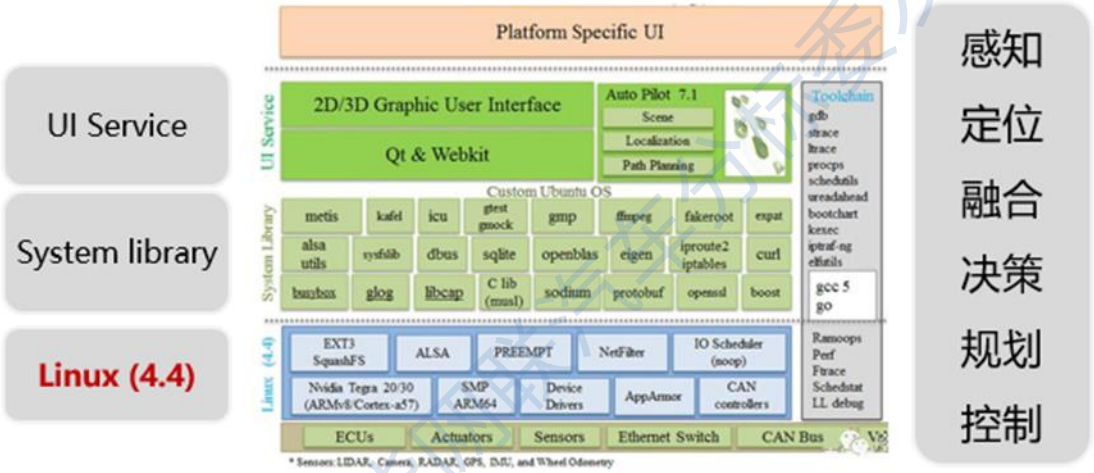

图 9 特斯拉 Autopilot 软件架构（来源于网络）

Autopilot 与娱乐控制层掌控了所有的摄像头和雷达传感器。在模块内部，Autopilot 系统和娱乐系统这两大部分通过 CAN 和高速串行总线 FPD-Link 打通，两者之间甚至可以传递视频数据。Autopilot 数据流处理机制如图 10 所示，融合自动驾驶与车内感知，实现服务驾驶闭环。

图 10 Tesla Autopilot 数据流处理机制

通过 E/E 架构的集中化，特斯拉将汽车的软件开发内化，将汽车底层硬件标准化和抽象化，此举让特斯拉通过软件定义汽车和创新变得更容易。特斯拉的 Autopilot 进化沿着功能集中化、资源共享化的道路前进，体现了特斯拉软硬件解耦，通过软件定义汽车的实践。

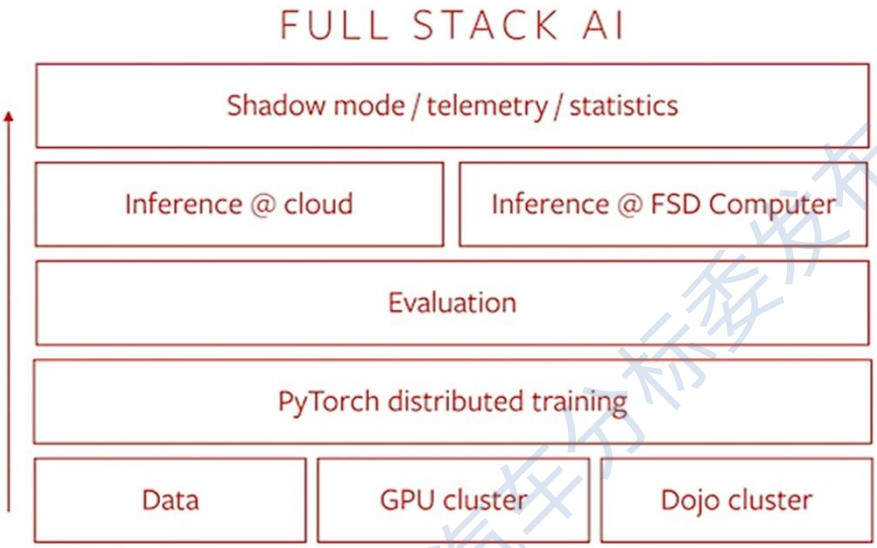

图 11 特斯拉全栈 AI 架构（来源于网络）

特斯拉在打造 Autopilot 整体软件栈时采用的理念（参见图 11）：站在巨人肩膀上进行创新，充分利用开源项目进行全栈开发，从开发（比如 UI 框架基于 QT，前端也基于一些开源的库）、构建、到部署都采用了开源的方案，推动车端算力服务、调度机制，以及云端算力资源优化布局，培育智能驾驶新业务模式。

最底层的是数据、GPU 集群以及 Dojo 计算集群，这一层主要进行数据采集、标注和训练，生成算法模型；采用基于 PyTorch 开源框架的深度神经网络对模型进行分布式训练；用损失函数对模型进行评估；在评估层之上，是云端推理和车端 FSD 芯片推理，到这一层，意味着算法模型走完了大部分流程，然后就是部署到车端；在车端，这

是车控操作系统运行范围，特斯拉通过影子模式将这些算法模型与人类驾驶行为进行比对，检测是否存在异常。

这样从数据采集到算法部署的闭环，随着更多汽车上路展开数据收集，可实现基于海量数据的驱动，让系统性能不断迭代，更加优秀。在这个闭环当中，涉及到数据集、模型训练神经网络、云端和车端推理算法等要素。

总结来看，软件定义汽车深刻地改变了汽车行业的盈利模式，将高性能的硬件预埋作为投资，通过软件更新服务盈利，已经成为特斯拉为代表的新造车势力的标准操作。

（2）大众中央集中式软件参考架构

大众汽车为了加速自动驾驶技术的应用，组建了庞大队伍自主开发汽车操作系统 vw.OS。vw.OS 采用的是基于 Adaptive AUTOSAR 面向服务的软件架构，其中，中央集中式软件参考架构如图 12 所示。大众新一代 EE 架构的设计特点主要有：1）采用高性能处理器、高速网络；2）兼容 POSIX 的内核（Linux/QNX 等）+ Adaptive AUTOSAR 操作系统；3）应用软件和 I/O 功能解耦，减少整个系统的复杂性和应用之间的依赖性；4）高效、快速地开发用户功能；5）采用面向服务的通信。

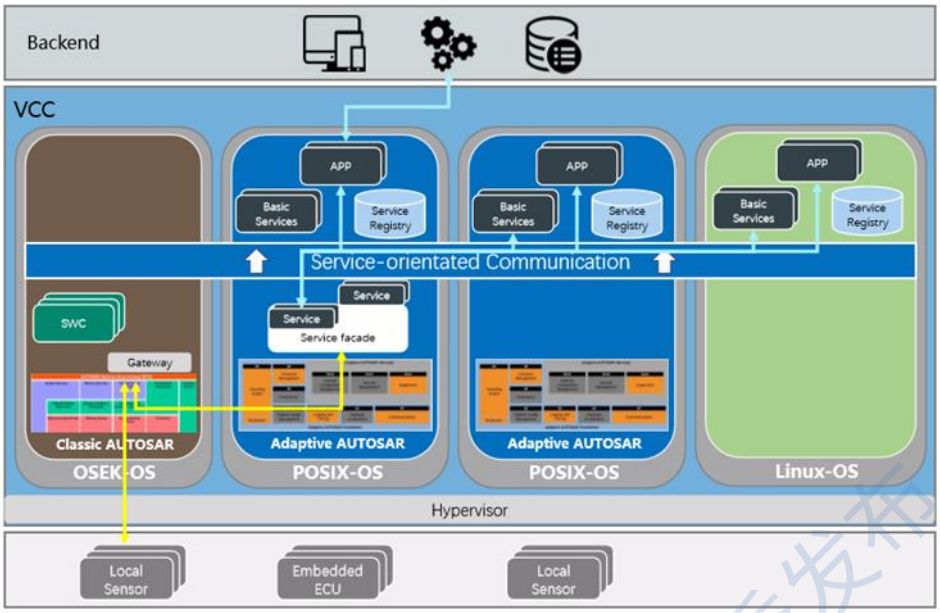

图 12 大众中央集中式软件参考架构（来源于网络）

（3）华为 MDC 智能驾驶计算平台架构

华为 MDC（Mobile Data Center：移动数据中心）定位为智能驾驶的计算平台。此平台集成华为在 ICT 领域 30 多年的研发与生产制造经验，基于 CPU 与 AI 处理器芯片，搭载智能驾驶 OS，兼容 AUTOSAR，支持 L2~L5 平滑演进，结合配套的完善工具链，客户或生态合作伙伴可灵活快速的开发出针对不同应用场景的智能驾驶应用。华为 MDC 智能驾驶计算平台（以下简称华为 MDC 平台），性能强劲、安全可靠，是实现智能驾驶全景感知、地图&传感器融合定位、决策、规划、控制等功能的汽车“大脑”。适用于乘用车（如拥堵跟车、高速巡航、自动代客泊车、RoboTaxi）、商用车（如港口货运、干线物流）与作业车（如矿卡、清洁车、无人配送）等多种应用场景。华为的 MDC 智能驾驶计算平台架构主要特点有：1）提供软硬件解决方案，且高度解耦，可独立升级，硬件升级路线和软件升级路线分别独立；2）对主流传感器的适配性好，支持主流 GNSS、IMU、摄像

头、激光雷达和毫米波雷达等传感器的数据接入，且支持摄像头和激光雷达点云的前融合；3）对主流中间层软件的适配性很好，可兼容 ROS 和 AUTOSAR，支持 Caffe 和 TensorFlow 等常用深度学习框架；4）核心组件（芯片、操作系统内核）自主可控；5）华为是业界唯一同时拥有 CPU 与 AI 芯片研发能力的厂家，MDC 平台硬件集成具有 CPU 与 AI 计算能力的强大 SoC 芯片，为智能驾驶提供可扩展的异构算力；6）功能软件基于 SOA 架构，遵循 AUTOSAR 规范，定义了智能驾驶基本算法组件（如感知算法组件、融合算法组件、定位算法组件、决策算法组件、规划算法组件、控制算法组件等）的调用框架与组件之间的软件接口；上层场景应用可以灵活选择不同的算法组件组合，实现具体的场景应用功能；7）提供安全可信，高效便捷，灵活开放的应用开发端到端工具集，支持可视化&拖拽式操作及自动代码生成，可一站式开发、测试、调优，帮助客户或生态合作伙伴快速开发满足 AUTOSAR 规范的智能驾驶应用。（4）英伟达自动驾驶平台架构

英伟达（NVIDIA）是全球领先的人工智能计算公司，利用其先进的硬件芯片开发优势，以行业较领先的高性能安全芯片为核心，提供完整的硬件平台和基础软件平台，其架构如图 13 所示。NVIDIA 计算平台硬件目前处在“Xavier”阶段，下一代平台“Orin”已发布但并未上市。Xavier 是 NVIDIA 首次生产的车规级系统级芯片，该芯片采用了六种不同类型的处理器，包括 CPU、GPU、深度学习加速器(DLA)、可编程视觉加速器 (PVA)、图像信号处理器(ISP) 和立体/光流加速器。基于 Xavier 芯片，NVIDIA 提供面向自动驾驶开发的

DRIVE AGX Xavier™，算力达到 30 TOPS，面向 L2+和 L3 级自动驾驶；提供 DRIVE AGX Pegasus™使用两块 Xavier 系统级芯片和两块 Turing GPU，算力达到 320 TOPS，面向 L4 级和 L5 级自动驾驶。NVIDIA Drive 的主要特点有：1）软硬件解决方案，且高度解耦，可独立升级，硬件升级路线和软件升级路线分别独立；2）硬件优势明显，是 GPU 设计、生产领域的领导者；3）软件生态非常好，有业界最完善的官方开发套件，开发者社区相对完善；4）软件层面开放程度较高，可在 DriveWorks（功能软件层）开放 API，也可在 Drive AV 和 Drive IX（应用软件层）开放 API；5）系统软件层融合了第三方 RTOS+AUTOSAR，设有 Hypervisor 层，第三方量产 RTOS 方案通过 ASIL D 认证；6）算法加速全部基于自身 CUDA 架构和 TensorRT 加速包，二者是 NVIDIA 独有，因此其软件开发生态不可脱离其硬件平台。

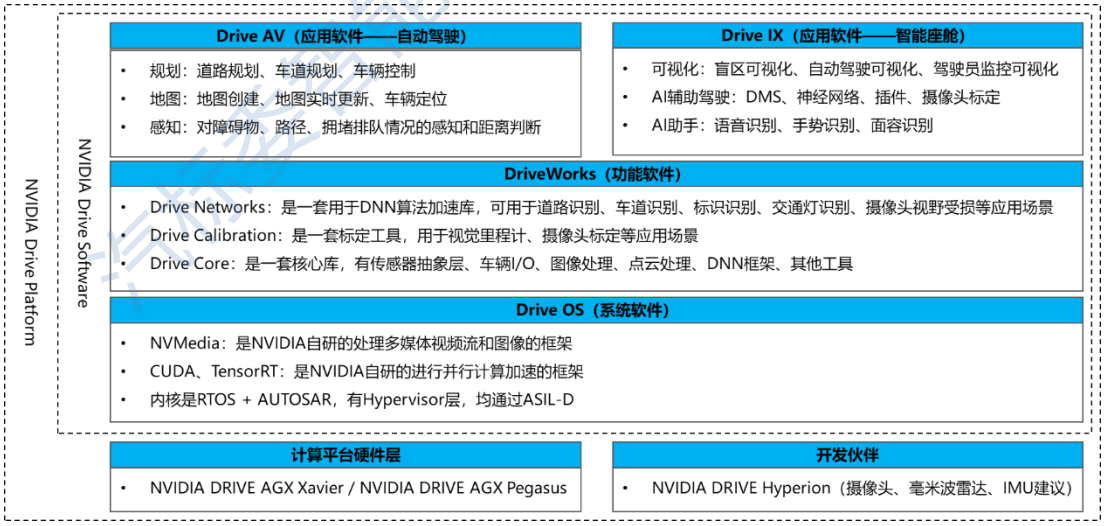

图 13 英伟达自动驾驶平台架构

（5）百度 Apollo 开放平台架构

百度 Apollo 是一套软件平台，其依赖的计算平台硬件需要采用

第三方的 IPC，Apollo 开放平台架构如图 14 所示。百度自行研发了两款辅助性硬件 ASU（Apollo 传感器单元）和 AXU（Apollo 扩展单元），其中，ASU 用于收集各传感器的数据，通过 PCIe 传输至 IPC，此外，IPC 对车辆的控制指令也需通过 ASU 向 CAN 发送；AXU 用于满足额外算力、存储的需求，以 GPU、FPGA 形式接入已有硬件平台。百度 Apollo 的主要特点有：1）为网联云控（V2X）进行软硬件端到端的开发；2）提出“认证平台”的概念，包括车辆认证、硬件认证；3）很好地融入了云服务，其中包括众多百度自家的其他产品，如：基础百度云服务、在线仿真产品、高精度地图、小度助手（Duer OS），各产品间彼此受益；4）由于开源，核心的算法模块在 Github 进行长时间优化后已充分产品化；5）主要侧重系统软件的开发，包含定制优化的操作系统、系统中间件及算法功能模块，大部分硬件则采用第三方方案；6）产品没有涉及到 AUTOSAR 架构的额外开发适配，也无需对车辆现有的 ECU/MCU 进行改变。

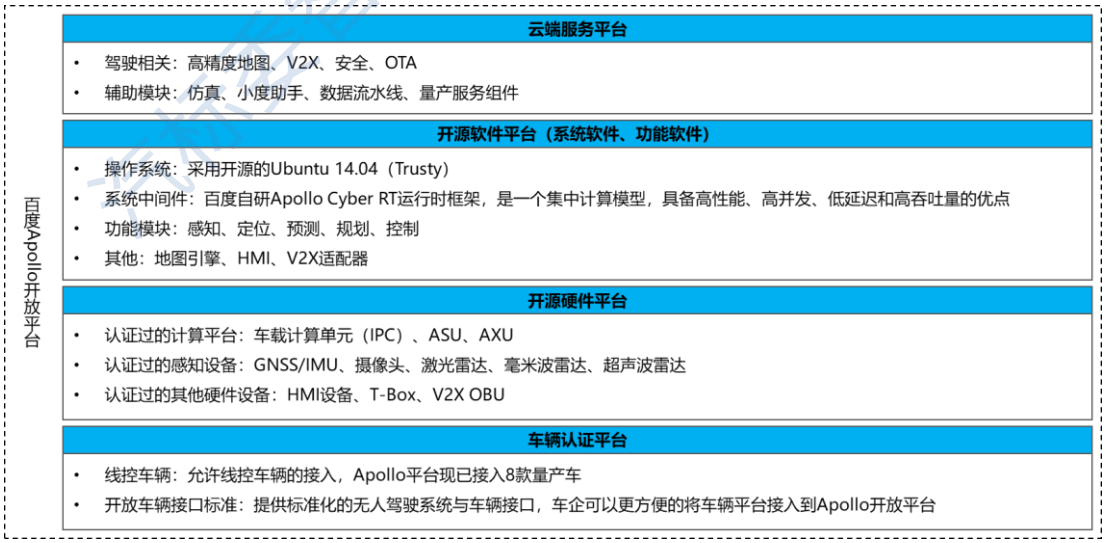

图 14 百度 Apollo 开放平台架构

2.3 车控操作系统架构研究目的及范围

2.3.1 车控操作系统架构研究目的

（1）梳理车控操作系统的国内外发展现状，明确汽车智能化、网联化发展趋势下的车控操作系统总体架构需求；

（2）在保证车控操作系统能够参考或兼容现有 AUTOSAR 架构的同时，又能够基于智能网联汽车的需求特点进行创新，并满足中国智能网联汽车发展的区域属性和社会属性；

（3）研究符合“中国方案”智能网联汽车发展的车控操作系统架构，总结车控操作系统技术发展路线，输出车控操作系统架构标准化建议。

2.3.2 车控操作系统架构研究范围

车控操作系统将安全车控操作系统与智能驾驶操作系统纳入整体研究范围，架构层面包含系统软件和功能软件框架。车控操作系统架构的研究主要涉及车控操作系统的总体参考架构，主要包括系统软件、功能软件、安全体系和工具链。通过对系统软件、功能软件、安全体系和工具链各模块的深入研究分析，形成车控操作系统系统软件架构和功能软件架构。车控操作系统架构应满足并引领未来智能网联汽车发展趋势，为车控操作系统的产品研发提供有效参考和指导。

3 车控操作系统总体架构

软件或计算机系统的软件架构是该系统的一个（或多个）结构，而结构由软件元素、元素的外部可见属性及它们之间的关系组成。从技术角度看，软件架构在软件开发过程中重要性主要有：1）架构明

确了系统实现的约束条件，架构是架构设计师对系统实现的各方面进行权衡的结果，是总体设计的体现，因此，在具体实现时必须按架构的设计进行；2）架构为维护的决策提供根据，在架构层次上能为日后的更改决策提供推理、判断的依据；3）在较高层面上实现软件复用，软件架构作为系统的抽象模型，可以在多个系统间传递（复用），特别是比较容易地应用到具有相似质量属性和功能需求的系统中。在软件定义汽车时代，车控操作系统是自动驾驶汽车的核心技术，其架构作为软件架构的一种，设计搭建高质量的车控操作系统架构对自动驾驶汽车的快速发展具有重要影响。

高度变化的需求、智能化的持续演进、车载硬件和软件系统复杂程度的提升对车控操作系统的性能、可扩展性、易用性、系统可靠性提出了严峻的挑战。车控操作系统运行基础是异构、分布式计算平台，既具有安全车控操作系统的功能和特点，还能够提供高性能、高可靠的传感器、分布式通信、自动驾驶通用框架等模块以支持自动驾驶感知、规划、决策、控制与执行的共性实现。车控操作系统总体架构与电子电气架构的革新和对应的车载计算架构相辅相成，因此从整个驾驶闭环角度，将安全车控操作系统与智能驾驶操作系统进行整体架构研究。

车控操作系统是自动驾驶相关功能开发的共性及核心软件平台。其中安全车控操作系统是指在经典车控 ECU 的主控 MCU 芯片上装载运行的嵌入式实时操作系统，其架构可参考、兼容或直接采纳 Classic AUTOSAR 软件架构，吸收其模块化和分层的思想；智能驾驶

操作系统架构可参考或兼容 Adaptive AUTOSAR，并进行扩展。

车控操作系统采用纵向分层、横向分区式架构，并在逻辑层次上包含系统软件和功能软件框架，是车载智能计算基础平台安全、实时、高效的核心和基础。系统软件创建复杂嵌入式系统运行环境，可以实现与 Classic 和 Adaptive 两个平台的兼容和交互。功能软件根据中国智能网联汽车应用特点，以及各类辅助驾驶/自动驾驶功能的核心共性需求，明确定义和实现各共性子模块，并进行通用模块定义和实现。

在参考并吸收国内外主流车控操作系统架构的基础上，提出基于“中国方案”的车控操作系统参考架构，如图 15 所示，其中，红色框线部分代表车控操作系统，蓝色框线部分代表车载智能计算基础平台，包含车控操作系统和分布式异构硬件平台。

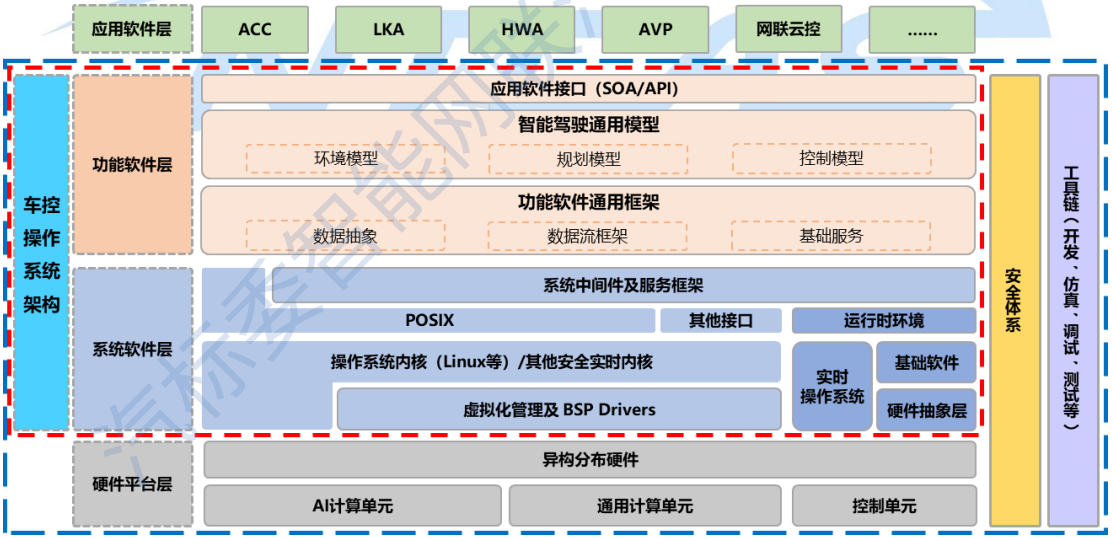

图 15 车控操作系统参考架构

车控操作系统架构（图 15）的创新点体现在既能够兼容 Classic AUTOSAR 平台，可基于 Classic AUTOSAR 平台进行扩展，也能够跟 Adaptive AUTOSAR 平台适配，又在业务机制上有所创新。

3.1 系统软件

车控操作系统系统软件是针对汽车场景定制的复杂大规模嵌入式系统运行环境。系统软件一般包含操作系统内核、虚拟化管理（如Hypervisor）、POSIX、系统中间件及服务等。

1）操作系统内核

车控操作系统内核要求与 Classic AUTOSAR 和 Adaptive AUTOSAR 对内核的要求类似。车载智能计算基础平台支持异构芯片，需考虑功能安全、实时性能要求。当前异构分布硬件架构各单元所加载的内核系统安全等级有所不同，AI 单元内核系统 QM ~ ASIL-B，计算单元内核系统 QM ~ ASIL-D，控制单元内核系统 ASIL-D，因而出现不同安全等级的多内核设计或单内核支持不同安全等级应用的设计。保证差异化功能安全要求的同时满足性能要求，是车控操作系统系统软件设计的关键。

目前应用在汽车或嵌入式系统中的 RTOS，如 OSEK OS、FreeRTOS、ThreadX、MicroITRON、INTEGRITY、QNX、VxWorks 等，从技术维度可以作为计算单元内核的选择，满足自动驾驶不同功能安全等级。其中，QNX 是目前广泛应用的汽车嵌入式 RTOS 内核系统，其建立在微内核和完全地址空间保护基础之上，硬实时、稳定、可靠、安全，满足 ASIL-D 功能安全等级。

另外，车载智能计算基础平台的复杂性也要求内核对功能软件及应用软件的库支持和高度可编程性。Linux 内核紧凑高效，开源灵活，广泛支持芯片和硬件环境及应用层程序。目前也有对 Linux 系统进行

定制优化的技术探索，可以实现部分 CPU 和内存资源保护并高效实时的混合系统，进行功能安全增强，以期望达到功能安全等级要求。

2）虚拟化管理

车控操作系统是基于异构分布硬件，应用程序如 AI 计算和实时安全功能可能分别依赖不同的内核环境和驱动，但在物理层面共享 CPU 等，这就要求通过虚拟化管理实现有效的资源整合和隔离。虚拟化管理技术是实现跨平台应用、提高硬件利用率的重要途径，Hypervisor 是一种硬件虚拟化技术，管理并虚拟化硬件资源（如 CPU、内存和外围设备等），提供给运行在 Hypervisor 之上的多个内核系统。

3）POSIX

系统软件可借鉴 Adaptive AUTOSAR 平台思想，采用 POSIX API。POSIX 能够很好地适应自动驾驶所需要的高性能计算和高带宽通信等需求。Adaptive AUTOSAR 采用基于 POSIX 标准的内核系统，可使用所有标准的 POSIX API，旨在满足未来高级自动驾驶的需求。车控操作系统系统软件基于实时嵌入式软件单元架构，可借鉴 Adaptive AUTOSAR 平台思想，在不同内核系统采用 POSIX API 与应用软件、功能软件交互。

4）系统中间件及服务

系统中间件及服务主要用来为上层应用提供基础的系统服务，例如分布式通信服务等。其中一种有代表性的分布式通信中间件技术规范即 DDS。车控操作系统需可建立跨多内核、多 CPU、多板的通用、高速、高效的 DDS 机制。DDS 可采用发布/订阅架构，强调以数据为

中心，提供丰富的 QoS（服务质量）策略，能保障数据进行实时、高效、灵活地分发，可满足各种分布式实时通信应用需求。

5）实时安全域

实时安全域操作系统是系统软件层上运行在 MCU 上的安全车控操作系统。主要包含硬件抽象层、基础软件、实时操作系统内核和运行时环境等模块，最基本的要求是高实时性，系统具备硬实时特性，需要在规定时间内完成资源分配、任务并发、同步等指定动作，可参考 Classic AUTOSAR 软件架构。

3.2 功能软件

车控操作系统功能软件主要包含自动驾驶系统的核心共性功能模块。核心共性功能模块包括智能驾驶通用模型和功能软件通用框架，结合系统软件，共同构成完整的车控操作系统，支撑驾驶自动化功能实现。具体描述如下：

1）智能驾驶通用模型

智能驾驶通用模型是对智能驾驶中智能认知、智能决策和智能控制等过程的模型化抽象，主要分为环境模型、规划模型和控制模型三部分。

2）功能软件通用框架

功能软件通用框架是承载智能驾驶通用模型的基础，分为数据流框架和基础服务两部分。数据流框架向下封装不同的智能驾驶系统软件和中间件服务，向智能驾驶通用模型中的算法提供与底层系统软件解耦的算法框架，基础服务是功能软件层共用的基本服务，其主要服

务于智能驾驶通用模型或功能应用，但其本身不局限于智能驾驶。

3）数据抽象

数据抽象通过对传感器、执行器、自车状态、地图以及来自云端的接口等数据进行标准化处理，为上层的智能驾驶通用模型提供各种不同的数据源建立异构硬件和驾驶环境的数据抽象，达到功能和应用开发与底层硬件的解耦和依赖。

4 车控操作系统系统软件架构

系统软件是车控操作系统中支撑自动驾驶功能实现的复杂大规模嵌入式系统运行环境。根据图 15 车控操作系统参考架构，结合系统软件的主要核心功能，形成车控操作系统系统软件架构（参见图 16）。

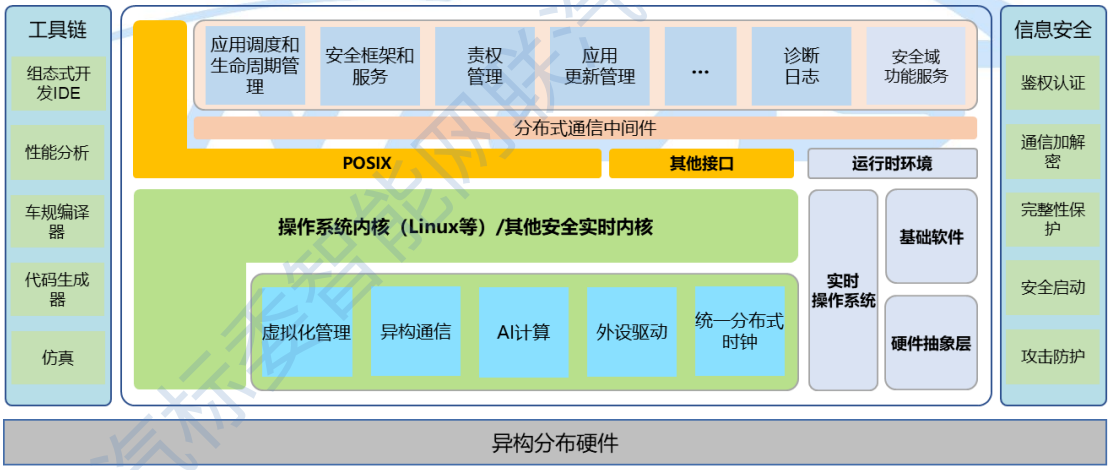

图 16 系统软件架构

4.1 操作系统内核

操作系统内核（如 Linux、QNX 、VxWorks、RT-Linux、鸿蒙内核等）主要提供操作系统最基本的功能，主要包含线程调度、地址空间、进程间通信、中断管理等系统必须功能的最小实现，决定着系统的性能和稳定性。可调度单元可以采用时间驱动或事件驱动、实施优

先级以及时间片轮转等多种调度方式，保证实时性，同时也需要兼顾公平性。进程间通信可以采用同步事件通知机制和消息传递机制来完成。事件通知机制主要用于传递信号，如计数型信号量，包括触发事件、等待时间以及取消事件等；消息传递机制用于任务间传递消息，包括普通的数据及能力权限，主要有发送消息，接收消息和取消消息等操作。内核通过实时调度算法来减少任务上下文的切换时间，从而满足任务的时限要求。

4.2 虚拟化管理及板级支持包

4.2.1 虚拟化管理

虚拟化就是通过某种方式隐藏底层物理硬件的过程，从而实现多个操作系统可以透明地使用和共享硬件。在典型的分层架构中，提供平台虚拟化的层称为 Hypervisor（也可称为虚拟机管理程序或 VMM）。常用硬件虚拟化的简单分层架构参考图 17。

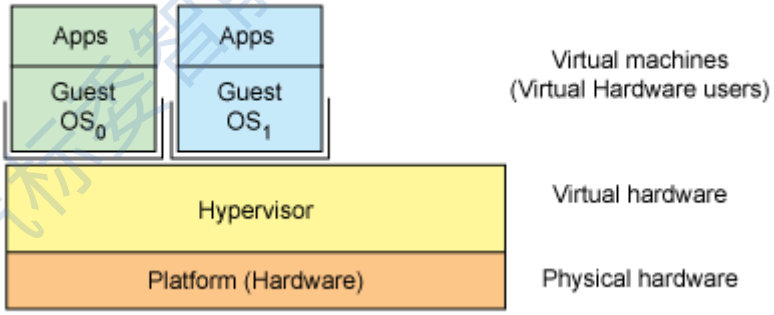

图 17 常用硬件虚拟化的简单分层架构（来源于网络）

车载领域的 Hypervisor 需要支持强隔离、设备复用和配置定制化。具体负责管理并虚拟化异构硬件资源，以提供给运行在 Hypervisor 之上的多个操作系统内核。Hypervisor 支持异构硬件单元（包括控制单元、计算单元、AI 单元）的隔离，在同一个异构硬件平台上支持不同

的操作系统内核，从而支持不同种类的应用。Hypervisor 是实现跨平台应用、提高硬件利用率的重要途径。

Hypervisor 需要具备高安全性、高实时性和高可靠性，实现硬件 CPU、内存、外设等资源在不同操作系统之间的隔离。由于服务之间的依赖性，任何一个部分的异常都可能会影响到系统的其他部分，Hypervisor 应定义明确的服务边界，有效地隔离故障，构建系统的容错能力和异常处理能力。多系统之间的高效通信机制也是 Hypervisor 的关键技术之一，操作系统间可以建立虚拟以太网、共享内存和快速消息等通信机制。操作系统可以根据不同通信需求来选择合适的通信方式。

4.2.2 异构通信

异构通信为分布式通信中间件提供统一的通信协议栈，使不同硬件、不同语言和不同操作系统的通信通道、接口使用相同的协议 API。异构通信是指部署在车载网络上，同时考虑通信可靠性、安全性的数据通信机制。其应能满足汽车电控系统的日益复杂，以及汽车内部控制功能电控单元相互之间通信需求的日益增长的趋势。

智能汽车异构通信协议的发展趋势是车载时间敏感网络（TSN）。时间敏感网络（TSN）基于标准以太网，在数据链路层提供一整套保障网络 QoS 的机制，例如时钟同步、门控队列、帧抢占、报文复制、丢弃等。随着 TSN 向汽车行业不断渗透，与汽车相关的国内外标准组织或者产业联盟纷纷着手制定相关标准。

1）时间同步

TSN 中的时间同步过程是从一个中央时间源(Grand Master)直接通过网络向所有网络设备和终端设备发送同步时间报文，然后通过 IEEE1588v2 协议，计算得到各个设备的时间与中央时间源的偏差，并存储在设备中，作为其报文发送的时间参考。此外，考虑到自动驾驶需要多传感器数据融合，为了提升检测精度，一般需要各个传感器与网络设备、控制器等同步，可靠性是车载系统的第一要素，一个 GM（Grand Master）发送两份相同的同步报文至终端(N)和交换机（B），甚至可以采用 2 个 GM 发送 4 分同样的同步报文。

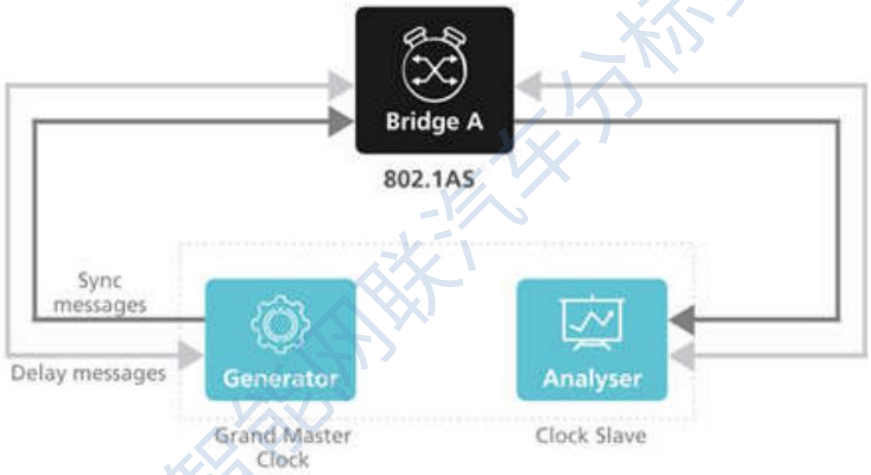

图 18 TSN 时间同步基本流程

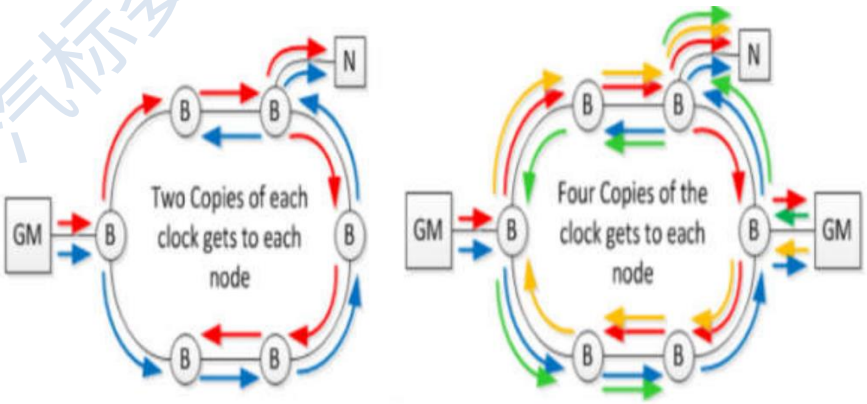

图 19 TSN 时间同步多发机制

2）流量整形

方案	标准	原理	评论
CBS-Credit-Based Shaper	IEEE802.1Qav	当队列中没有报文传输时，且其所在队列的Credit的值大于等于0，可以开始传输AVB报文，同时Credit值以sendslope速率减小； 上述报文发送完毕，且队列中没有等待传输报文，当Credit值为负数时，Credit值以idleSlope速率累加,直到Credit值为0； 当Credit值以idleSlope增加的过程中，可以传输普通以太网报文（如Best Efforts）,在此过程中，若credit值大于0，且Best Efforts流量没有完成传输，则继续传输。在这种情况下Credit的值不再以0为上界	CBS机制是AVB的核心，在车载通信领域已经有初步应用,可满足娱乐系统的音视频传输,由于其技术成熟,配置简单,当前国外厂商（如Volvo等)研究将其用于车载控制信号的传输的可行性
TAS-Time-Aware Shaper	IEEE 802.1Qbv	基于门控的思想，且需严格的时间同步。门有“开”、“关”两个状态，通过预先规划，计算得到每条流对应的门的开关时间，终端与网络设备严格遵守发包时刻，并配合帧抢占机制，严格保障每条流的延时和抖动。	机制适用于周期性信号一般用于有超低延时、超低抖动要求的控制信号传输,且其配置极其复杂,在车载通信的应用尚存争议
PS-Peristaltic Shaper	IEEE 802.1Qch	将时间片或周期分为相等的奇（odd)偶(even)间隔(interval)。在偶间隔接收报文，并在奇间隔发送。	对时间同步要求没有TAS高。从实现方式角度出发，PS的奇偶间隔也可作为TAS的两个队列
ATS-Asynchronous Traffic Shaper	IEEE 802.1Qcr	源于Urgent-Based Shaper，该技术不再依赖于时间同步协议，利用交换机本地的时间变量，提前计算并为每条流量分配本地合格时间（Eligibility time）,合格时间在队列中生效，在达到每条流对应的合格时间时开始发送对应的报文,ATS队列转发基于令牌桶算法。	适合非周期性流量传输

3）高可靠

TSN 关于网络高可靠保障的协议是 IEEE 802.1CB，如图 18 所示。其核心机制是报文复制与报文丢弃，即在发送端发送两份一样的报文，若在接收端同时收到，则丢弃一份，接收一份。针对车载应用，可能需要对现有 TSN 可靠性机制进行完善和补充。

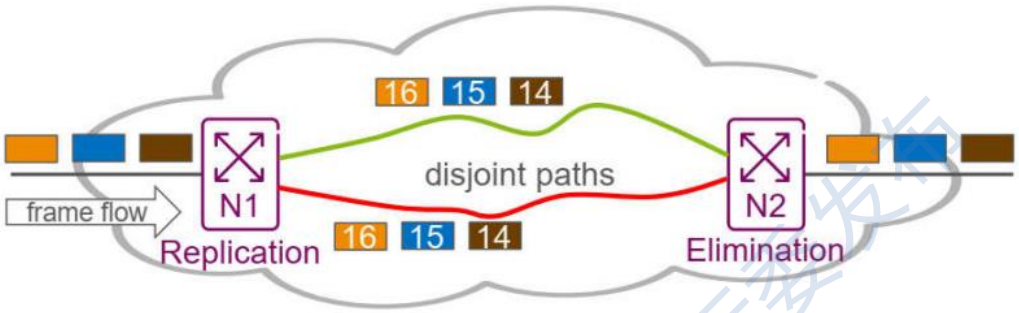

图 20 TSN 可靠性工作原理

4.2.3 AI 计算

由于智能驾驶领域的高性能计算的需要，传统的 CPU 执行 AI 算法速度慢、性能低，用 GPU 虽然速度快，但功耗大，而且通用 GPU 不是专门针对 AI 算法开发的。因此 AI 计算领域一般使用针对 AI 算法的专用芯片，简称 AI 芯片。AI 芯片使用的方法原理当前还在探索阶段，百花齐放，包括 GPU、FPGA、DSP 等。AI 计算模块需要广泛考虑不同种类的 AI 芯片的适配，灵活方便的对其提供支持。

4.2.4 外设驱动

驱动程序本质上是软件代码，主要作用是计算机系统与硬件设备之间完成数据传送的功能，只有借助驱动程序，两者才能通信并完成特定的功能。如果一个硬件设备没有驱动程序，只有操作系统是不能发挥特有功能的，也就是说驱动程序是介于操作系统与硬件之间的媒介，实现双向的传达，即将硬件设备本身具有的功能传达给操作系统，

同时也将操作系统的标准指令传达给硬件设备，从而实现两者的无缝连接。外设驱动主要包括片内外设、板级外设等，属于 BSP 的驱动。

4.2.5 统一分布式时钟

统一分布式时钟是通过对本地时钟的某些操作，达到为分布式系统提供一个统一时间标度的过程。在集中式系统中，由于所有进程或者模块都可以从系统唯一的全局时钟中获取时间，因此系统内任何两个事件都有着明确的先后关系。而在分布式系统中，由于物理上的分散性，系统无法为彼此间相互独立的模块提供一个统一的全局时钟，而由各个进程或模块各自维护它们的本地时钟。车控操作系统各个域间时钟技术不一致有两个原因：

（1）时钟偏移：各个域启动顺序不同，初始化流程也不同，因此各个域本地时钟开始计数的时间本身就是不一致的，这种偏差称为时钟的偏移。

（2）时钟漂移：受不同温度、寿命与物理特性的影响，各个时钟的晶振频率并不完全一样，因此随着时间的增加，原本一致的时钟也会逐渐产生偏差，这种偏差称为漂移。

对于系统中的时钟同步，并不要求各时钟完全与统一标准时钟对齐。只要求知道各时钟与系统标准时钟在比对时刻的钟差以及比对后它相对标准钟的漂移修正参数即可，勿须拨钟。只有当该钟积累钟差较大时才作跳步或闰秒处理。因为要在比对时刻把两钟“钟面时间对齐，一则需要有精密的相位微步调节器会调节时钟用动源的相位，另外，各种驱动源的漂移规律也各不相同，即使在两种比对时刻时钟完

全对齐，比对后也会产生误差，仍需要观测被比对时钟驱动源相对标准钟的漂移规律，故一般不这样做。

4.3 POSIX 及其他接口

POSIX 全称为可移植性操作系统接口，是一种关于信息技术的 IEEE 标准，它包括系统应用程序接口（API），以及实时扩展 C 语言。POSIX 定义了标准的基于 UNIX 操作系统的系统接口和环境来支持源代码级的可移植性。目前，POSIX 主要提供了依赖 C 语言的一系列标准服务，在将来的版本中，标准将致力于提供基于不同语言的规范。

为嵌入式实时操作系统考虑，POSIX 创建了 PSE51 子集，在 IEEE.13-2003 中进行了定义。AUTOSAR 也仅支持 PSE51，仅为应用程序提供了受限的操作系统 API。在车控操作系统中，也推荐这种受控的系统 API。

对于 POSIX 中未定义的系统功能，需要在其他接口部分定义。例如，系统定义了一系列用于系统性能监控的 API，用于应用程序自感知其占用内存、CPU 等资源的情况。这些 API 需要在其他接口中定义，并提供完善的文档。

4.4 系统中间件及服务

系统中间件及服务主要是为上层应用如功能软件平台提供基础的系统服务，其采用面向服务的架构设计思路，减少服务之间的耦合，保障服务实现的灵活性和可扩展性，更加方便兼容更多的成熟的标准服务和生态，从而提高功能软件和应用软件的开发和运行效率。

为了最大化车控操作系统的应用，基于服务化设计的基础计算框架可以灵活扩展，兼容其他标准，如 AUTOSAR Adaptive 平台提供的系统服务和平台服务，ROS 服务等，最大化的利用现有的智能驾驶生态。

4.4.1 分布式通信中间件

分布式通信中间件主要对上层业务模块的数据交互提供一种可靠、实时的异构分布式通信框架，使得不同硬件、不同语言、不同操作系统平台提供的服务可以在正确的时间、正确的地方，分享正确的数据。分布式通信中间件提供了应用程序和网络数据传输的标准化绑定，实现了以数据为中心的分布式实时通信。分布式数据分发主要分为数据本地重构层和以数据为中心的数据发布和订阅。数据重构层主要对发布和订阅服务进行抽象，与底层服务建立映射关系；数据的发布和订阅是整个机制的核心和基础，负责数据的传输以及相关服务质量（QoS）的控制保障。分布式通信中间件应支持操作系统内和不同操作系统间的通信，支持 DDS、SOME/IP 等通信协议，也支持 RESTful 的方式。分布式通信中间件的交互应广泛支持流行的通信介质，包括不限于车载以太网、CAN、无线通信等。以分布式通信中间件为核心的通信框架未来可以拓展支持车-云，车-车等统一通信服务。

一般来讲，分布式通信中间件一般采用面向服务的架构（Service-Oriented Architecture，SOA）实现。SOA 是一种架构设计思想，它将具备不同功能的应用程序单元以服务的形式提供给用户，通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的语言进行

定义的，它独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。

4.4.2 应用调度和生命周期管理

应用调度和生命周期管理提供操作系统在平台和应用两个层面的执行管理能力，负责操作系统平台初始化和应用程序的启动和关闭，以及使用操作系统内核 API 和 POSIX 系统服务，执行应用程序的运行时调度。在 AUTOSAR AP 标准中对应于 Execution Management 模块（执行管理），包括两个方面的职责：

（1）平台生命周期管理

执行管理作为 AP 平台启动阶段的一部分启动，并负责初始化 AP 平台和部署应用程序。

（2）应用程序生命周期管理

执行管理负责按顺序启动和关闭部署的应用程序。根据配置清单（Manifest 文件）确定部署的应用程序集，并根据应用程序依赖关系启动（或关闭）其他应用程序。

4.4.3 安全框架和服务

安全框架和服务需要提供支持高功能安全目标实现的系统架构和基本服务能力。一般需要从硬件、软件、方法和流程等多方面协同设计和保障实施。在硬件层面，需要提供多传感器冗余配置和融合设计，计算硬件、线控、供电/线束等进行冗余设计，以及计算硬件提供底层安全机制（锁步、诊断、保护等）。在软件层面，需要实现操作系

统内核支持隔离和分区机制，运行时监测，checkpoint，watchdog 等安全机制。在方法流程层面，需要参考功能、预期功能安全、信息安全和软件工程等领域的标准流程和最佳实践。

信息安全指的是汽车全生命周期的信息安全，而不仅仅是车辆交付时的信息安全，因为在智能网联汽车的整个生命周期随时可能受到非法数据的攻击。信息安全也是全方位的信息安全，现在由于对不同应用的需求不同，车辆平台会预留各种接口，除 CAN/CAN-FD/FlexRay/LIN 等总线系统、OBDII 等法规接口，还有目前车辆广泛使用的蓝牙、USB 以及 TPMS、PEPS 等天线系统等，以及随着网联化发展而安装的各种应用及其联网数据，都可能是信息安全攻击的切入点，因此信息安全防护也需要全方位的。

目前常用的信息安全措施包括：鉴权认证、通信加密、完整性保护、安全启动、攻击防护等策略。

鉴权认证：用户只有获取有效口令方可访问信息通信链路，且系统对其接入访问权限进行限制；

通信加密：可通过特定算法对通信数据进行加解密，以防止非法的通信数据访问或者攻击；

完整性防护：通过特定算法或协议对通信数据或者存储的重要信息进行计算以确保信息完整可用；

安全启动：在系统启动引导 ECU 固件代码之前，使用身份验证机制（通常以基于硬件支持的 Root-of-Trust 机制），对启动模块和代码进行逐级数字验证以保证代码的真实性和完整性。一般情况下 ECU

固件代码和配置文件会被检查，常用方法为检查 ECU 固件散列值的签名；

攻击防护：通过技术手段实现对于攻击数据的鉴别，并在数据遭受到非法破坏或获取时能够进行有效的防护，防止产生进一步的损害。

4.4.4 责权管理

自动驾驶系统的复杂性使得传统的驾驶安全理念不再适用。系统开发和功能验证面临挑战，系统故障和发生事故后定位困难，主机厂和供应商责任难以界定。责权管理提供在操作系统层面的底层机制，将自动驾驶各类软硬件资源进行统一信息标识和跟踪溯源，进而实现信息安全管理、故障定位和责任归属管理。

4.4.5 应用更新管理

应用更新管理提供通过 OTA 以安全可靠的方式更新软件及其配置的服务和能力。车控操作系统要求内置支持车辆诊断的软件配置管理，并支持安全、可靠和高效的系统和应用（包括算法和数据）的更新过程，以及满足支持多个客户端更新并实现快速下载的要求。

4.4.6 诊断日志

诊断日志提供支持应用开发的调试和日志记录能力，提供包括 ISO14229-1 在内的通用诊断服务，应支持不同诊断客户端的并行处理和数据收集。

4.5 实时安全域

车控 ECU 需要能够快速响应外界复杂环境或者状态的变化，并及时做出必要的响应以确保安全，因此，车控操作系统的实时安全域

不可或缺。由于 MCU 经济性、便利性等方面的优势，当前车控 ECU 领域实时安全基于 MCU 实现。实时安全域操作系统内核提供了内核的基本通用服务、诊断服务、存储服务、板内和板间的通信服务。系统基础软件包含了对于 ECU 板上的外部设备的驱动，统一内部设备和外部设备的 I/O 接口，同时对于复杂的传感器和执行器驱动，也可以提供直接访问硬件资源的能力。实时安全域通过 CAN/CAN-FD/FlexRay 等高速总线与车身主干网络其它域控制器或当前 ECU 挂载的传感器进行通信，获取当前车辆及环境数据，并通过其上运行的各种检测任务针对获取到的各种数据进行监测判断以确定当前车辆是否处于安全状态。在必要的时候，实时安全域可通过其上运行的安全控制应用实现对车辆的安全控制，因此实时安全域需要具备任务调度，实时通信等能力，外设及状态监控等能力。出于对车辆不同来源数据的获取、判断以最终实现对车辆控制的目的，实时安全域亦需要与 SOC 部分进行基于时间同步的实时数据交换。同时，实时安全域也需要通过对系统各部分电源进行监控，以确保 ECU 电源处于安全状态。

随着汽车智能化和网联化的发展，实时安全域需要 FOTA，以适应车辆不断增长的应用需求；随着传感器复杂化，各总线上需要传输数据量越来越大，目前使用 CAN/CAN-FD//FlexRay 等总线类型也越来越无法满足数据对总线带宽的要求，且目前市场上 MCU 普遍开始实现对 Ethernet 的支持，实时安全域需要实现对 Ethernet 的支持，从而实现自动驾驶应用对于数据的大带宽、低时延的需求，降低系统开

发和维护的难度。

4.6 系统软件工具链

4.6.1 组件模型化开发 IDE

组件模型化开发 IDE 为用户提供了一种应用快速开发的方式。其将自动驾驶相关的常用功能模块组件化、标准化、图形化，在 IDE 中以组件列表的形式展现，用户可以拖拽相应的组件到功能设计视图中，配置其属性和连接关系，生成业务框架代码，以可视化的方式快速构建业务应用。

4.6.2 性能分析工具

通过一定的测试次数、一定的采样间隔，统计应用程序占用的 CPU、内存、网络、I/O 请求等随时间变化的数据，并以图形化的形式展现测试结果，便于用户直观地分析程序的性能，进而对程序运行效率进行有针对性的优化。

4.6.3 车规编译器

软件是智能汽车的灵魂，其可靠性和安全性对汽车行驶安全至关重要，因此，采用功能安全达到特定级别的编译器去构建自动驾驶应用是一种很好的选择。例如，可根据 ISO 26262 认证，对选定的编译器进行危害分析和风险评估，确保编译器的各种功能达到汽车完整性安全等级（ASIL）的预期级别。

4.6.4 代码生成器

基于中间件架构实现的各种服务是以使用中立语言描述的文件（IDL 文件）作为彼此之间通信的契约。在被具体语言使用前，需要

使用代码生成器生成与具体语言相关的接口文件，保证协议对齐。

4.6.5 仿真工具

自动驾驶算法通过各种传感器探测周围环境的信息，进行决策后对车辆进行控制。为了在仿真环境对自动驾驶算法进行测试，需要有仿真工具支持，模拟自动驾驶算法的各种输入，并响应其输出，从而“欺骗”它，让它以为在真实世界工作。

采用仿真测试，可以尽早地对系统各阶段的交付物进行测试，及时发现问题，避免问题累积，降低修正成本。同时，采用仿真测试，可以对自动驾驶的各场景进行海量测试，加快算法的训练和测试速度。

5 车控操作系统功能软件架构

功能软件是车控操作系统中根据面向服务的架构设计理念，通过提取智能驾驶核心共性需求，形成智能驾驶各共性服务功能模块，高效实现驾驶自动化功能开发的软件模块。根据图 15 车控操作系统参考架构，结合功能软件的主要核心功能和应用需求，形成车控操作系统功能软件架构（参见图 21）。

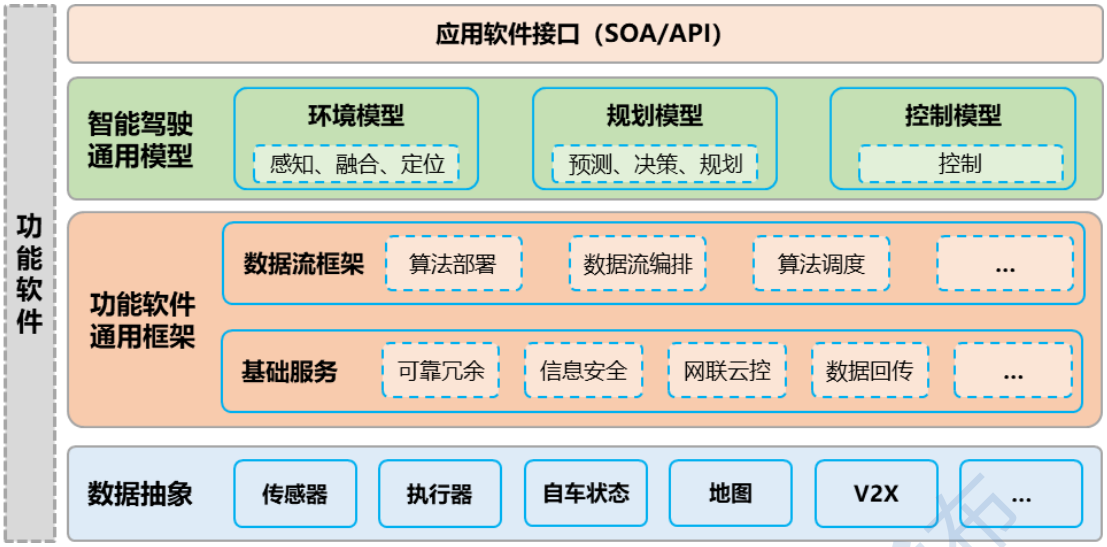

图 21 功能软件架构

5.1 应用软件接口

应用软件接口提供统一的开发环境和工具，体现为各种不同形式的 SDK，例如环境模型、功能配置以及各种算法 SDK 等，以及包括应用开发所必要的工具链、软件包、开发接口、开发文档、示范应用和配置等。应用软件接口可以是基于面向服务(SOA) 的订阅形式架构，也可以是使用统一开发接口 (API) 函数调用的形式，使用这些应用软件接口和 SDK 可以进行高效、灵活、敏捷的定制化开发。

5.2 智能驾驶通用模型

智能驾驶通用模型是对智能驾驶中智能认知、智能决策和智能控制等过程的模型化抽象。

5.2.1 环境模型

环境模型作为智能认知框架，为智能决策和智能控制提供模型化的广义环境信息描述。环境模型调度各类感知、融合和定位算法，对传感器探测信息，车-路、车-车协同信息，以及高精地图先验信息进行处理加工，提供探测、特性、对象、态势、场景等各级语义的道路

交通环境和自车状态信息。

5.2.2 规划模型

规划模块根据环境模型、自车定位、个性化设置和自车状态反馈等信息，为自车提供未来一段时间内的行驶轨迹，主要分为行为预测、行为决策和运动规划三大部分。行为预测是根据感知和地图数据对其他交通参与者未来的行驶轨迹进行预测，为行为决策提供更全面、可靠的参考信息；行为决策为自车提供行为策略，同时为运动规划提供相应的规划约束条件，保证规划结果不仅满足交通法规等硬性要求，同时更加符合人的驾驶策略；运动规划根据以上信息，为自车规划未来一段时间内的安全、舒适、正确的轨迹。

5.2.3 控制模型

控制模型主要由常规工况和降级工况组成，其中常规工况主要针对 ODD 以内的动态驾驶任务，降级工况主要针对发生系统性失效或者超出 ODD 以外的动态驾驶任务，均需要进行输入处理、状态决策、控制计算及执行输出等。针对上游及底盘信息的输入，以及控制输出均需要适配层去匹配不同的功能算法框架平台及车辆平台；针对横纵向及紧急控制等算法模块需要进行故障诊断、配置及标定接口模块统一管理。

5.3 功能软件通用框架

功能软件通用框架是承载智能驾驶通用模型的基础，分为数据流框架和基础服务两部分。

5.3.1 数据流框架

数据流框架向下封装不同的智能驾驶系统软件和中间件服务，向智能驾驶通用模型中的算法提供与底层系统软件解耦的算法框架。数据流框架的主要作用是对智能驾驶通用模型中的算法进行抽象、部署、驱动，解决跨域、跨平台部署和计算的问题。

5.3.2 基础服务

基础服务是功能软件层共用的基本服务，其主要服务于智能驾驶通用模型或功能应用，但其本身不局限于智能驾驶。

5.3.2.1 可靠冗余

基础服务平台的可靠冗余组件是保证自动驾驶安全可控的关键，也是车控操作系统取得操作系统全栈功能安全认证的重要保障。可靠冗余组件将系统中其它所有软件和硬件模块都抽象为被管理实体，通过与所有被管理实体的交互，完成对整个系统的监测和故障处理。可靠冗余组件对于所有被管理实体都应该是透明的，任何软硬件实体都不应该意识到有可靠冗余模块的存在。

5.3.2.2 信息安全

信息安全基础服务为车端数据定义了数据类型和安全等级，为车端功能和应用定义的数据处理功能定义。根据以上分类和车辆运行场景，定义数据处理规则。数据流框架上的算法部署和数据流编排模块，按规则定义控制算法部署和数据交换。

5.3.2.3 网联云控

基础服务中的网联云控服务可提供操作系统的安全冗余信息、超视距信息和通用模型的信息。网联云控可通过 LTE-V2X、4G/5G 的通讯方式，实现与车车通讯、车云通讯、车人通讯和车与路侧基础设施通讯。通过网联云控服务，可使得操作系统具备车路协同的能力，实现协同感知、协同规划等服务，加速智能驾驶的落地。

网联云控服务既提供标准的、抽象的信息服务，如红绿灯信息、交通提醒信息、安全预警信息、周边车辆行驶信息，也提供可插扩算法的能力，可以新增、转换、适配不同的云控算法和应用。网联云控服务是车内外信息通信的桥梁，操作系统可把自车状态、行驶意图广播到周围环境中或上传到云平台，同时也可从周围环境或边缘云获得感知信息(如障碍物信息)、决策规划建议，甚至运行轨迹信息。网联云控服务也可提供当前车辆中通讯模块的工作状态，如网络连接状态、信号强度、断网或异常通知等信息。

5.4 数据抽象

数据抽象通过对传感器、执行器、自车状态、地图以及来自云端的接口等数据进行标准化处理，为上层的智能驾驶通用模型提供各种不同的数据源进而建立异构硬件数据抽象，达到功能和应用开发与底层硬件的解耦和依赖。

6 标准化建议

6.1 技术发展路线总结

随着通信技术和高性能计算技术发展，汽车电子电气架构正在发生革命性的变革，相关的技术发展路线总结如下：

（1）汽车电子电气架构从分布式架构到域集中式架构和中央集中式架构转变。

（2）汽车电子控制单元逐步演进到域集中式计算平台和中央集中式计算平台。依据研究报告提出的车控操作系统架构，结合车载操作系统架构研究报告，提出了一种中央集中式计算基础平台参考架构，参考图 22，集成了车控操作系统和车载操作系统。

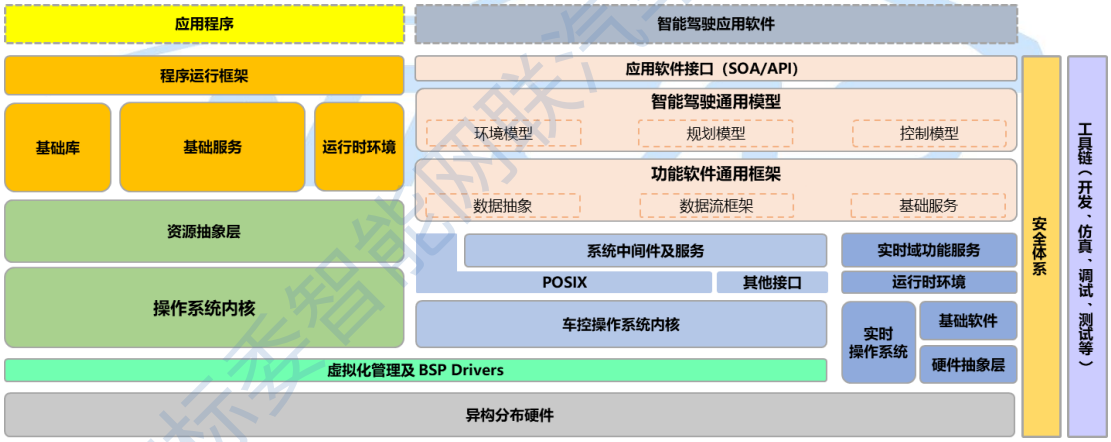

图 22 中央集中式计算基础平台参考架构

（3）由于技术发展的渐进性和汽车产业链的复杂性，当前阶段智能汽车还是以域集中式计算平台为主，未来很长一段时间域集中式计算平台和中央集中式计算平台将会在智能汽车产品上共存。

（4）计算平台将全面支持网联和云控协同感知、决策、控制，数据收集与处理，支持高等级自动驾驶。

在智能网联汽车的众多技术中，软件已经成为核心技术。车控操

作系统作为智能网联汽车的软件平台，整合和调度车内的异构计算资源，明确产业链的分工，推动智能网联汽车技术的不断演进。除了整合车内资源外，车控操作系统也将和车路云基础软件进行深度融合，作为整个智能交通网联的支撑平台。

6.2 标准化建议

车控操作系统是智能网联汽车的基础软件部分，运行于智能网联汽车车载智能计算基础平台。智能网联汽车的复杂性，需要通过车控操作系统软件架构和接口的标准化实现产业链的分工协同，提高开发效率，保障软件平台的安全可信，减少对接成本，构建统一生态。

6.2.1 架构和要求类

架构是操作系统的顶层设计，明确了对系统实现的约束条件，为维护决策提供依据，以及便于在较高层面上实现复用。因此，将车控操作系统的架构和技术要求进行标准化有助于实现操作系统的行业共识，降低主机厂、零部件供应商等之间的沟通成本，实现操作系统软件复用，提高开发效率。车控操作系统的技术要求包括功能要求和性能要求等，通过技术要求和测试方法的标准化，可以评价车控操作系统的基本功能和性能能否满足产品设计要求。

表 3 架构和要求类标准化建议

标准化对象	分析	必要性	启动建议
车控操作系统总体架构及要求	通过标准化，可以实现域集中式计算平台驾驶自动化功能的高效开发，实现可扩展、可复用。	必要	优先级高
车控操作系统性能要求及测试方法	通过规定车控操作系统的基本技术要求、功能要求和性能要求及对应的测试方法，保证产品的质量。	必要	优先级低

车用操作系统总体架构	融合了车控操作系统和车载操作系统，满足中央集中式计算平台技术发展需求。	必要	优先级低

6.2.2 安全要求类

功能安全和信息安全是车控操作系统产品可靠安全运行的必要组成部分。车控操作系统为智能汽车自动驾驶功能提供运行环境，其功能安全要求是保证整个系统功能安全的基础和核心。车控操作系统是单域核心攻击目标，对于车控操作系统的攻击，可以直接影响到智能网联汽车的功能安全和人身安全，也会涉及到个人隐私数据和包括测绘等敏感数据。对于车控操作系统的安全要求类的标准制定，可以设定操作系统安全基线，最大限度保护智能网联汽车的安全运行，数据的安全存储和处理。

表 4 安全要求类标准化建议

标准化对象	分析	必要性	启动建议
车控操作系统信息安全要求	车控操作系统作为智能网联汽车的基础软件平台，信息安全要求和测试方法是保证智能网联汽车的网络安全和数据安全前提，建议尽快启动国标。	必要	优先级高
车控操作系统功能安全要求	车控操作系统作为智能网联汽车的基础软件平台，功能安全要求和测试方法是保证车辆安全运行的前提。	必要	优先级低

6.2.3 接口和互操作类

车控操作系统模块间接口标准化主要是为了为智能驾驶等应用提供标准化的运行环境和服务，满足不同操作系统之间以及操作系统内部不同层次之间的实现通信，高效实现和互操作，实现操作系统解

耦，满足跨平台、跨车型、可扩展等要求。

表 5 接口和互操作类标准化建议

标准化对象	分析	必要性	启动建议
应用于自动驾驶功能的传感器接口	通过将感知传感器接口标准化，可以实现传感器的抽象，从而屏蔽不同类型的传感器，方便功能软件感知融合模块的算法实现。	必要	优先级低
车控操作系统面向应用程序的接口	通过向上提供标准化的面向应用程序的接口，可以屏蔽应用软件对操作系统的依赖，实现解耦，方便应用软件的开发。	必要	优先级低
车用操作系统间通信要求	通过标准化实现车控操作系统和车载操作系统的互操作性，降低不同开发商之间的沟通成本。	必要	优先级低

智能网联汽车的应用正在高速发展期，车控操作系统技术作为智能网联汽车的支撑技术，具体的功能模块和接口也会随着系统的不断演进而演进，接口类标准比较倾向于市场化行为，因此可以通过制定接口类团体标准，以适应快速变化的市场需求。