RTOS 与 Linux 共同定义车载计算

随着行业从功能特定的硬件和软件转向集中式车载计算和软件定义车辆，Linux 显然将在未来的车辆平台中发挥重要作用。鉴于在其他行业中久经验证并且极具灵活性，Linux 也自然成为汽车应用的选择。

然而，随着 OEM 厂商引入具有特殊需求的应用（特别是在全自动或半自动驾驶的功能安全方面，功能安全场景中需要采用具有优化的确定性延迟的实时操作系统（RTOS）。

在评估潜在的未来车载软件架构时，了解 Linux 和（RTOS 各自的相对优势以及它们如何在不断变化的环境中互补是非常重要的。

主要差异

并非所有应用都有相同的时延要求，而应用之间的差异需要操作系统采取不同的方法来提供支持。

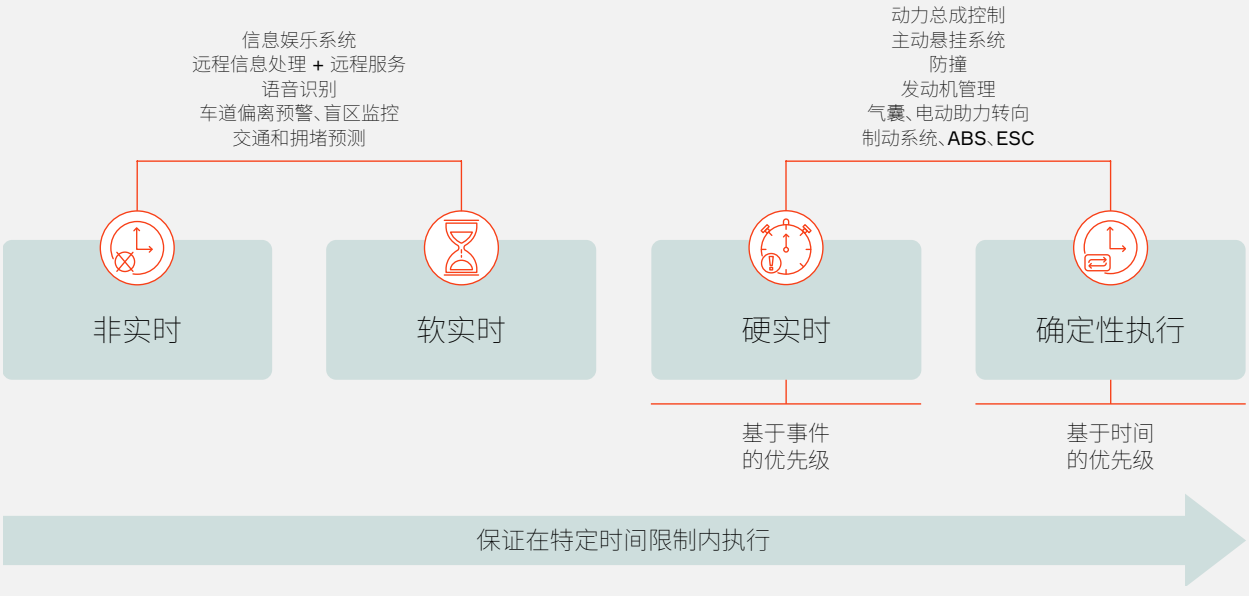
有些应用不需要保证在特定时间范围内完成任务。这类应用应尽可能快地运行，但可以接受轻微的延迟。在汽车领域，这类应用包括远程信息处理、远程服务和交通预测。这些是非实时（非 RT）应用。

对于一些应用来说，延迟可能会导致用户体验不佳。在汽车领域，这类应用包括高级驾驶辅助系统的一些功能，例如车道偏离预警和盲区监控。它们提供重要的及时信息，但不需要驾驶员立即做出响应。操作系统需要实现任务调度，但不需要保证可预测的性能或保护任务免受其他应用的干扰。这些应用被视为“软”实时应用。

但对于另一些应用来说，延迟则可能会造成严重的后果。例如控制车辆运动的软件功能，包括自动紧急制动、变道、防碰撞自动操控等。这些被称为“硬”实时应用，它们通常对安全至关重要。

什么是实时？

实时系统可以保证任务在确定的时间内一致地执行。确定性是用于描述系统在时间限制内执行任务的一致性的特性。在确定性系统中，任务的执行顺序始终相同。



竞争操作系统

作为一个通用操作系统，未经修改的 Linux 仅适用于可以容忍延迟的应用。

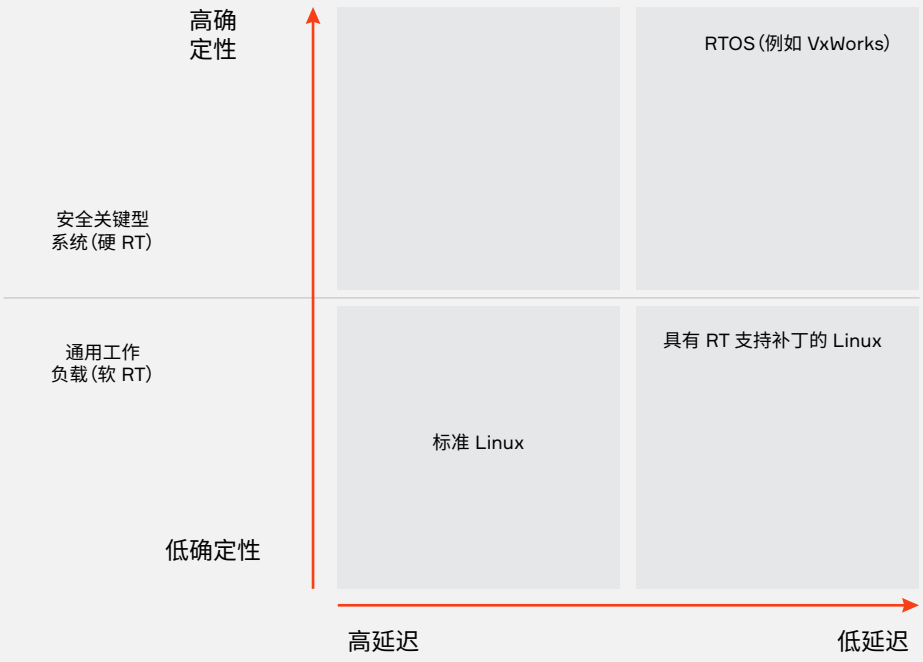
Linux 之所以能在企业和行业中广泛应用，部分原因在于其成本低廉，并且拥有庞大的开源开发人员和工具生态系统。OEM 已经在各种车辆应用中广泛采用或评估了 Linux，并证明 Linux 非常适合某些应用。

相比之下，RTOS 专为实时应用而设计，可保证给定的输入每次都能按时产生相同的结果。RTOS 使实时应用能够在特定时间保留操作系统资源，并使用这些资源按照严格的时间限制完成任务。它确保事件在正确的时间发生，而不仅仅是尽快发生。这些特性对于硬实时应用来说不可或缺。

此外，RTOS 的代码量通常比通用操作系统少。这为 OEM 带来了诸多优势。从安全角度来看，源代码越少，攻击者可利用的攻击面越小，从而降低了风险并减少了缓解风险所需的工作量。也就是说，使用具有安全开发生命周期的RTOS仍然很重要。在安全性方面，较小且较简单的操作系统更容易获得功能安全

选择合适的工具

不同的操作系统适用于车辆内的不同应用。



标准认证。这些优势可以帮助 OEM 更快地将车辆推向市场。

在 RTOS 上运行实时负载还可以降低成本。由于源代码较少，RTOS 对 CPU 和内存的要求也较低，因此需要的硬件物料更少。如果 RTOS 已预先通过安全认证，那么 OEM 就可以节省自行认证的大量成本。此外，RTOS（尤其是其源代码和 API）变更较慢，降低了维护和安全成本，保护了 OEM 的投资，并减少了因变更而重新认证操作系统的需求。

针对软实时应用修改 LINUX

尽管 RTOS 无疑是硬实时应用的明确选择，但现在借助一些 Linux 补丁，也可以实现在多个 Linux 版本上运行软实时应用。

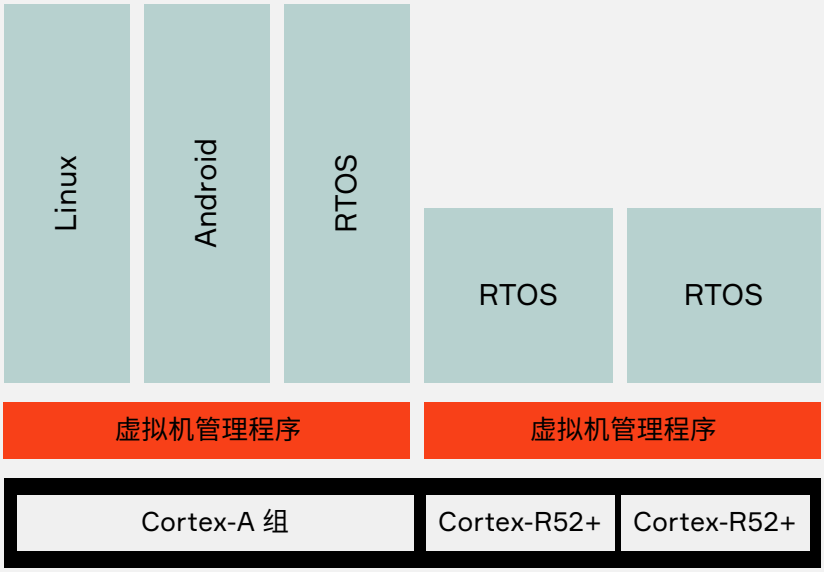
这些补丁允许 OEM 将一些实时功能从 RTOS 转移到 Linux，但这样做可能会在某些方面影响性能和可管理性，具体包括以下几点：

定时器松弛

Linux内核的补丁通过更频繁地唤醒CPU来提高内核计时器的分辨率。 精度可达纳秒级，这是一些实时应用线程所需要的。 然而，更频繁地唤醒 CPU 可能会增加功耗并降低 CPU 效率。该补丁还可以在实时线程中实现更精确的计时，但它是通过使用定时器松弛来实现的，这是一种使用正常计时策略延迟线程中事件的技术。这可能会导致异步应用中出现更多抖动。

混合搭配

各种操作系统可以在同一硬件上并存，但应通过虚拟机管理程序在逻辑上隔离，并根据需要在不同的内核上运行。



进程挂起

由于实时系统被设计为在截止时间内执行当前优先级最高的任务，因此通常需要使用自定义调度方法，而不是 Linux 内核的标准调度方法。如果 RT 进程试图占用 100% 的 CPU 资源，则内核的标准调度方法会每秒将该进程挂起 50 毫秒。用于 RT 支持的补丁会停用该挂起机制，但该功能对于内核的调度系统至关重要，移除它可能会导致系统不稳定或出现故障。

业界不断致力于提升 Linux 在安全关键型应用中的适用性，包括推出了名为“Enabling Linux in Safety Applications (ELISA)”的项目，该项目由成员公司、认证机构和标准化组织合作，旨在确定如何将 Linux 用于安全关键型系统中。

RTOS 与 LINUX 如何共存

通过在 RTOS 上运行所有硬实时车辆应用并将 Linux 专用于非实时和软实时工作负载，OEM 可以在尽享 RTOS 固有优势的同时，更大限度地降低 Linux 的复杂性。 这样可以降低软件开发和维护成本并提高安全性。

集中车载处理为 OEM 提供了并行运行 Linux 和 RTOS 的理想机会。共享计算平台可以通过使用两种关键技术来托管多个运行不同关键级别应用程序的操作系统： 通过虚拟机管理程序实现软件虚拟化，以及在单独的处理器内核上实现硬件隔离。

软件虚拟化

软实时应用可以在虚拟化系统中与 Linux 和其他操作系统在同一组内核上运行，从而可以灵活使用资源。 （请参见侧栏中的示例。）每个操作系统都在自己的虚拟机中运行，并由虚拟机管理程序（例如风河的 Helix 虚拟化平台）管理一组 ARM Cortex-A 高性能 内核的通用内存、计算资源和处理器内核。在应用中，容器化可以提供进一步的虚拟化层，以实现更灵活的开发和维护。

用例：自动紧急制动

Linux 和 RTOS 在车辆中协同工作的一个示例是实现自动紧急制动（硬 RT 应用）。

当车辆接近物体时，其摄像头、雷达、激光雷达和惯性传感器会将数据发送到区域控制器，区域控制器再将数据转发给中央车辆控制器。设备上运行的 Linux 软件使用计算机视觉和机器学习算法来融合传感器输入并将其置于上下文中。

Linux 软件将这些数据发送到制动调节器（一个连接到制动系统的小型应用）。制动调节器运行在经过安全认证的 RTOS 上，以确保它能够在精确的时间点实施制动。通过对与上下文关联的传感器数据应用算法，制动调节器不断计算发生碰撞的风险，并在必要时实施制动。

硬件隔离

在 RTOS、虚拟机管理程序和内核均支持 ASIL-D 风险级别的前提下，安全关键型应用可以在一个或多个 RTOS 上运行，并可以与非安全关键型应用在同一个SoC上共存，但要使用 Cortex-R 内核在“安全岛”上隔离。 安全岛可以包含通过虚拟机管理程序虚拟化的多个操作系统。 安全岛上的隔离可防止不太关键的应用程序中的故障导致影响生命安全的应用程序崩溃。

随着硬件架构的发展而演变

这些配置可以在当前和未来的车辆架构中实现，不过在域架构、区域架构或两者组合的架构中，集中化程度各不相同。在所有情况下，中间件都是实现车辆中所有容器、应用程序和操作系统之间通信和协作的重要粘合剂。多个操作系统将在一个软件栈下的多个内核上运行，内核根据需要分配给不同的操作系统。

拥抱无限可能

从专用电子控制单元和软件向软件定义车辆转变的趋势提供了在车辆中引入多种操作系统（包括 Linux）的可能性。Linux 具有高性能和庞大的开发生态系统，RTOS 具有确定性并经过安全认证，两者相辅相成。了解每种操作系统的优缺点后，OEM 可以在集中式硬件上以前所未有的灵活性部署这些操作系统，从而改进未来的车辆，并使开发过程更快、更具成本效益。

2022 年，安波福收购了风河。风河专注于提供用于支持稳健、可靠和安全的嵌入式解决方案的工具，包括 Wind River Linux 和 Wind River VxWorks RTOS。
两者均为汽车应用提供了独特的优势，并且通过长久在其他行业中的安全关键操作方面提供支持而汲取了丰富的经验。

作者简介

Rob Woolley
风河首席技术专家

Rob Woolley 是风河 CTO 办公室的首席技术专家。他在企业和嵌入式 Linux 方面拥有 25 年的经验，在 VxWorks RTOS 方面拥有超过 15 年的经验。他积极参与开源社区，负责维护 OpenEmbedded 机器人操作系统框架，并参与 Linux 基金会的 Zephyr RTOS 和 ELISA 项目。他目前专注于研究使用云原生技术来编排边缘设备上的工作负载，包括软件定义车辆。

详情请见 APTIV.COM/MWD →