车载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

随着智能网联汽车的飞速发展，传统的分布式的电子架构已经不 能满足整车的需求，电子架构正在朝着由分布式向集中式，由封闭到 开放，面向智能化、网联化和集中化的路径发展。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，构建了车用操作系统标准体系，为车用操作系统标准化工作的开展提供了指导方向。车用操作系统是运行于车内的程序集合，管理硬件资源，提供软件平台和界面接口，为上层应用提供基础服务。车用操作系统从与整车正常运行是否相关的角度，分为车控操作系统和车载操作系统。车载操作系统主要应用于中控、仪表和 T-box，提供车载信息娱乐服务，可具备网联功能，提供导航、多媒体娱乐、语音、辅助驾驶、AI 等高级功能。近年来，随着汽车电子、云计算、大数据等技术的快速发展，车载操作系统的架构和技术功能不断演进，急需开展相关标准化需求研究，指导车载操作系统的研发、测试、示范和运行等。

本研究报告的车载操作系统研究范围是介于应用程序和硬件抽象层之间，主要由车载操作系统内核、资源抽象层、基础库、基础服务、运行时环境及程序运行框架组成。

本研究报告涉及的技术内容在《车载操作系统总体技术研究报告》中有相应的详细分析。

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：阿里巴巴（中国）有限公司、上汽大众汽车有限公司

参与单位：斑马网络技术有限公司、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、国汽（北京）智能网联汽车研究院有限公司、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）、东软集团（大连）有限公司、惠州市德赛西威汽车电子股份有限公司、大陆汽车车身电子系统有限公司、高通无线通信技术（中国）有限公司、长城汽车股份有限公司、北汽福田汽车股份有限公司、北京汽车股份有限公司、上海博泰悦臻电子设备制造有限公司、华为技术有限公司、中兴通讯股份有限公司、泛亚汽车技术中心、江苏智能交通及智能驾驶研究院、Elektrobit、上海机动车检测认证技术研究中心有限公司、安徽江淮汽车集团股份有限公司、东风商用车有限公司、一汽解放汽车有限公司、上海汽车集团股份有限公司零束软件分公司、东风日产乘用车公司、东风汽车集团有限公司技术中心、上汽通用五菱汽车股份有限公司、襄阳达安汽车检测中心有限公司、北京百度智行科技有限公司。

参与人员：王琳、刘大鹏、卜烨雯、金一华、刘克、满志勇、田思波、潘晏涛、吴含冰、张路、刘卫国、霍克、郭盈、周波、周海龙、李伟、伍宇志、唐侨、许劲、李俨，陈书平、石晓坤、毛雷、来冬清、钱国平、刘东、刘珺、秦文婷、顾照泉、周铮、陈晓、陈琨、江浩、刘翔海、王红燕、周鹏、王观、李兵、黄慧丽、桂绍靖、管杰、吴勇刚、刘时珍、孙华、王圭、邹德英、武扬、程周、王振、彭杨、江恒、高海龙、彭伟、贾元辉。

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。

1.1.2 车载操作系统 on-vehicle operating system

运行于车载芯片上，管理和控制智能网联汽车车载软件、硬件资源的软件集合，为智能网联汽车提供除驾驶自动化功能实现以外的服务，包括车载信息娱乐、网联、导航、多媒体娱乐、语音、辅助驾驶、AI 等服务。

1.1.3 单系统架构 single system architecture

单个车载操作系统的架构，由车载操作系统内核、资源抽象层、基础库、基础服务、运行时环境、程序运行框架和车载操作系统安全模块组成，对底层硬件和上层应用程序提供统一的接口。

1.1.4 多系统架构 multisystem architecture

在同一套硬件之上运行多个车载操作系统单系统的架构，分为硬件隔离、虚拟机管理器、容器三类多系统基础架构，以及两类或三类基础架构的混合架构。

1.1.5 资源抽象层 resource abstraction layer

运行于车载操作系统内核上，为车载操作系统应用和基础服务提供SoC芯片平台硬件的资源抽象、整车信号的资源抽象、外部IoT设备的资源抽象和外围IC的资源抽象。

1.2 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）

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

OBD：车载诊断（On-Board Diagnostics）

OBU：车载单元（On-Board Unit）

OTA：空中下载（Over the Air）

RTOS：实时操作系统（Real Time Operating System）

V2X：车与外界的互联（Vehicle-to-Everything）

2 车载操作系统架构现状及趋势

2.1 车载操作系统现状

目前，应用于车载领域的操作系统有括QNX、Linux/AGL、RT-Linux、AliOS、AliOS RT、Android和鸿蒙OS等，各个车载操作系统的介绍参见附录A。车载操作系统从功能角度可分为非实时操作系统和实时操作系统；从应用领域角度通常可分为用于中控的车载操作系统，用于仪表的车载操

作系统和用于 T-box 的车载操作系统。他们之间的对应关系分析如表 1 所示。

表 1 车载操作系统现状

车载操作系统的特征		Linux/AGL	RT-Linux	QNX	AliOS	AliOS RT	Android	鸿蒙 OS *[2]
功能	非实时	Y			Y		Y	Y
	实时		Y	Y		Y		Y
功能安全 *[1]	具备功能安全认证			ASIL-D		ASIL-D		ASIL-D
应用领域	中控	Y		Y	Y		Y	Y
	仪表	Y	Y	Y		Y		Y
	T-box	Y		Y				
	注：表中对各操作系统支持的应用领域仅作一般性分类参考。 *[1]通过的 ASIL-D 认证不一定是全部车载操作系统 *[2]鸿蒙 OS 根据应用需求属性分别运行在实时和分时内核之上							

2.2 国内外标准化现状

2.2.1 AGL 开源规范

AGL（Automotive Grade Linux）是一个协作开源项目，由 Linux 基金会管理，它将汽车制造商，供应商和技术公司聚集在一起，以加速开发和采用完全开放的联网汽车软件堆栈。以 Linux 为核心，开发一个开放式平台，作为事实上的行业标准，以实现新功能和技术的快速开发。2014 年，Linux 基金会发布了 AGL （Automotive Grade Linux）开源规范 1.0 版本，它是业界首个开放式车载信息娱乐（IVI）软件规范。这也是第一次汽车制造商、供应商以及开源开发者可以基于同一个规范进行协作，该规范

很好的定义了将来的联网汽车提供基于 Linux 的软件堆栈。AGL 发布了首个 AGL 参考实现平台，平台基于 Tizen IVI 平台，用来运行 HTML5 应用。基于 Tizen IVI,AGL 添加了直观的 UI / UX 以及用 HTML5 和 JavaScript 编写的各种应用程序，并支持多种硬件架构。

2.2.2 GENIVI 车载信息娱乐系统标准

GENIVI 是以宝马为首的知名企业建立的应用于车载系统的开放式软件平台和操作系统，基于 Linux 平台，形成从研发到应用的闭环生态。它是一个非营利行业联盟，致力于推进车载信息娱乐系统(IVI，In-Vehicle Infotainment)开源开发平台被广泛的采用。该联盟提出需求、提供参考实现、提供认证服务，以及维护一个开源 IVI 社区。GENIVI 的工作将会缩短开发生命周期、及时响应市场,以及降低公司开发 IVI 设备和软件的成本。

2.2.3 国内车载操作系统标准化

2019 年 10 月，汽标委发布《车用操作系统标准体系》，提出了车载操作系统和车控操作系统的标准化建议。目前，汽标委正在开展车载操作系统的标准化需求研究，但尚未启动车载操作系统的标准化工作。

2.3 车载操作系统架构演进趋势

随着车辆的功能从单一的安全驾驶功能逐步向智能化、娱乐化、个性化过渡，单一系统已无法满足多样的功能需求，车载操作系统架构由单系统向多系统转变。

3 车载操作系统架构标准化需求

目前各车载操作系统的术语、架构和各模块要实现的功能没有形成业界共识，导致车厂在选择其所需服务和应用的车载操作系统时，无据可依；

供应链相关方在互操作和产品研发测试时，以定制化为主，很难实现可复制规模化推广。

具体来说，车载操作系统架构的标准化需求如下：

a) 术语标准化

1) 车载操作系统的边界定义不清晰，需要明确与车用操作系统、车控操作系统的关联关系；

2) 车载操作系统对各种资源的抽象缺乏具体的定义。

b) 车载操作系统模块的划分和各模块的功能要求标准化

1) 车载操作系统对外围 IC、IoT 设备、硬件等资源的抽象尚无统一的架构设计；

2) 车载操作系统可提供的基础服务类型以及各类服务的功能要求尚无共识，特别是对于较新的辅助驾驶、AI 等服务。

c) 多系统架构的标准化

由于不同的服务和应用对车载操作系统的实时性、安全性和功能等要求不同，多系统是必然趋势，但目前多系统架构方案众多，尚未形成统一共识，车厂很难根据自身对多系统的安全性和灵活性等需求选择合适的多系统架构方案。

4 车载操作系统架构分类建议

车载操作系统架构可分为单系统架构和多系统架构。两类架构均可实现一芯多屏（多屏融合、多屏互动）、单屏多系统（虚拟运行环境、多应用生态融合）、一芯多功能单元（信息娱乐、T-box 等）三类应用。

a) 单系统架构

单系统架构仅涉及单个车载操作系统的架构，由车载操作系统内核、基础库、基础服务、运行环境及程序运行框架组成。车载操作系统对底层硬件和上层应用程序提供统一的接口，实现车载操作系统与硬件和上层应用程序的解耦。

b）多系统架构

多系统架构是同一套硬件之上运行多个车载操作系统单系统的架构，可分为硬件隔离、虚拟机管理器、容器三类多系统基础架构，以及两类或三类基础架构的混合架构，用于满足不同功能、性能和安全的隔离需求。

5 单系统架构和功能要求建议

5.1 总体建议

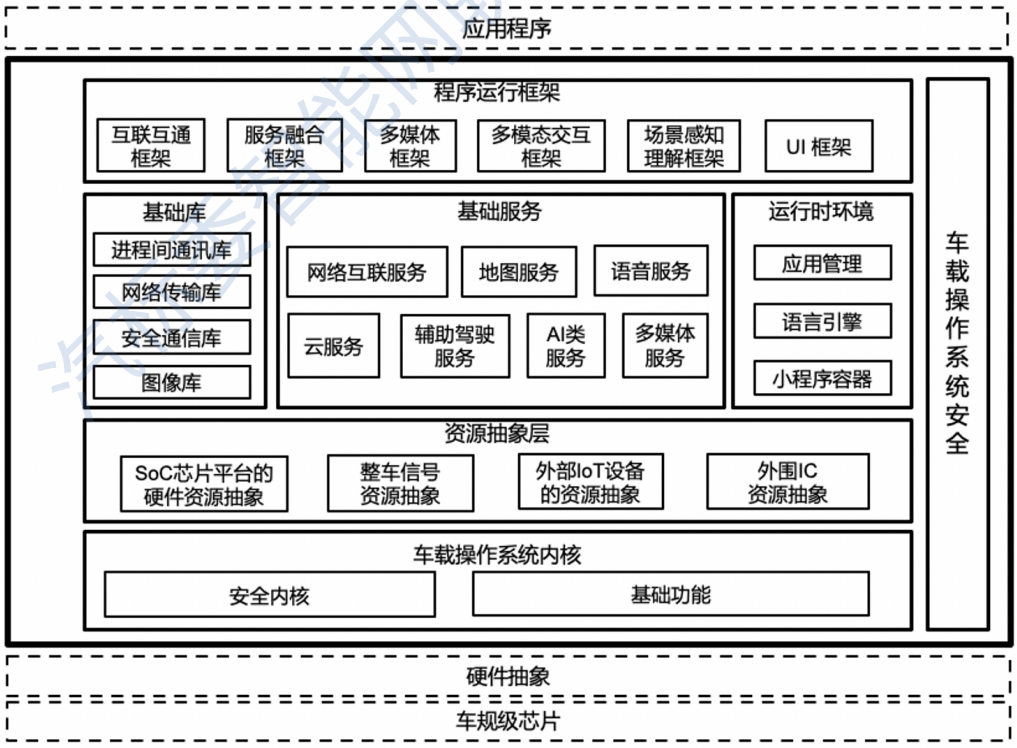

图 1 车载操作系统单系统架构

车载操作系统单系统架构如图 1 所示，其边界介于应用程序和硬件抽象层之间，由车载操作系统内核、资源抽象层、基础库、基础服务、运行时环境、程序运行框架和车载操作系统安全模块组成。车载操作系统通过组件化和自包含的模块设计架构，支持通过配置对系统功能进行灵活裁剪。针对特定的功能域，根据比如娱乐信息域、仪表盘域或者基础服务系统等方面的需求，车载操作系统内核可配置为不同的核心软件架构。不同车载操作系统与车载操作系统单系统架构各功能模块的对应关系如附录 B 所示。

a) 车载操作系统内核

车载操作系统内核可管理系统资源，提供对软件层面的抽象和对硬件访问的抽象，包括安全内核和基础功能两部分。这两部分的具体功能要求见 5.2 节。

b) 资源抽象层

资源抽象层为车载操作系统应用和基础服务提供四种类型的资源抽象，包括 SoC 芯片平台硬件的资源抽象、整车信号的资源抽象、外部 IoT 设备的资源抽象和外围 IC 的资源抽象。各类资源抽象的功能要求见 5.3 节。

c) 基础库

基础库为车载操作系统应用提供基础资源库，包括进程间通讯库、网络传输库、安全通信库、图像库等。基础库的具体功能要求见 5.4 节。

d) 基础服务

基础服务是车载操作系统为上层应用程序提供的基础服务的封装，使得应用程序按照基础服务提供的方式，有效调度和使用硬件资源。车载操作系统提供的基础服务包括网络互联服务、地图服务、语音服务、多媒体

服务、云服务、辅助驾驶服务、AI 类服务等。各类基础服务的功能要求见 5.5 节。

e）运行时环境

运行时环境为应用程序的运行提供环境支持，包括应用管理、语言引擎和小程序容器。运行时环境的具体功能要求见 5.6 节。

f）程序运行框架

程序运行框架为应用程序提供访问接口，实现车载操作系统与上层应用的解耦。程序运行框架可包括互联互通框架、服务融合框架、多媒体框架、多模态交互、场景感知理解框架、UI 框架等。各类程序运行框架的功能要求见 5.7 节。

g）车载操作系统安全

车载操作系统安全确保车载操作系统的信息安全和功能安全，具体要求见 5.8 节。

5.2 车载操作系统内核

车载操作系统内核提供安全内核和计算内核（任务优先级调度等）、存储管理、进程间通信管理、网络接口等基础功能。车载操作系统内核既可以是宏内核，也可以是微内核。

任务优先级调度保证高优先级业务被优先执行。如果辅助驾驶相关的优先级较高的任务不能按时执行，不仅会影响软件功能，还会带来系统安全的隐患。其基本实现方式是基于任务优先级的抢占调度，优先级高的任务可以无理由地抢占并利用处理器的执行资源。任务优先级调度应支持两个优先级相同的任务的调度机制，以及周期性的高优先级任务调度机制。

基于任务优先级调度，车载操作系统应支持系统快速启动。可通过资源场景化调度、动态优先级配置、服务启动切片等技术，对系统的初始化流程进行优化，从系统框架层面对系统的资源进行合理化的分配，充分利用 CPU／IO 以及多核、多线程并行等能力，实现系统的快速启动。

5.3 资源抽象层

5.3.1 SoC 芯片平台的硬件资源抽象

SoC 芯片平台的硬件资源抽象对各种设备类型分别定义数据、控制和事件接口，不同硬件平台只需适配这层接口，以保证系统其他组件的可移植性，屏蔽硬件平台的差异。

SoC 芯片平台的硬件资源抽象具体包括摄像头硬件资源抽象、通用的车控管理系统和接口硬件资源抽象等。

a）支持摄像头的硬件资源抽象

支持不同的车载芯片硬件上的不同链接方式的摄像头控制与数据发送，包括适配摄像头硬件和根据应用需求建立摄像头数据通路两部分。

b）支持通用的车控管理系统和接口硬件资源抽象

支持电源抽象、亮度抽象、时间日期抽象、标定抽象等，提供相应的 API，基于此车控系统框架开发特定服务和功能可支持多车型平台。

5.3.2 整车信号资源抽象

架构中设置整车控制单元，支持对整车电子设备进行接口硬件抽象，应用层 API 接口标准化和 MCU 的 CAN 消息上报接口标准化。不同车型开发时只需要进行 MCU 软件适配，将不同的 CAN 报文统一到标准的 CAN 消息上

报接口，如空调的显示和控制、座椅状态的显示和调节控制。

5.3.3 外部 IoT 设备的资源抽象

车载操作系统应支持通过物模型实现外部 IoT 设备的资源抽象。物模型包括属性、服务、事件三种对象，按照 URI 协议为每个设备对象定义车联网内唯一标识符。物模型定义完成后，可通过物模型解析器自动生成物模型代码，由开发者根据接口添加具体功能，无需关心数据解析和封装。

- 属性：是对象的数据元素，描述对象的自身属性，例如图片的大小，颜色等，属性支持简单数据类型，也支持自定义的复杂类型；
- 服务：是对象提供的功能结构，提供对数据操控的能力，提供对物理设备的控制能力，服务是对象主动提供的能力，客户端可以获取对象的服务列表，例如相机的拍照能力；服务可以是协议标准提供的标准定义，也可以是产品对象自定义的功能集合；
- 事件：是对象需要外部被动感知的功能结构，事件由对象主动发起，事件描述当前对象的状态或者通知，客户端可以注册需要感知的时间列表；事件可以是协议标准提供的标准定义，也可以是产品对象自定义的功能集合。

5.3.4 外围 IC 的资源抽象

支持对蓝牙、GNSS（Global Navigation Satellite System，全球导航卫星系统）、收音等外围 IC 的硬件资源抽象，将功能调用接口标准化，通过外围 IC 驱动的支持，实现应用层对外围 IC 功能调用接口的标准化，更换了不同型号的外围 IC 设备，应用层软件接口调用无差别。

5.4 基础库

基础库为车载操作系统应用提供基础的通用功能，包括 kdbus，SQLite，Openssl，FreeType、MQTT 等提供最基础的通用功能，以及字体库、控件库等提供最基本的系统资源。

5.5 基础服务

车载操作系统提供的基础服务包括：

a) 互联服务

车载操作系统应支持通过 CAN、T-BoX、OBD、Tuner、V2X-OBU 等互联的服务。

b) 地图服务

车载操作系统应支持地图服务（位置服务），为包括地图在内的各类应用程序提供位置相关的 API。

c) 语音服务

车载操作系统应支持语音服务，其架构可分为语音形象、语音编程框架、语音服务模块、语音开放平台和语音自学习系统五部分。其中，语音形象提供交互展示型功能，并向上层的语音类应用提供 API。语音编程框架提供语音 API 支持基于 CAF 编写的语音应用（CloudApp）或车载小程序语音应用。语音服务模块提供本地语音语义能力和与云端能力对接，形成端云一体组件化可插拔的技术架构。语音开放平台支持第三方应用开发者自助式编写离线/在线的 NLU、DM、NLG 等技能（Skill），通过技能管理还可进行可视化编辑、管理等工作。语音自学习系统通过闭环优化系统，定期更新新的能力，提升用户体验。

d) 多媒体服务

车载操作系统应支持基础的图像、音频、视频的编解码播放和控制服务。

e) 云服务能力

车载操作系统应支持 OTA、账号、辅助驾驶、AI 等云服务能力。其中，OTA 通过网络自动检测到新版本（如新功能、安全补丁等）并下载安装，为新功能、安全补丁的发布提供升级通道。OTA 的架构由客户端与服务端两部分构成。客户端以系统级服务的方式运行在操作系统中，支持服务端检测系统版本、下载升级包、安装升级包等操作。服务端提供操作控制台，为配置管理员提供发布版本、上传升级包的操作功能，并向客户端提供版本检测查询服务、升级包下载服务、数据存储服务等。

5.6 运行时环境

5.6.1 应用管理

车载操作系统的应用管理包括生命周期管理、页(Page)管理、应用进程的创建、应用安全、资源管理机制等。

5.6.2 语言引擎

因为安全、开发高效等因素考量，程序运行框架和应用一般会采用托管语言来进行开发，语言引擎即负责支撑托管语言的执行。语言引擎需要满足性能高效（比如不能有长时间的 GC 卡顿，不能有长时间的引擎初始化动作、引擎本身不能有大量内存占用等）、安全（定期修复安全漏洞）、跨硬件平台以及向前兼容等要求。

5.6.3 小程序容器

小程序容器是AliOS、Linux、Android、iOS等不同系统的小程序运行环境（虚拟机），支持让开发者开发一份小程序代码可运行在不同系统、不同设备终端的能力。小程序容器满足不同小程序应用的安全要求（例如，支持支付类小程序应用的小程序容器应具备金融级安全要求）。

5.7 程序运行框架

5.7.1 互联互通框架

车载接入的设备包括以数据共享和多媒体互动为主要业务场景的 Wi-Fi 设备和以远程控制和低带宽数据共享为主的蓝牙设备。不同设备和不同设备型号具有不同的配置，有明显的功能差异，互联互通框架应按照不同的接口类型支持设备接入。接口类型包括设备发现，快速接入，服务发现，协议校验，语音控制，在线升级，流量代理等。

互联互通框架可分为设备配网层，网络层，设备抽象层和业务表示层。

- 设备配网层：提供基础的设备组网功能，如 Wi-Fi 快速接入，以及接入后的设备身份识别和认证。
- 网络层：对设备接入后的连接管理和抽象，应支持多种网络协议的设备，如 HTTP，MQTT。
- 适配层：为智能设备提供的 SDK 功能模块，用于支持以物模型定义的智能设备，如运动相机，行车记录仪等智能硬件的接入。
- 设备抽象层：以物模型为基础，对接入的智能设备的功能和对上提供的访问控制接口进行抽象，使得应用开发时不需要关心设备种类和型号的差异，通过统一的互联互通接口层进行数据的访问和共享。

- 业务表示层：为上层应用提供业务逻辑数据支持。

5.7.2 服务融合框架

服务融合框架是一种 SOA 框架，用于服务的注册、发现和调用，通过服务融合框架，不同种类的服务之间可以有机、灵活地组合构建成功能更加复杂强大的新服务或者应用。比如通过融合语音和机器视觉服务，来更加准确地识别用户的意图并提供更好的交互体验。

5.7.3 多媒体框架

多媒体框架提供音视频文件播放、音视频文件扫描和呈现、在线音视频，屏幕录制、跨 IVI 的媒体交互（如投屏到仪表、手机投屏音视频播放、车内行车记录仪实时视频、小程序多媒体等）、摄像等功能，包括播放器内核，音频输入输出与合成、预处理（如回声消除，滤波，去噪），后处理（如均衡、重低音、立体音效）等增强模块、A/V 编解码器等模块，支持 AC3，MP3，Dolby，DTS，MPEG2，MPEG4，H.264，H.265 等编解码技术。此外，多媒体框架可支持将编解码后的音视频内容由 App 封装和传输等，支持多媒体功能的定制化需求，其内部组件可复用，也可被不同的硬件平台和不同的应用程序复用。

5.7.4 多模态交互框架

因行车安全要求，面向移动设备的触控交互并不合适。多模态交互框架是一套以语音交互为主，以视觉、手势、AR 等交互为辅助的车载系统的交互框架，应用程序可以选择接入一种或者同时接入多种方式来提供交互。比如通过手势来接听或者挂断电话等。

5.7.5 场景感知理解框架

场景感知理解框架，是一套针对所接入的数据（比如车身数据、用户交互数据）等，并根据场景规则或者机器学习模型，来计算理解当前场景的一套框架，应用程序可以订阅场景感知理解框架的输出信号来执行动作或者推送服务。典型的场景理解结果比如是否发生了事故、是否在颠簸路面、是否违停禁入区域等。

5.7.6 UI 框架

UI 框架是程序运行框架的基础，主要提供各类窗口、丰富的界面元素和动画的编程接口。同时，UI 框架需要能支持提供 Surface 供三方渲染引擎来绘制界面。

5.8 车载操作系统安全

a）信息安全

车载操作系统应构建完整的信息安全防护机制，包括内核完整性保护、数据隐私及加密、安全隔离机制、安全存储机制、安全启动机制等，提供国密安全服务，降低漏洞对系统安全的危险，强化车载操作系统的安全防护能力。

b）功能安全

车载操作系统各功能模块的安全等级不同，有的有功能安全的需求（比如：仪表、车外后视镜、HUD），有的则没有（比如：IVI、RSE 等）。针对车载不同功能模块的不同功能安全要求，可选择不同的车载操作系统类型。

6 多系统架构和功能要求建议

6.1 硬件隔离架构

如图2所示，硬件隔离（Hardware Partition）架构将硬件资源通过硬件分区的方式进行划分和管理，硬件资源的所属分区拥有对该资源的访问和管理权限，其他分区不能对该资源进行操作。通过硬件分区的方式对资源进行管理，简化了资源从属和管理问题，方便了软件开发，但灵活性稍差。

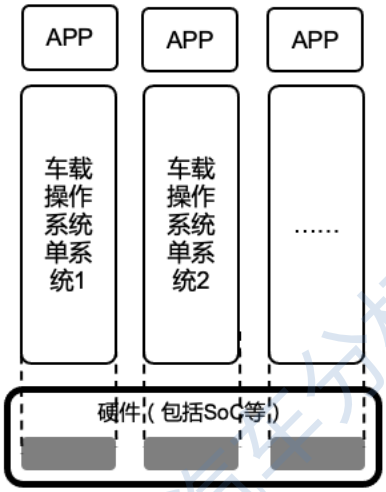

图2 硬件隔离架构

硬件隔离架构的典型技术方案是 ARM TrustZone。基于 ARM TrustZone 的隔离技术可将硬件分离出两个独立的 Domain:

- 关键应用 Domain: 可以在 1 秒内启动，主要用于运行仪表，倒车影像，提前输出声音和错误处理等关键应用；
- 多媒体应用 Domain: 主要用于运行非关键应用负载，如 Android 等 IVI 功能。

基于 TrustZone 技术，硬件被静态隔离，可预先分配硬件归属于关键应用域或非关键应用域，不需要 hypervisor 的支持，相关系统可以同时并行的访问硬件资源。这种方案的优点是可以消除 Hypervisor 引入的性能延迟，由于双系统间实现了物理隔离，多系统间相互影响小，稳定性及安全性在多系统方案中都达到最高标准，而且系统间交互通过底层模块保证，

对于系统应用的开发难度较低。该方案的缺点是，各系统对于硬件资源的需求在前期需要进行较为准确的规划设计，在分配后不可以动态运行规划，一旦一侧系统资源紧张不可以使用其它系统的空闲资源，灵活性较差，也容易导致资源利用率不足。

6.2 虚拟机管理器架构

虚拟机管理器（Hypervisor）架构基于虚拟机管理器为每个虚拟机（即车载操作系统单系统）分配不同的资源，可以协调不同芯片、不同芯片底层软件、不同应用层软件，使得硬件和软件资源可以按照产品需求，灵活地在不同的虚拟机操作系统中分配。虚拟机管理器是一种运行在硬件层和操作系统之间的中间软件层，通过限制或允许访问 CPU、内存和外设等片上资源来定义每个单系统可用的功能，而单系统之间完全隔离，某个单系统无法访问未提供给自身的资源。

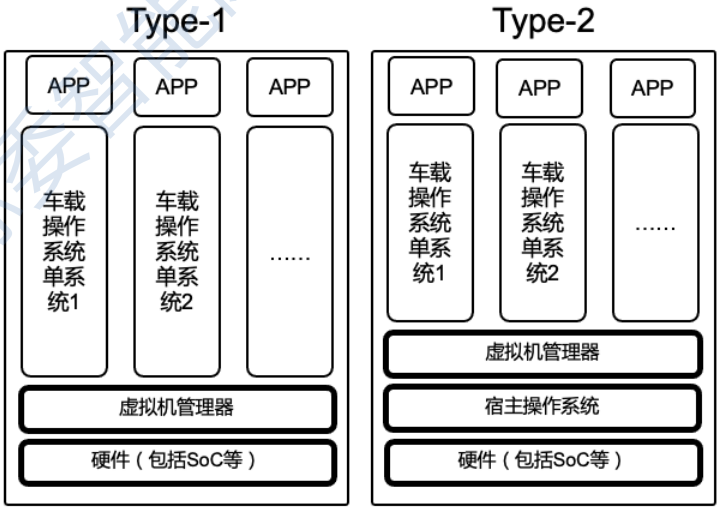

图3 虚拟机管理器架构

如图3所示，虚拟机管理器（Hypervisor）包括两种类型：

a）Type-1裸机型

裸机型 Hypervisor 最为常见，直接安装在硬件资源之上，操作系统安装并运行在 Hypervisor 之上。目前常见的虚拟机管理器包括黑莓的 QNX，英特尔主导的 ACRN，Open Synergy 的 COQOS、SYSGO 的 PikeOS、Elektrobit 公司的 EB Corbos、Mobica 为代表的 XEN、德国大陆汽车的 EB Corbos Hypervisor，法国 VOSyS 的 VOSySmonitor、三星哈曼 Redbend 的 Device Virtualization，其余还有如 Green Hills 的 Integrity，日本的 eSOL，Mentor 的 Nucleus，EPAM 的 Xen 等。

b) Type-2宿主型

宿主型（又称基于操作系统虚拟化）将虚拟化层安装在主机操作系统中，虚拟化软件以应用程序进程形式运行在主机操作系统中，而主机操作系统运行在物理硬件之上。

虚拟机管理器架构的正面影响包括：

- 充分隔离了车载应用的功能域，使得对功能安全要求较高的应用（如仪表程序）和非功能安全先关的应用可以运行在同一个 SoC 上，而彼此相互不受影响，从而节约了硬件资源，减少了功耗。
- 使多用户同时操作多屏幕成为可能。通过 Hypervisor 的虚拟化支持，可以在一个 SoC 上同时延展出多个 IVI 系统(Android 或 AGL)，系统之间彼此隔离，用户的操作不会彼此相互影响。
- 更高效的互联互通机制，通过 Hypervisor 的支持，多个虚拟机间可以利用主机内存作为媒介，相互间进行数据的交互和共享，这种通信速率和稳定性远高于基于网络的通信，使得多系统间的屏幕共享，大量数据共享访问成为可能。
- 更有效的硬件资源共享，默认情况下，所有的硬件资源都由特权虚

拟机进行管理，通过虚拟化的技术，多个虚拟机用户可以共享分配给特权虚拟机的硬件，这样就节约了硬件资源，不需要为每个系统都分配硬件，所有系统都可以通过虚拟通道访问虚拟硬件。

虚拟机管理器架构的负面影响包括：

- 增加了系统开发的复杂度。在车载环境中，由于 Hypervisor 的引入，通常需要维护 3 个虚拟机系统（Android、AGL 和 QNX），同时还要维护虚拟机之前的通信和相互间的协调关系，这样引入的开发工作量和难度相比单一操作系统要大很多。
- 增加了系统调度开销。由于虚拟机中需要对底层硬件的操作都需要嵌入到 Hypervisor 中进行，因此相关指令会导致虚拟机暂停并进入到 Hypervisor 中执行操作后再退回到虚拟机中继续执行，这样就增大了系统额外开销。
- 引入了安全性问题。当两个虚拟机间做内存共享时会引入安全相关问题，如果有某个虚拟机安全性不足被非法入侵后，可能导致非法入侵者通过安全漏洞进入其他虚拟机甚至 Hypervisor 中。其中 Type-1 裸机型方案由于不存在底层支撑的宿主操作系统内核,对于上层虚拟机的攻击需要个个击破，安全性相对较高，而 Type-2+Application 的方案，其 Application 层直接搭建在宿主操作系统之上，因此更容易直接被攻破，造成整个多系统架构的崩溃，在虚拟机监控器架构方案中安全性最差。

对于 Hypervisor 环境引入的负面影响可以通过优化的手段来降低，常用的优化手段有：

- 减小虚拟 cpu 退出到 Hypervisor 的次数；

- 通过将相关硬件直接 pass 到虚拟机中进行操作，减小对 Hypervisor 的依赖；
- 对GPU采用硬件虚拟化技术针对多虚拟机进行隔离,减小软件开销。

总的来说，Hypervisor 环境的引入对车载系统是利大于弊，能够更充分的发挥车载 Soc 的性能，更好的满足多人多屏的需求。

6.3 容器架构

如图 4 所示，容器（Container）架构基于 Linux 内核和容器框架承载多个应用程序，这些承载应用程序的容器共享底层计算机和操作系统，Linux 内核确保运行在不同容器中的应用程序互相隔离，具有不同的文件系统、网络和存储接口、处理器和内存。容器映像中的文件用作从该映像创建的容器的只读文件系统，在容器中运行的程序看不到其他文件，且在容器内运行的代码的行为独立于基础系统。

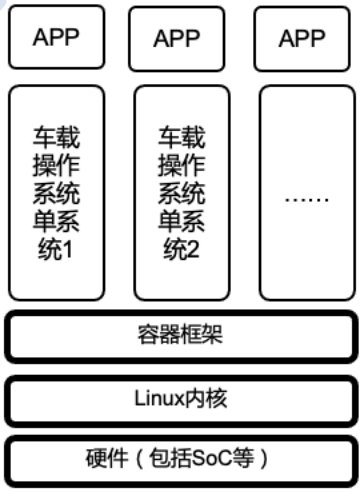

图4 容器架构

容器的典型架构包括 runC 方案和 LXC 方案。

a) runC容器架构

runC 容器架构如图 5 所示，runC 是一个容器运行环境，最初是作为 Docker 的一部分开发的，后来被提取为一个单独的开源工具和库。 作为“低级”容器运行时，runC 可以用作独立工具，用来生成和运行容器。像 Docker 这样的“高级”容器运行时通常将实现诸如映像创建和管理之类的功能，并将使用 runC 来处理与正在运行的容器相关的任务-创建容器，将进程附加到现有容器（docker exec）等。EB corbos Linux 包含 runC 的容器方案，该方案已经在大众 MEB 平台量产。

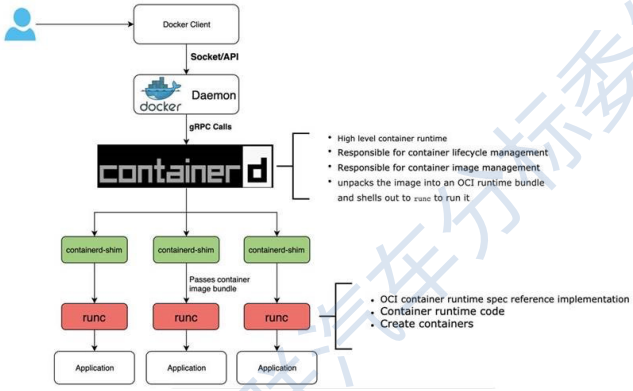

图5 runC容器架构

runC 相关的标准化主要在 Open Container Initiative (OCI)开展。OCI 是一个开源组织，由 Docker 和其他容器行业的领导者于 2015 年 6 月建立，其明确目的是创建容器格式和运行规范的行业标准。OCI 目前包含两个规范：运行时规范（runtime-spec）和映像规范（image-spec）。运行时规范概述了如何运行在磁盘上解压缩的 “文件系统包”。在较高级别上，OCI 实现将下载 OCI 映像，然后将该映像解压缩到 OCI 运行时文件系统捆绑包中。此时，OCI 运行时捆绑包将由 OCI 运行时运行。

b) LXC容器架构

LXC（Linux Container）可提供轻量级的虚拟化，以便隔离进程和资源。LXC 可以在操作系统层次上为进程提供的虚拟的执行环境，一个虚拟

的执行环境就是一个容器。LXC 在资源管理方面依赖与 Linux 内核的 cgroups 子系统，cgroups 子系统是 Linux 内核提供的一个基于进程组的资源管理的框架，可以为特定的进程组限定可以使用的资源。

容器架构的特点包括：

- 轻量级隔离，与宿主机使用同一个内核；
- 无需指令级模拟；
- 无需即时(Just-In-Time)编译；
- 容器可在 CPU 核心的本地运行指令，无需任何专门的解释机制；
- 提供共享机制，可简单实现容器与宿主机的资源共享。

6.4 混合架构

在实际应用中，为平衡功能与安全性的不同要求，可采用三类基础架构中的两类或三类组合而成的混合架构，例如，硬件隔离+虚拟机管理器 Type-1 的混合架构（如图 6 所示）。

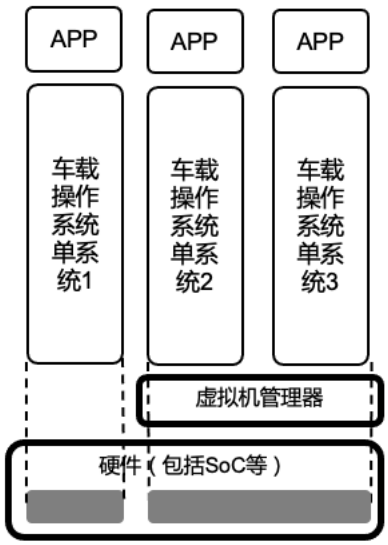

图6 硬件隔离+虚拟机管理器Type-1的混合架构

此外，如图7所示，有些虚拟机管理器在Type2的基础上，将虚拟机管理器实现在一个宿主操作系统之上，但该操作系统既可以支持硬件虚拟化及其它单系统运行在其虚拟机管理器之上，又可以运行自身操作系统的功能和应用，并实现与其它单系统间的隔离。

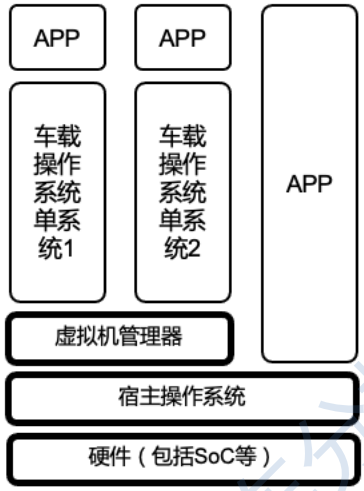

图7 虚拟机管理器Type-2 + Application 混合架构

7 总结和标准化项目建议

基于对车载操作系统架构现状和发展趋势的研究，本报告提出了车载操作系统架构可分为单系统架构和多系统架构，并提出了相应的架构和功能要求。

车载操作系统单系统架构仅涉及单个车载操作系统的架构，范围介于应用程序和硬件抽象层之间，由车载操作系统内核、基础库、基础服务、运行环境及程序运行框架组成。

车载操作系统多系统架构支持多个车载操作系统运行于同一套硬件，每个车载操作系统的架构符合单系统架构。多系统架构分为硬件隔离、虚拟机管理器（Hypervisor）、容器三种方案，用于区隔不同安全等级的系统。

基于上述研究成果，标准化项目建议及必要性分析如表 2 所示。

表 2 架构和要求类标准化建议

标准化项目建议	必要性	启动时间建议
补充《车用操作系统术语定义》	通用的术语定义是标准化的前提，建议在通用标准中补充车载操作系统相关的术语和定义。	优先级高
车载操作系统参考架构	参考架构标准的制定，为车载系统的技术创新、标准研制、试验验证、应用实践、产业生态等提供参考和指导依据。建议启动国推标制定计划。	优先级高

附 录 A
（资料性附录）

车载操作系统简介

A.1 QNX 车载操作系统

QNX 由黑莓公司推出，最初应用在仪表，是第一个符合 ISO 26262 ASIL D 规范的实时操作系统，能满足数字化仪表盘功能性安全的要求，同时兼满足数据安全要求，通过美国军方 EAL4+。因为 QNX 并不开源，针对 QNX 开发需要支付一定金额的商业授权费用。

A.2 Linux 车载操作系统

Linux 是基于社区开源模式的多用户、多任务、支持多线程和多 CPU 的分时操作系统，因为 Linux 是在 GNU 公共许可权限下免费获得的，无需商业授权费用，用户可自主定制，开发成本相对较低。正因为开源，Linux 系统开发工具非常丰富，几乎所有应用芯片都支持 Linux。Linux 系统上的程序是编译执行的，执行效率比较高。

但是，Linux 内核复杂度较高，Linux V4.19 的内核代码有一千七百多万行，且还在持续增长。正因如此，在安全性和启动速度等方面，Linux 要落后于 QNX。针对汽车的应用场景，AGL（Automotive Grade Linux)和 GENIVI 都在极力打造 Linux 生态系统,以加速 Linux 在车载系统上的应用。

Linux 是分时操作系统,实时性较差。基于 Linux 改进的 RT-Linux 可满足实时性的要求，技术方案包括 RT-Linux 双内核方案和 RT-Preempt Patch 等。其中， RT-Preempt Patch 方案对 Linuex 的主要

修改如下： 1）采用 RT-Linux 的 patch 包修改 Linux 的进程/任务调度策略；2）修改 Linux 内核里耗时比较多的 driver 等模块。

A.3 AliOS 车载操作系统

AliOS 车载操作系统基于 Linux 内核，具备自主知识产权，支持多个类型的端设备，除了具备最基础的进程、线程调度等功能外，还支持定位和地图、辅助驾驶、轻量化的车载语音智能等服务，可连接 AIoT 设备 2.72 亿+，提供 5200+实用 AI 技能和 3 亿+生态资源，为智能汽车、数字交通和产业数字化三类解决方案提供技术基础。AliOS 车载操作系统架构主要包括 AliOS 核心系统、数据服务平台以及工程化支持平台三大部分。AliOS 核心系统主要实现 Linux 内核、设备驱动、内核安全、多核实时调度、硬件抽象、基础库、图形处理、多媒体框架、电源管理、网络连接、安全管理、云应用管理和运行时环境等功能；数据服务平台提供基础服务、核心云服务以及车云互联的服务；工程化支持平台基于开放的车辆服务融合平台和云测平台，提供开发、测试、服务接入为一体的车载应用和服务生态。

A.4 Android 车载操作系统

Android 车载操作系统基于 Linux 内核，主要功能模块集中在用户空间，在 Linux 内核中仅增加了用于进程间通信的 Binder 和用于内存共享的 ION，其突出优点是应用生态。Android 应用程序（Java 代码）在 Android 虚拟机里是解释执行的，相对于编译执行来说，执行效率较低。Google 于 2019 年开放了 Android Automotive，原本为移动互联设备开发的 Android 应用生态可迁移

到 Android Automotive。目前，Tier1 和车厂基于 Android 自研车载操作系统。

A.5 鸿蒙 OS

鸿蒙是基于微内核的分布式 OS，目标是提升操作系统的跨平台能力，包括支持全场景、跨多设备和平台、低时延和高安全性。鸿蒙系统包括三层架构：第一层是内核，第二层是基础服务，第三层是程序框架。鸿蒙车载操作系统 Harmony 是通过一芯多屏、多并发、运行时确定保障等能力，满足出行场景需要，具有“多用户，多外设，多联接”的特点，定义了 HMS-A（HMS for Auto，包括：语音、音效、视觉、AI 等 7 大核心能力）、12 个车机子系统和 500 多个 HOS-C API，支撑 OEM、合作伙伴、第三方应用快速开发、持续升级。

A.6 用于仪表盘的车载操作系统

用于仪表盘的车载操作系统基本分为三种：

a) 以 ucOS 或 iTron 为代表的 RTOS

该系统适用于低端仪表，配合简单的点阵或段码显示屏使用，功能单一。

b) 以 Linux 做为操作系统

该系统在 QT 等图像库支持下可以实现对于全液晶仪表的支持，因为 Linux 的开放性，开发成本较低。

c) 以 QNX 做为操作系统

该系统适合全液晶仪表的开发，需要商业授权，开发成本较高。

A.7 用于 T-box 的车载操作系统

用于 T-box 的车载操作系统可分为两种：

a) 以 ucOS 或 iTron 为代表的 RTOS

该系统实时性好，代码量小，对硬件算力要求低，适合于硬件算力不高且功能单一的低成本低端 T-box。

b) 以 Linux 做为操作系统

该系统有完整的 TCP/IP 网络支持，适合于集成功能较多硬件算力较强的 T-box。

附 录 B
（资料性附录）

不同车载操作系统支持的单系统架构模块

功能模块	Linux	AGL	RT-Linux	QNX	AliOS	AliOS RT	Android	鸿蒙
内核	Linux	Linux	Linux	QNX	Linux	自研	Linux	Linux+ 自研
资源抽象层	具备资源抽象功能	具备资源抽象功能	具备资源抽象功能	具备资源抽象功能	有	有	Android HAL	有（Harmony OS Driver Foundation）
基础库	无	有	无	有	有	有	有	有
基础服务	无	有	无	有	有	有	有	有
运行时环境	无	有	无	有	有	有	有	有
程序运行框架	无	有	无	有	有	有	有	有