Xiaomi HyperOS

小米澎湃OS

技术白皮书 V1.0

序言

2023 年对于小米而言，是开启全新发展周期的一年，也是一系列的深刻变革进入深水区之时。集团正处于一个全新殿堂的门槛上，全新阶段的起跑线上。这一年我们的关键词是“稳健推进，蓄势待发”，这一年里，我们要不断夯实基础、构建强化体系能力，为未来 5-10 年的长期发展打下坚实基础 ，并为未来 2-3 年的新一轮爆发做好充分的准备。

（雷军 2023/01/30）

从小米创办之初，我们就开始研发 MIUI。
可能不少朋友并不了解，MIUI 不只是基于安卓原生系统交互层改进，而是早已深入系统底层的 Linux 内核，在系统框架、性能调度、内核能力做了大量的底层“魔改”。比如在 2013 年，我们推出了进程对齐唤醒机制，领先安卓半年，为保护用户隐私推出的运行时权限管理，更是领先了安卓近三年。

随着 IoT 业务率先启动，小米对嵌入式系统的研发也越发深入。为了解决 IoT 网络中设备系统的碎片化和连接隔阂，2014 年，小米就发布了 IoT 设备的连接协议。2017 年，小米自研的 Vela OS 正式发布。为了弹性适应各类智能硬件产品，Vela OS 做到了可伸缩、可裁切，支持从简单到复杂的各类设备，开始逐步统一 IoT 设备生态，目前已经装机量已达 2000 万台以上。

2016 年，小米就开始研发跨端应用框架。2019 年，我们开始并行研发纯自研通用系统 Mina OS，并在部分产品上小规模量产验证，同时在实验室中也成功在手机上跑通，其中部分技术成果也已融入小米澎湃 OS。
2021 年，小米开启了车机 OS 的研发。

2022 年初，我们决定统一 MIUI、Vela、Mina、车机 OS 四个系统的软件架构。自此，小米的操作系统底层合并完成。

超过 5000 人研发团队，又经过两年打磨，小米澎湃 OS 应运而生，「人车家全生态」由此开启。

（雷军 2023/10/23）

序言

小米操作系统大事记

2010 年　MIUI 诞生

2014 年　统一的 IoT 设备连接协议、通用 IoT 模组发布

2016 年　跨端快应用框架启用

2017 年　自研 Vela OS，逐步统一 IoT 设备系统

2019 年　预研 Mina OS；自研微内核安全系统启动

2020 年　跨端互联互通协议融合

2021 年　统一技术架构设计；小米汽车 OS 启动

2022 年　统一软件架构，系统研发并线完成

序言

在全球技术进步与数字化转型的驱动下，我们正在见证一个技术领域内的巨大变革。随着互联网技术的普及，物联网设备的广泛应用以及大数据与 AI 的结合，人类的生活和工作方式正发生着翻天覆地的变化。这不仅仅是一场关于技术的革命，它还在社会、经济和文化等多个领域带来了深刻的影响。

在这样的背景下，传统的、基于单一设备的计算模式显得过于陈旧，难以适应日益复杂的技术需求。现代社会的技术环境已经超越了单一设备，形成了一个由人、机器和各种物体组成的互动网络。在这种环境中，我们需要一个全新的、更为先进和全面的操作系统来进行管理。

于是，小米澎湃 OS 应运而生。它不仅要满足不同设备间的连接与管理需求，还要保证系统交互的流畅性、易用性和安全性。小米澎湃 OS 可以搭载在手机、汽车、智能家居等 200 多个品类上，最终构建「人车家」一体的全生态平台。

同时，随着 AI 技术的逐步成熟，它正在改变软件的传统角色。软件不再只是一套指令，而是成为了支持和驱动各种创新应用的基础。因此，软件开发和管理的方法也需要随之变革，以适应这个新的技术环境。

面对这一系列的挑战，小米集团站在了最前端，决心为社会提供最前沿的技术解决方案。小米软件团队在技术研究上投入了大量的资源，他们的研究范围涵盖了从系统内核、系统服务、跨端互融、开发生态的各个方面。

在这个既充满机遇又充满挑战的技术趋势下，我们需要不断地创新和进步。而小米澎湃 OS 无疑是小米在这个万物互联的时代，为用户和开发者提供的一份最优质的答案。

2 技术架构

2.1 技术架构简介

我们正式公布了小米澎湃 OS 完整系统架构。

这个架构图背后，凝聚着小米创业 13 年来，数千名工程师研发探索的心血。从 2010 年 MIUI 诞生起，小米在 OS 方面的研发探索，覆盖了手机操作系统、嵌入式操作系统、纯自研通用操作系统、车机系统等全部领域，最终在小米澎湃 OS 上并线合流，如今拥有一支 5000 多名工程师的操作系统研发团队。可以说，小米的创业史，就是一段操作系统研发的探索史。

小米澎湃 OS 是一个超级庞大的工程体系。从架构设计之初，我们就明确了四个目标：第一，实现单端性能表现最强 ；第二，AI 赋能，成为整个生态的“智能大脑”，能够为用户提供主动服务；第三，更加便捷高效的连接；第四，实现全端隐私安全坚固防护。

在最底层的系统内核层，我们将自研的 Vela 系统内核与深度修改的 Linux 系统内核进行融合，重构了性能调度、任务管理、内存管理、文件管理等各个基础模块，实现了性能、效率的大幅提升。这一全新的融合内核，支持 200 多个处理器平台、20 多种文件系统，还能根据硬件能力差异灵活配置，具有很好的兼容性，使得每个独立设备的性能都能得到彻底解放。

在系统内核层之上的服务与框架层，我们将安卓的服务框架和自研 Vela 系统的服务框架，都作为“中间件”纳入其中；同时，全新打造了 8 大子系统，其中全新的 AI 子系统融合大模型能力，成为整个系统的“智能大脑”，不仅可以让单设备实现极强的端侧 AI 能力，同时赋予整个生态智能能力。

最上层 HyperConnect 跨端层，则是彻底打破了硬件设备的隔阂，让所有设备可以统一连接协议，并且实时通信。让整个小米澎湃 OS 生态打通“任督二脉”，每个设备如同感知世界的触角，像一张巨大的网一样连接一起，最终构建「人车家全生态」的智能世界。特别值得一提的是，这次我们打造了贯穿内核层、服务框架层、跨端层的全端安全系统，尤其是内核层，我们启用了完全独立的「自研微内核安全系统」，保障了安全从最底层实现。

（雷军 2023/10/23）

小米澎湃 OS 是一个为下一代的计算与多种设备生态设计的系统。主要针对以下关键挑战提供了解决方案：

- **设备多样性**
  - 在当前的数字世界中，各种操作系统和技术标准使设备高度分散。小米澎湃 OS 旨在提供一个一致性的平台，使从手机到 IoT 设备的所有设备能更好地统一与集成。
- **设备间的交互**
  - 大多数传统操作系统并未为多设备互动进行设计。而小米澎湃 OS 注重设备间的流畅交互，以确保高效的跨设备合作。
- **开发流程的简化**
  - 通过小米澎湃 OS，开发者能为不同设备编写统一的代码，消除了开发跨设备应用的复杂性。
- **增强安全保障**
  - 小米澎湃 OS 打造高安全的 TEE OS（EAL 5+）和其他安全特性，以减少潜在风险并提高系统整体的安全水平。
- **高效的性能调配**
  - 小米澎湃 OS 在设计时就已考虑到各类设备的性能和资源需求，尤其是在资源有限的场景中。
- **鼓励生态发展**
  - 为了吸引更多的开发者参与并促进生态的建设，小米澎湃 OS 的核心组件选择了开源的道路。

小米澎湃 OS 针对设备的多样性、设备间的交互、开发流程、系统安全、性能调配和生态发展等挑战提供了综合的解决方案。

2.2 技术架构图

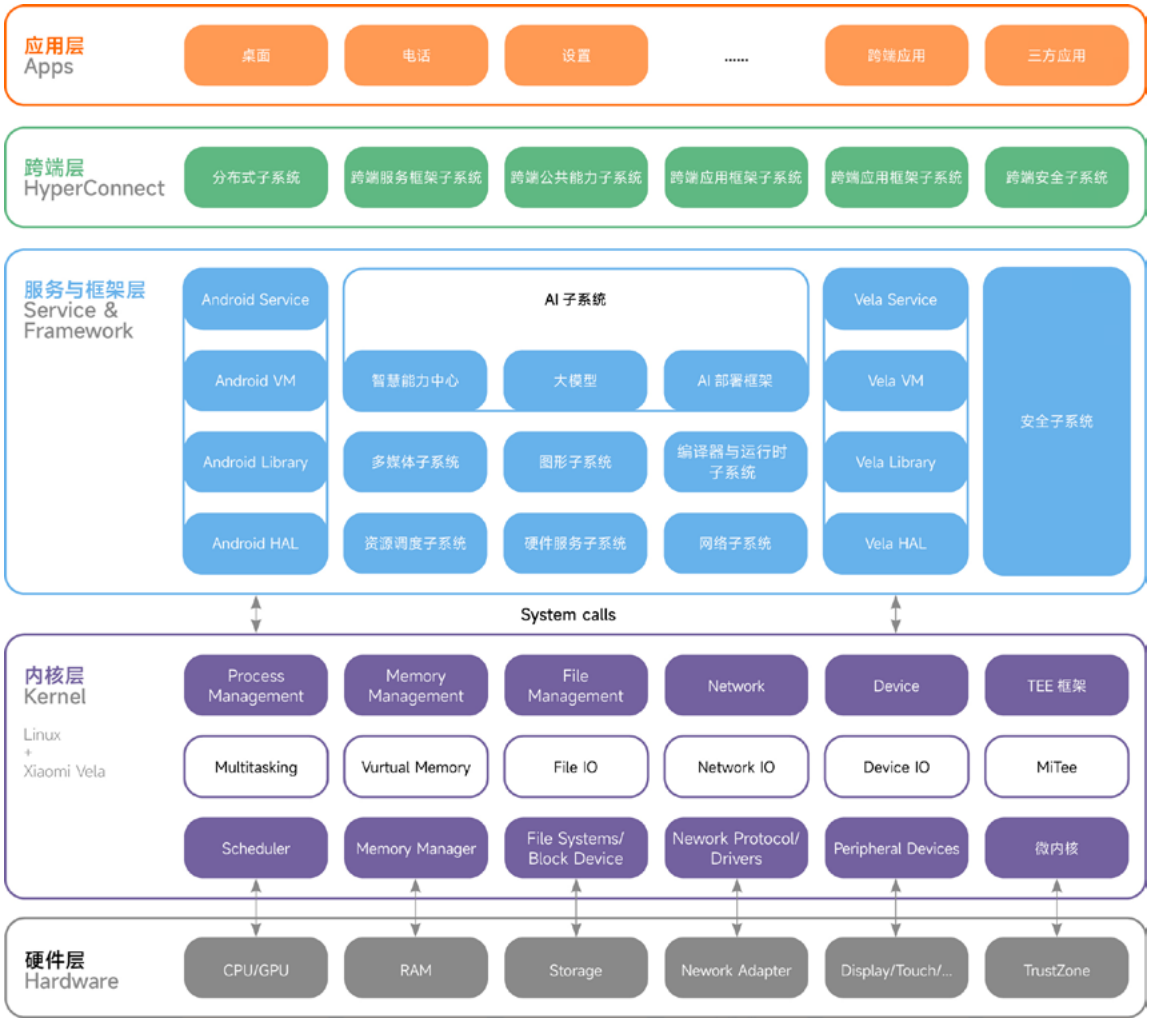


图 2.2-1 小米澎湃 OS 技术架构

2.3 核心组件

内核层

- Linux 内核：融合内核的标准化组件集，适用重载硬件资源的抽象与管理。
- Vela 内核：融合内核的轻量化组件集，适用轻量硬件资源的抽象与管理。

系统服务层：

- 资源调度子系统：负责合理、高效地分配和管理系统资源，以确保应用和任务能够流畅运行。
- 网络子系统：旨在支持多设备、分布式、安全和高性能的网络通信，以满足不同设备和应用场景的通信需求。

- 多媒体子系统：提供了处理多媒体内容（如音频、视频和图像）以及与多媒体硬件交互的功能和服务。
- 图形子系统：负责管理和控制图形显示以及用户界面（UI）渲染，高性能和响应性的用户界面。
- 硬件服务子系统：负责管理和控制硬件资源以及提供硬件访问和互动的功能和服务，提供一致的硬件访问接口。
- AI 子系统：集成人工智能（AI）和机器学习（ML）技术，以提供更丰富、更智能、更个性化的体验。
- 安全子系统：满足不同设备和应用场景的安全需求而设计，提供全面的安全性和隐私保护，确保用户设备和数据的安全。
- 编译器与运行时：提供跨设备统一的编译器与运行时，确保应用的运行性能，同时保障跨端运行的安全性。

跨端层：

- 分布式子系统：支持多设备之间的协同工作和数据共享，以提供无缝的分布式计算和体验，从而提高用户体验和便利性。
- 跨端服务框架子系统：为应用程序提供一套丰富的服务和工具，以增强应用程序的功能和性能，提供了一个多元化的应用程序服务平台。
- 跨端公共能力子系统：提供了标准化的开发工具和库，以支持开发者构建各种应用程序和服务，有助于提高应用程序的可移植性和一致性，使开发者能够更轻松地开发和维护跨平台的应用程序。
- 跨端应用框架子系统：提供一种轻量级的应用程序模型，旨在提供更快、更流畅的应用体验，并减少了应用程序的安装和占用存储空间的需求。
- 跨端安全子系统：专注于提供安全性和隐私保护，以确保跨不同端设备之间的通信和数据共享的安全性。

开发者服务：

- 开发平台：开发平台是一套工具和资源，旨在帮助开发者创建应用程序和服务，以运行在小米澎湃 OS 操作系统上。
- 分发体系：小米澎湃 OS 的分发体系是一个多层次的策略，旨在将该操作系统部署到各种不同类型的设备上，提供多样性的应用程序和服务，以满足不同市场和用户需求。

3 跨端层（HyperConnect）

3.1 分布式子系统

3.1.1 子系统简介

分布式子系统是通过近场通信的跨设备互联协议，在跨平台基础互联的基础上构建跨端分布式服务框架。通过核心服务的分布式能力实现应用生态融合，影音互联，文件系统，硬件协同，硬件控制等领域的纵向和横向扩展，并且为上层应用提供基础的安全通信和同账号系统服务，助力跨设备功能的持续演进。

3.1.2 子系统架构

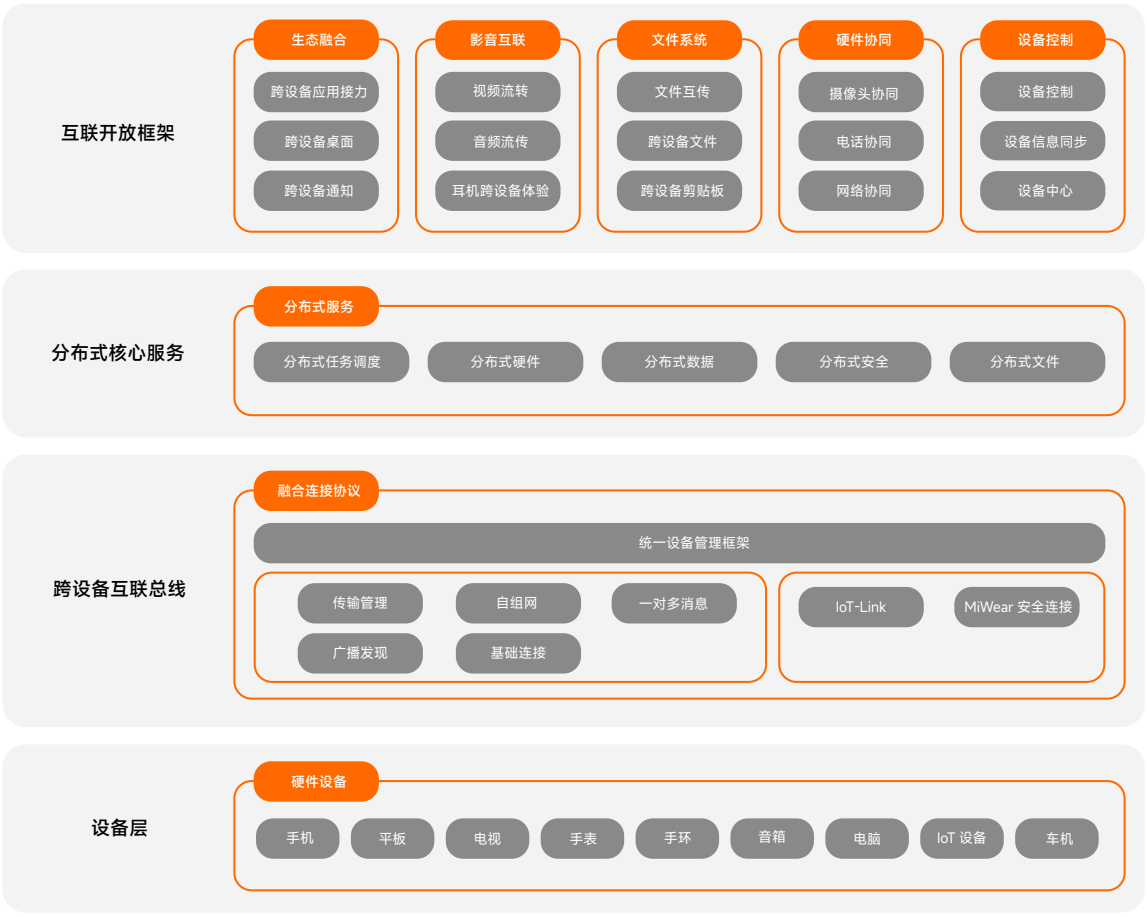


图 3.1-1 分布式子系统架构

3.1.3 重点特性

3.1.3.1 跨设备总线

技术介绍

跨设备总线提供面向多设备互联、跨平台、面向异构网络的广播、发现、连接、传输能力。

- 广播：提供统一的广播发送功能，支持自定义广播，协议兼容。
- 发现：提供统一的、可复用的发现功能。
- 连接：提供统一的基于服务驱动的经典蓝牙，低功耗蓝牙，局域网，点对点等连接通信能力，支持多端连接、通道复用。
- 传输：提供面向服务的信息交换传输通道。

技术架构

图 3.1-2 跨设备软总线 SDK

关键技术

1. **智能自组网**

自组网可实现周边设备和服务的自动感知、同步，实现安全认证和加密传输，简化业务开发难度，提高接入效率。提供多端统一开发模式，极大的简化接入业务的复杂度，并降低了系统的开销。

优势：低功耗、动态的发现策略；服务发现到设备发现；虚实连接结合的组网管理。

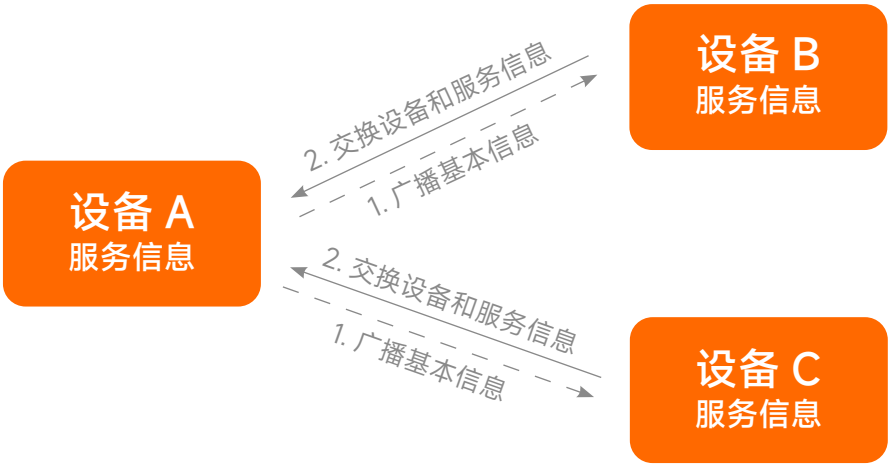


图 3.1-3 智能自组网

- 快速发现，稳定连接，跨设备可信互联

图 3.1-4 异构组网无感连接

- **设备靠近自感知**
  采用多种融合技术，包括 BLE、局域网广播、局域网组网等机制，快速发现周边可信设备。
- **组网策略自调节**
  感觉环境事件、用户事件动态调节自感知技术的参数，使得功耗、性能均衡，无需业务感知就能实现快速的组网。
- **拓扑与信息自维护自适应**
  自动维护设备的拓扑关系、设备的能力信息、以及设备的环境信息，并以极低的开销实现资源的快速同步。
- **运行时机自调度**
  基于自组网的业务，可定义业务激活条件，条件满足后，才拉起激活业务，避免业务必须常驻才能实现业务。
- **多链路动态复用**
  结合多种通信技术，决策最优媒介类型，快速建立链接，提升互联体验。

2. 一对多消息分发协议

- 一对多消息分发机制支持应用自定义的 Topic 主题进行消息订阅和发布，根据应用发布的数据大小自动选择链路并自动完成连接和传输的过程，为应用提供安全、快速、稳定的传输通道。

关键功能点：

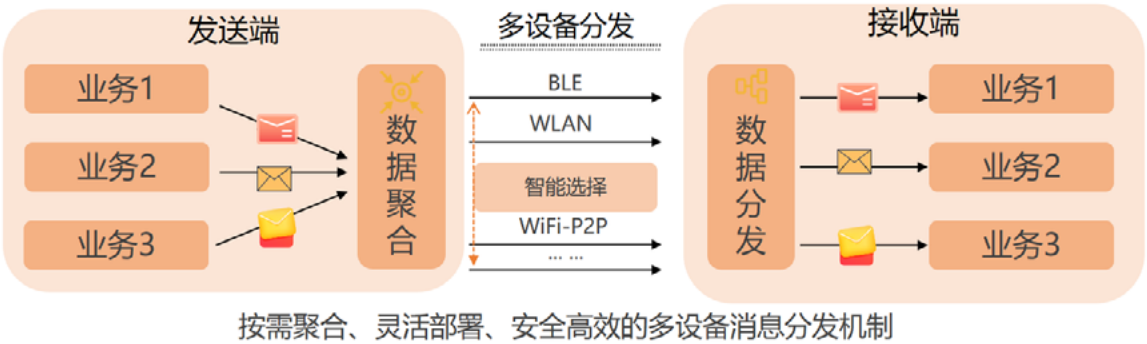


图 3.1-5 多设备分发架构

- **智能决策**
  根据设备间组网的状态、设备数量、设备状态、网络状态、消息的大小自选择合适的链路送达对端设备。
- **稳定可靠**
  消息支持按需获取全量数据，消息送达时，可保障消息的事件能快速达到，并按需获取全量消息。
- **通道复用**
  多个消息，可合并为单一消息，提高复用度。
- **安全加密**
  通过广播、连接等方式的数据，均通过加密方式送达，确保信息的安全。
- **灵活便捷**
  可支持所有同账号设备、指定设备、指定组网类型等多种方式送达消息。

3.1.3.2 服务框架

技术介绍

在跨端安全隐私的基础上，实现数据、文件、流媒体跨设备迁移，同时支持硬件的跨设备组合使用。

技术架构

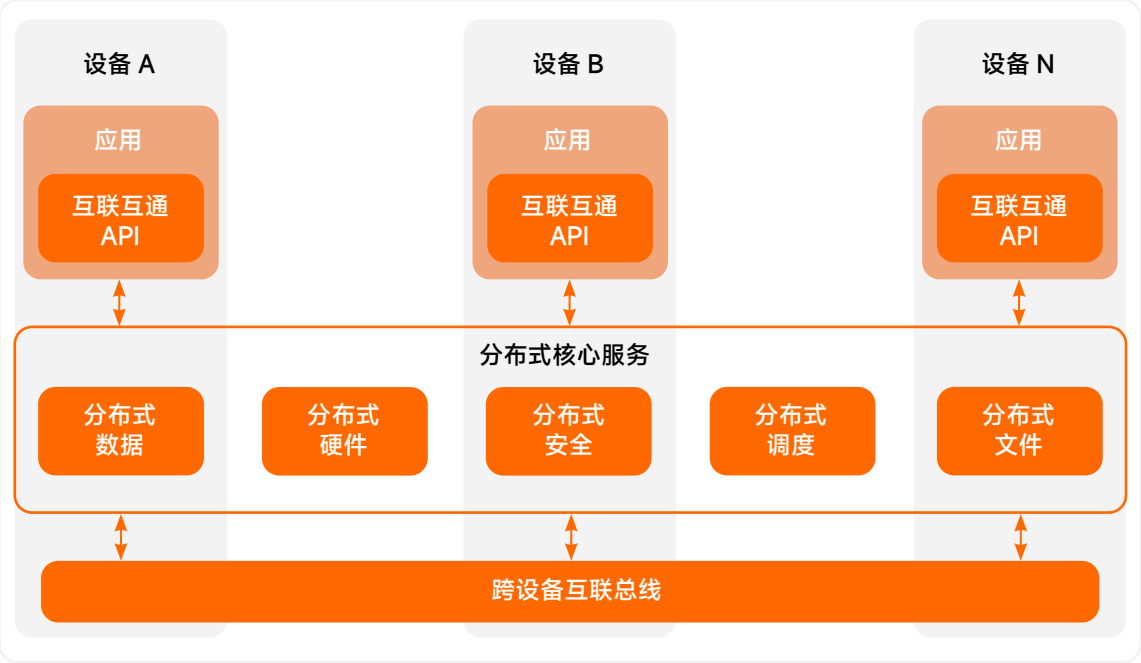


图 3.1-6 分布式服务框架

关键技术

1. **分布式硬件**

   分布式硬件实现跨设备的资源融合和访问。针对不同应用场景，为用户匹配并选择能力合适的执行硬件，使应用无感在不同设备间流转，充分发挥不同设备的能力优势。分布式硬件是多端协同交互体验的关键技术。

图 3.1-7 分布式硬件

2. 分布式数据

分布式数据基于跨设备互联软总线，实现系统和应用程序数据以及用户数据的分布式管理。支持跨设备相互同步用户数据，使多种终端设备上数据访问体验保持一致。通过调用分布式数据接口，应用可以将数据保存到分布式数据库中。

图 3.1-8 分布式数据

3. 分布式安全

分布式安全确保跨端可信服务，保证设备本身是合法的设备，防止被劫持，被 root 和破解，通过跨端权限子系统保证用户设备的安全和隐私安全，给用户进行敏感权限提醒，以及跨端行为记录追溯历史。

图 3.1-9 分布式安全

4. 分布式文件

分布式文件为用户设备中的应用程序提供多设备之间的文件共享能力，支持相同帐号下同一应用文件的跨设备访问，应用程序可以不感知文件所在的存储设备，在多个设备间获取文件。

图 3.1-10 分布式文件

3.1.3.3 跨设备应用开发框架

技术介绍

提供能力完备、高效低门槛的互联接入方案。面向互联服务开发、应用接力开发、硬件协同开发等不同层次开发者开放。

技术架构

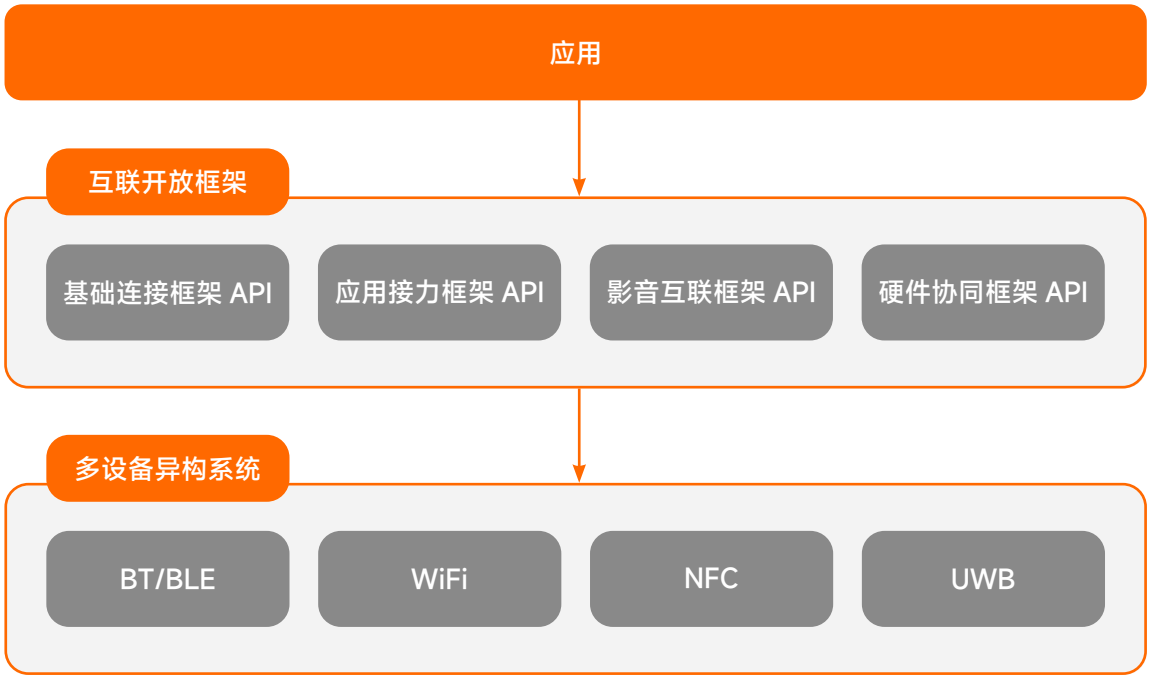


图 3.1-11 跨设备应用开发框架

关键技术

1. **基础连接框架**

   基础连接框架 API 包括组网发现、数据传输、消息分发。

基础连接框架 API

完成跨设备间快速通信，提供组网发现、数据传输、消息分发等系统能力

图 3.1-12 基础链接框架 API

- **组网发现**

  可提供设备发现、设备变更、服务上线、服务下线，组网类型变更等多种发现功能。发布者发布服务，自组网完成服务同步，观察者即可收到服务通知。支持静态注册方式，当满足条件时，自动拉起服务。

- **数据传输**

  归一化蓝牙、WLAN、WiFi-P2P 等不同通道创建的差异，提供一致性、极简接入。接收端指定服务名称，发送指定设备 ID 和服务名称，即可创建多种类型的通道，实现发送消息、文件等多种传输能力。数据传输支持静态注册方式，可在触发接收时，自动拉起服务。

- **消息分发**

  接收者指定 Topic 订阅内容，发布者指定 Topic 发布内容，自组网内所有订阅该 Topic 的接收者均能收到消息。消息分发支持静态订阅方式，可在收到消息时，自动拉起服务。

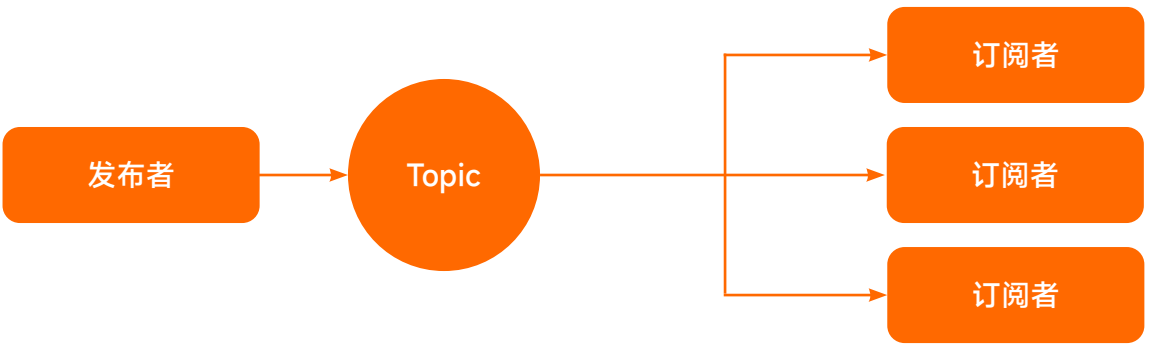


图 3.1-13 消息分发

2. **应用接力框架**

- 向系统注册接力服务后，在用户触发应用接力后，自动迁移。将在迁移发起端存储数据，接收端恢复数据，类似本地应用前后台切换一样便利。

应用接力框架 API

提供跨设备的应用接力能力，可提供跨设备间快速、稳定的应用接力体验

图 3.1-14 应用接力框架 API

3. **影音互联框架**

影音互联框架 API

提供跨设备的影音流转和控制能力，使用附近的大屏幕或扬声器，获得更好的影音体验

图 3.1-15 影音互联框架 API

- 基于小米跨设备影音互联框架打造，拥有极致互联性能。
- 支持全系小米核心设备，包括手机、平板、笔记本、电视、音箱。
- 支持全系主流音视频协议，包括 MiPlay、Miracast、DLNA。
- 支持控制中心操控体验，同时支持应用内集成系统播控交互。
- 在视频场景调用影音互联 API 弹出设备列表，即可完成互联迁移。

4. 硬件协同

硬件协同框架 API

提供跨设备的硬件虚拟化能力，用用使用附近设备的硬件能力，就像本地调用一样简单

图 3.1-16 硬件协同框架 API

- 远端硬件资源本地化，像访问本地硬件一样访问附近设备的硬件能力。获取服务时，指定设备 ID，即可获取远端指定设备的本地化服务代理。

3.2 跨端服务框架子系统

3.2.1 子系统简介

跨端服务框架子系统定位于 OS 以上，基于系统基础能力与接口，以 SDK 或者接口的方式为上层应用提供统一的服务能力。这些服务具有专业的领域知识，如应用服务提供 OCR/ 天气等能力，媒体服务提供抽帧 / 编辑能力等。

从行业整体技术发展来看：

- 服务组件是每个 OS 体系中必不可少的一个层级。
- 行业参与者均在服务组件层面不断升级更新，旨在助力 OS 生态大力发展。

3.2.2 子系统架构

图 3.2-1 跨端服务框架子系统架构

3.2.3 重点特性

3.2.3.1 应用服务框架

技术介绍

应用服务组件目标是为开发者的应用提供天气、OCR 及交互、高质录屏等一致的系统服务。应用服务组件位于开发者应用与操作系统之间，为应用开发提供基础服务能力。

技术架构

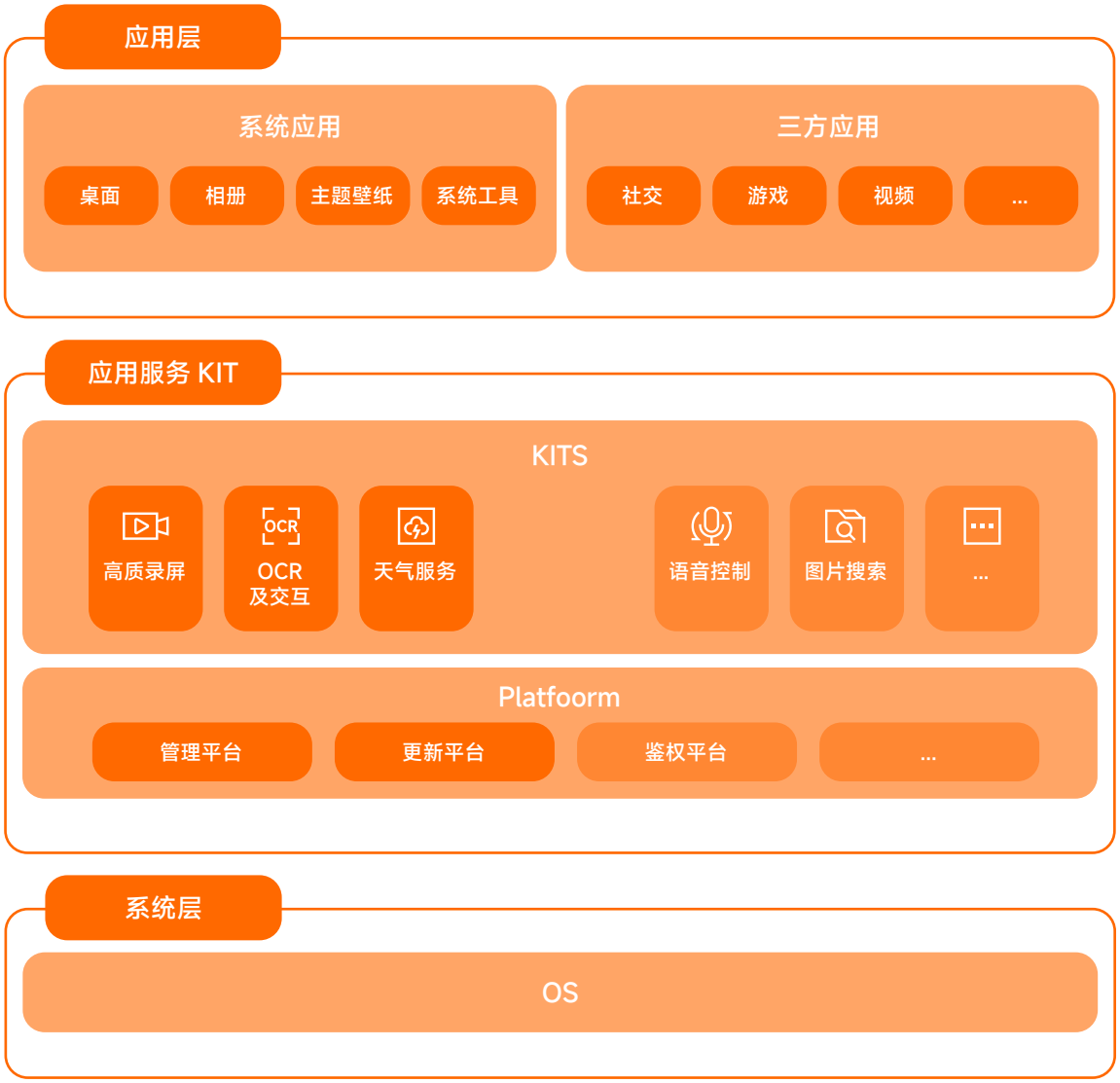


图 3.2-2 应用服务框架技术架构

关键技术

- 应用服务组件平台采用动态加载方案，支持组件管理器下载安装、组件版本管理和组件服务管理等功能。
- 天气服务组件提供城市天气和城市信息查询能力。
- 录屏服务组件提供高质量系统内声音和屏幕录制能力。
- OCR 服务组件提供便捷的图片文字识别和交互能力。
- 图片搜索组件提供图片名称、类型、文字识别等图片关键信息搜索能力。

3.2.3.2 高性能视频播放服务

技术介绍

高性能视频播放服务提供能力丰富、性能强悍的音视频播放服务，助力业务快速集成并定制自己的播放器，为用户提供优秀的视频媒体体验。

技术架构

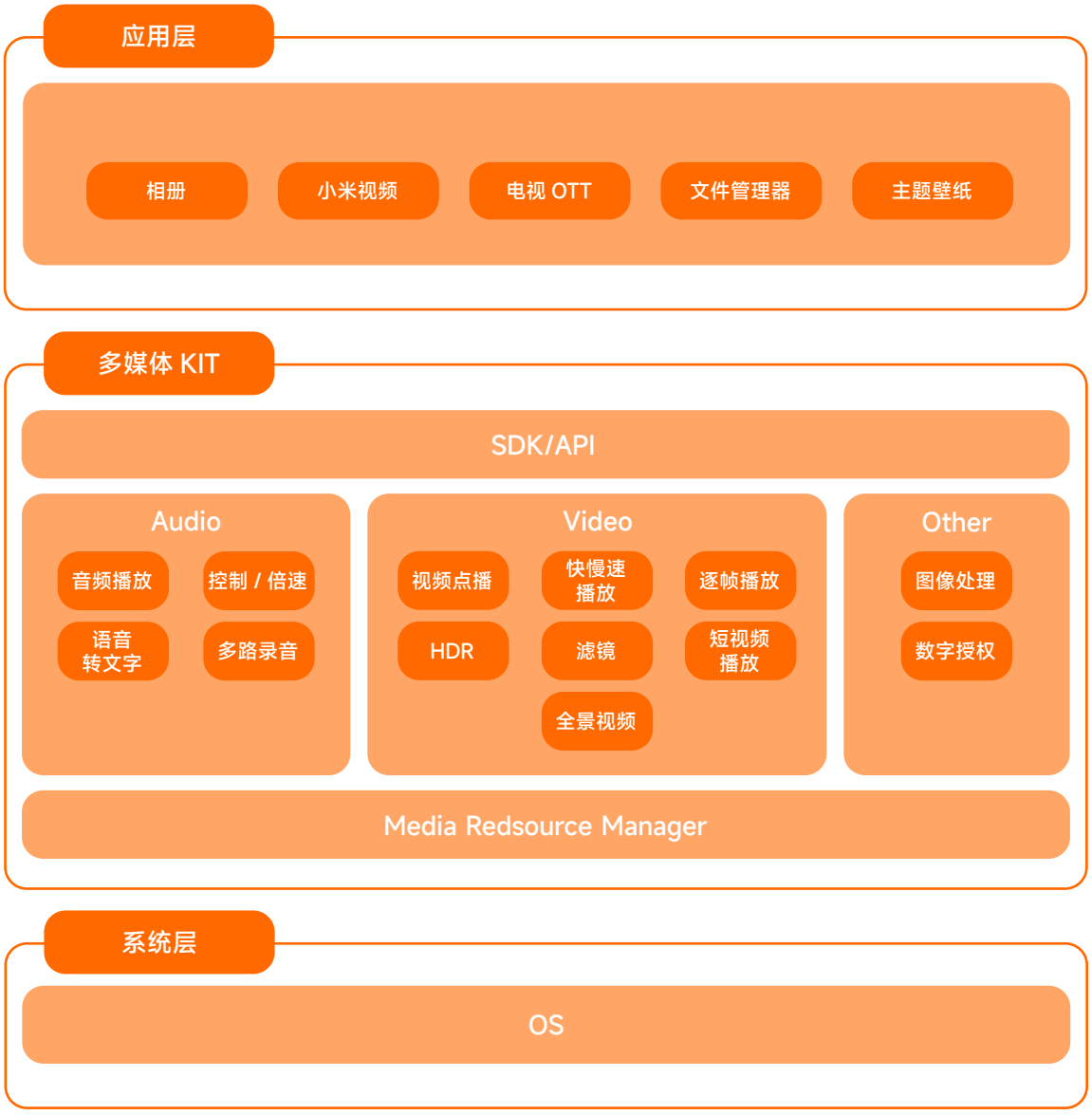


图 3.2-3 高性能视频播放服务技术架构

关键技术

1. **快慢速播放**：支持小米相机拍摄的慢动作视频的播放、编辑和配乐。
2. **逐帧播放**：支持对视频进行逐帧播放，满足快速浏览视频内容的需求。
3. **短视频点播**：性能强劲的短视频播放能力，可快速响应视频切换。
4. **全景视频**：功能完备的全景视频播放能力，可通过传感器和屏幕触摸控制播放视角。

3.2.3.3 响应式 UI 组件

技术介绍

为上层系统应用提供自适应的容器、布局、控件，支持一套代码自动适配不同各种不同小米设备（手机，平板，折叠屏）。

技术架构

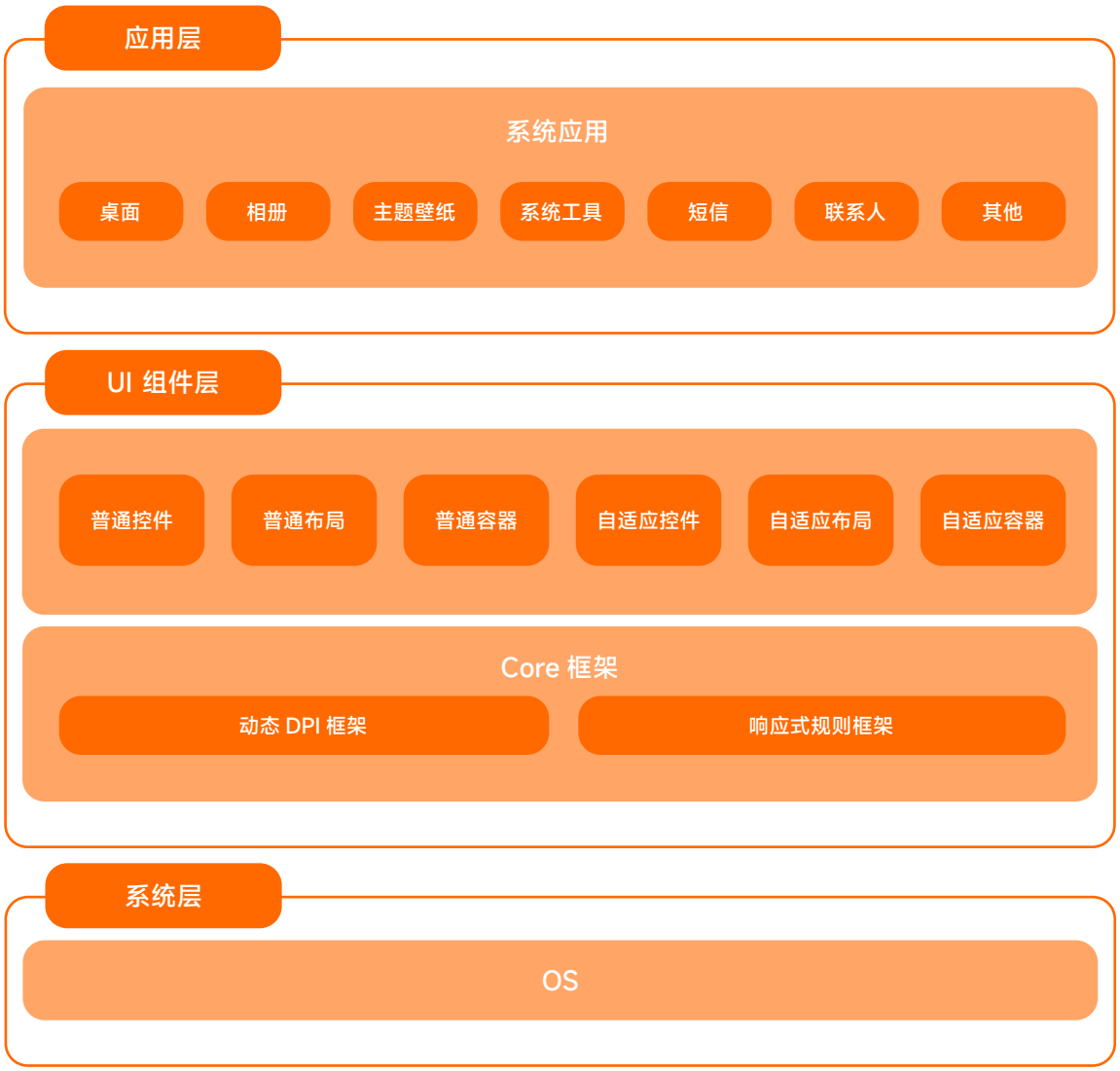


图 3.2-4 响应式 UI 组件技术架构

关键技术

1. **屏幕密度参数优化：**自动调整应用屏幕密度参数，适配不同分辨率。
2. **自适应规则优化：**根据不同设备 / 窗口类型，自动调整页面容器排版，控件形态变化。
3. **控件内容规则优化：**根据不同的文本内容，自动调整控件形态。

3.2.3.4 互联服务框架

技术介绍

通过近场通信的跨设备互联私有协议，实现跨平台的基础互联，进而支撑跨端分布式架构系统。在跨设备互联总线的基础上，传输实现快速互联、安全统一的架构体系，并为上层应用提供基础的安全通信和同账号系统服务，实现更全面的分级覆盖策略和协同升级能力、跨端无缝连接体验，以及持续连续演进的跨设备功能。使应用生态融合，影音互联，文件系统，硬件协同，硬件控制方向继续大力扩展。

技术架构

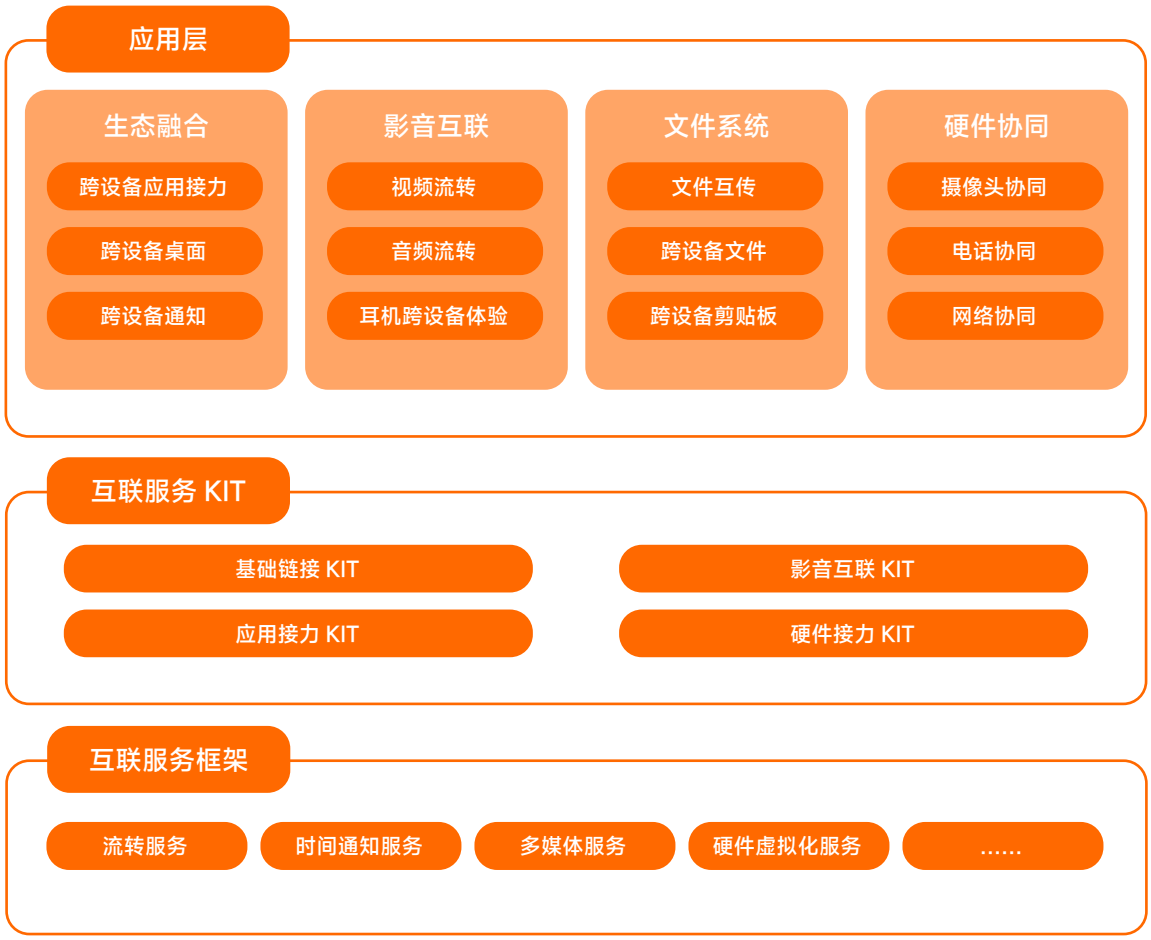


图 3.2-5 互联服务框架技术架构

关键技术

1. **高性能基础连接：**为跨端业务提供高性能的设备发现、连接、传输、组网和消息分发机制，构建便捷高效的跨端通信底座。
2. **硬件虚拟化服务：**实现资源全面虚拟化，达到跨端资源共享与融合，极简接入，丰富跨端业务场景。

3.2.3.5 AI 服务框架

技术介绍

AI 服务框架是包含大模型能力在内的系统 AI 能力集中管理框架，并且提供智慧能力服务。框架内部采用模块化设计，使得 AI 能力提供方能够灵活地接入，同时为 AI 能力需求方提供统一、可靠、高效、丰富的 API 接口，同时也提供了智慧能力的系统服务，助力应用程序实现 AI 相关的复杂功能。

技术架构

图 3.2-6 AI 服务框架技术架构

关键技术

1. **Xiaomi HyperMind**：全设备思考中枢，通过原子感知能力，学习人的习惯，让周边设备基于习惯来运作和协同。
2. **大模型**： 整合了视觉大模型和语言大模型，能够将用户输入的图片、视频、文字等信息进行精细化理解和整合。
3. **运行框架**：提供了模型加载、推理、优化、并行处理等关键功能。保障模型运行的高性能和效率，同时支持模型的灵活更新，使得 AI 服务框架能够不断适应新的 AI 技术。

3.2.3.6 安全服务框架

技术介绍

安全服务框架的目的是为应用提供系统级安全开放能力。为开发者提供了设备可信服务、恶意应用检测和虚假点击检测三方面的功能。

技术架构

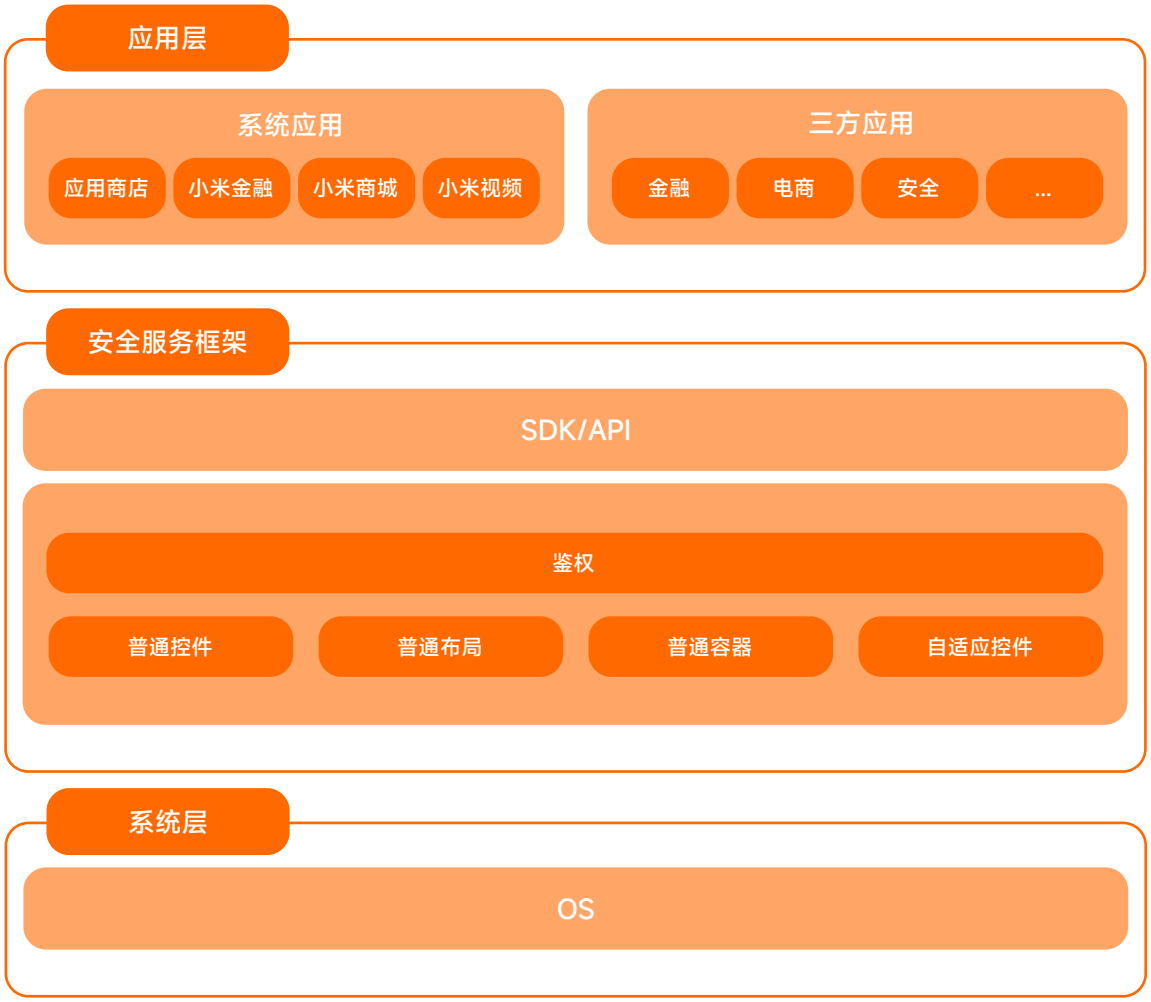


图 3.2-7 安全服务框架技术架构

关键技术

1. **设备可信服务：**在 TEE 中动态获取系统完整性数据，加密后的数据通过安全服务框架提供给应用。
2. **恶意应用检测：**系统侧动态检测系统中是否存在恶意应用。
3. **虚假点击检测：**系统侧根据点击事件输入结合当前系统状态检测是否发生虚假点击。

3.2.3.7 图形计算框架

技术介绍

图像计算框架基于 GPU/CPU 提供能力丰富、性能强悍的图形图像处理能力，帮助应用低成本集成相关技术，提升产品业务体验。

关键技术

1. **超分技术：**基于不同硬件平台，通过使用各种算法和深度学习模型，提高图像的质量，使其更清晰、更详细。
2. **图形引擎：**面向应用提供的一款高性能、低功耗、轻量级渲染引擎服务。

3.2.3.8 系统服务框架

技术介绍

系统服务框架为应用提供高阶系统能力，比如马达震动、网络增强、信号增强等能力。

关键技术

1. **马达振动：**支持应用自定义震动类型和频率，打造体验更好的震动用户体验。

3.3 跨端公共能力子系统

3.3.1 子系统简介

跨端公共能力子系统定义跨端兼容接口层，并提供跨端公共组件开发套件，让系统开发者可以使用 C/C++/Rust 等语言开发高性能的系统原生程序或库，可在标准系统与轻量系统共用，以实现标准系统和轻量系统一致的公共能力。

3.3.2 子系统架构

图 3.3-1 跨端公共能力子系统技术架构

3.3.3 重点特性

3.3.3.1 跨端兼容接口层

技术介绍

跨端兼容接口层致力于在高度约束的环境中维护一组标准内核与轻量内核的兼容性接口层，这一兼容性接口层是指内核空间到用户空间的接口层，其目的是让用户空间的程序或库可以在限定的条件下，无需关心底层的标准内核或轻量内核差异，使用统一的系统调用，无需针对特定的内核进行修改或重新编写。

技术架构

图 3.3-2 跨端兼容接口层

关键技术

1. **标准接口：**标准接口是指一组头文件（header files），用于定义和声明在 C/C++ 代码中使用的函数、结构体、宏和常量等，不仅规范了标准系统调用（POSIX APIs），还包含了处理网络、图形、线程、内存管理、标准加解密等。
2. **公共基础库：**公共基础库是指一组动态链接库、静态链接库，包含公共功能和算法标准实现，例如加解密、跨进程通信、序列化与反序列化等。开发人员可以在不同的程序中重复使用这些代码，不仅可以提高代码的复用性与一致性，还能够有效降低资源占用。支持多种 ABI（Application Binary Interface），每个 ABI 都有其特定的库文件。

3.3.3.2 账号

技术介绍

小米账号是用户用于访问所有小米服务和设备的统一身份标识，为用户提供账号身份验证服务。目前小米账号在中国大陆和海外多个地区本地部署，为全球小米用户提供符合当地隐私法规的服务，小米账号在手机、平板、笔记本、电视、有屏音箱等各种设备端提供身份能力与服务。

技术架构

图 3.3-3 账号子系统

关键技术

1. **用户身份和数据全面安全：**以跨端兼容接口层为基础，保障用户在手机、平板、笔记本、电视、有屏音箱等各种设备端的帐号拥有一致的安全性，安全的使用账号认证信息注册和登录，登录过程中和存储到本地的账号数据也是安全的。
   使用跨端原生开发套件开发，深度使用底层能力，基于 TEE 环境中的私钥提供真实设备数据签名。实现了跨端统一的安全秘钥、安全存储、加密网络传输，网络模块加固，Token 生命周期管理等。
2. **灵活方便快捷的集成：**基于不同业务场景提供多种 SDK 或 JSBridge 模块，并实现页面交互和多种登录方式，包括传统密码登录、取号短信登录、OAuth 授权登录、扫码登录、Webview 自动登录等。

3.3.3.3 云服务

技术介绍

云服务是多设备的个人数据中枢，主要功能围绕个人数据在多设备场景下的同步、备份、管理和分享等能力。云服务存储的是个人私有数据，安全的数据存储是基础能力，智能数据整理可以帮助用户更好的使用数据，共享协同能力方便用户更方便的分享数据。

技术架构

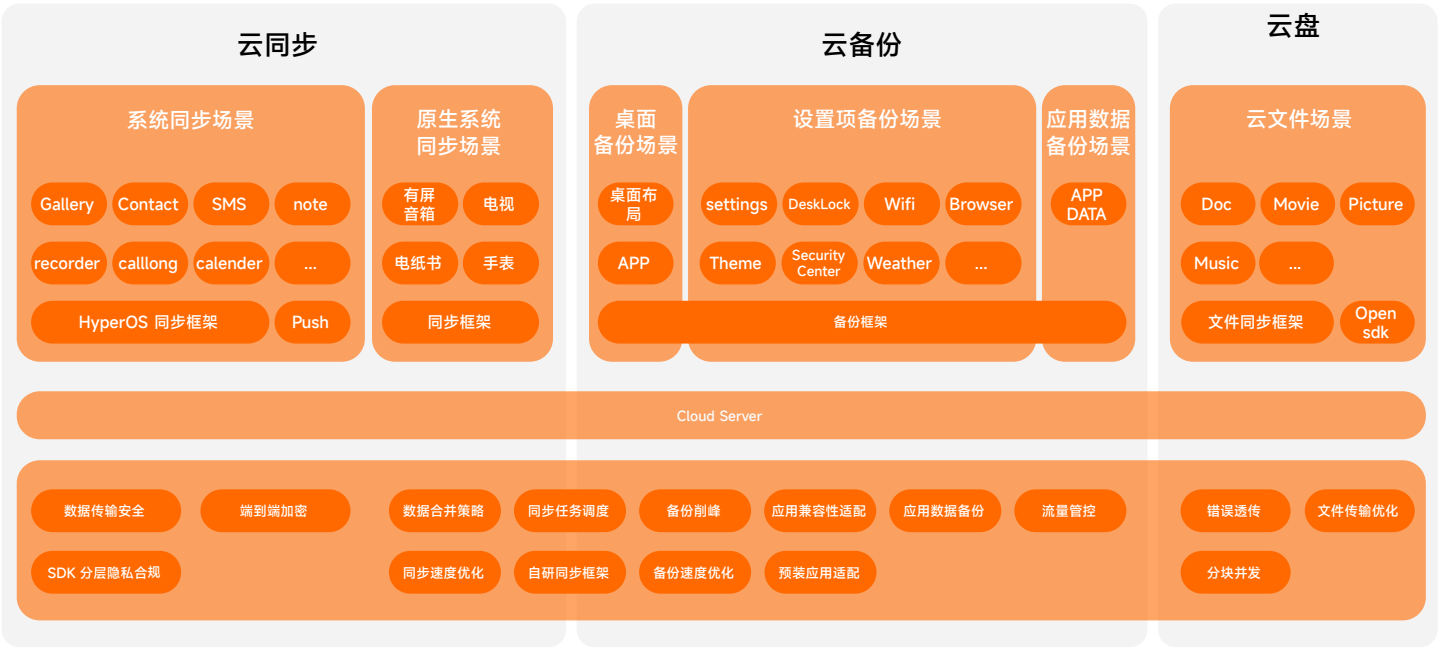


图 3.3-4 云服务子系统

关键技术

1. **端到端加密：**E2EE（End-to-end encryption）是只有参与通讯的用户可以读取明文信息的通信系统，可防止窃听者（潜在窃听者包括电信供应商、互联网服务供应商以及该通讯系统的提供者）获取双方通信的明文，能为用户提供更高的数据安全性保护能力。
2. **全平台统一接入：**以跨端兼容接口层为基础，实现跨端公共的 MiCloudKit 系统组件，集成云同步、云盘、云备份跨端统一，提供权限和访问控制、传输数据加密、数据同步、日历同步、便签同步、联系人同步等公共能力。

3.3.3.4 日志与系统事件

技术介绍

通过统一收集和记录系统组件在不同设备上的日志信息，开发者可以全面、有效的分析日志数据，识别潜在的问题，并进行故障排查。统一日志与系统事件系统可以提供运行时信息、错误日志、异常堆栈跟踪等，有助于快速定位和修复问题。并且可以根据预制的策略进行有效的自检、故障恢复、服务降级等。

技术架构

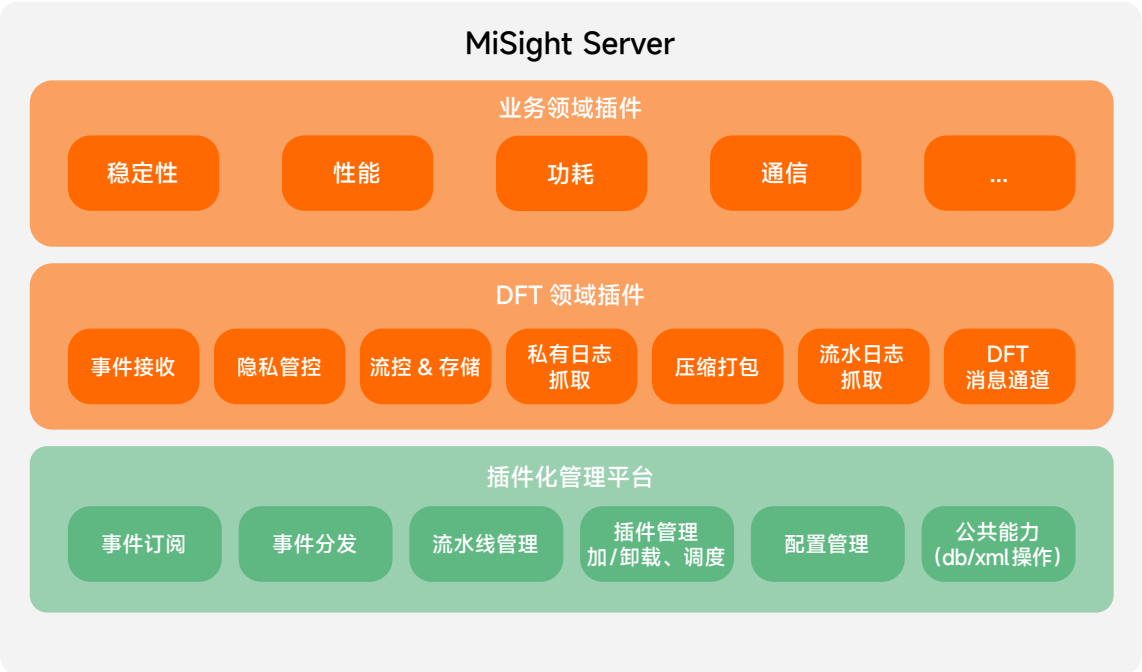


图 3.3-5 日志与系统事件

关键技术

跨端系统统一检测：以跨端兼容接口层为基础，实现跨端公共的日志与事件系统组件，统一标准系统与轻量系统的事件编码，支持各设备系统各层模块使用统一接口上报事件，实现了可跨端联合检测、观察、测试的事件设计。

3.4 跨端应用框架子系统

3.4.1 子系统简介

跨端应用框架子系统提供了面向具备跨端运行能力的轻量化应用的完整能力集合，开发者可以高效开发出运行在所有类型的澎湃 OS 终端设备上的工具应用、小部件、游戏等多种类型的应用程序。

跨端应用是一种新型的应用形态，它结合了原生应用和 Web 应用二者的优点，能够让用户无需下载安装，还能流畅的体验应用内容，实现了“即点即用”的用户体验；同时跨端应用尺寸小巧，可以灵活地分发到各种终端设备上，并可在多个终端设备之间进行快速流转，让用户可以随时随地使用跨端应用。

同时此子系统还提供了 IoT 框架，使采用米家 IoT 协议的设备能够轻松连接米家平台，实现与其他设备和云的互动。这包括多种基础通信协议和用于业务开发的功能模块。

3.4.2 子系统架构

3.4.2.1 整体架构

跨端应用框架子系统整体框架如下图所示：

图 3.4-1 跨端应用框架子系统架构

跨端应用框架子系统的设计以“一次开发、多端运行”为目标，通过对底层设备能力的抽象和封装，尽可能屏蔽各类设备的能力差异，并构建出满足不同应用类型的跨设备运行时环境；并在上层将这些运行时环境集成起来，为开发者提供一个规范统一、能力丰富的开发框架，降低跨端应用的开发难度；此外，跨设备运行时环境会根据各类终端设备能力差异性进行针对性的优化，保证应用在多种终端设备上均能提供流畅的用户体验。

3.4.2.2 技术特点

采用前端技术栈 + 原生渲染混合模式

- 前端技术栈 - 对前端开发者友好，充分发挥前端高效易学的优势，降低学习成本。
- 原生渲染 - 克服浏览器的渲染性能劣势，提供流畅的应用体验。
- 即点即用 - 继承网页应用的优势，避免过长的下载安装等待，快速触达用户。

提供能力完整的跨设备、跨系统应用框架

- 丰富的基础框架能力 - 丰富的 UI 控件、系统能力及三方服务、可满足诸如购物、阅读、影音娱乐等应用开发诉求。
- 高性能，优质体验 - 原生渲染技术，用户使用体验流畅。
- 自适应不同设备屏幕 - 通过界面逻辑分离设计、弹性布局、媒体查询等技术手段，实现组件位置尺寸灵活自适应能力。

提供高效动画渲染能力

- 高性能的物理动画引擎 - 采用逐帧实时计算的物理计算模型，可以更好地模拟真实世界中的物理效果。
- 创新的动画状态管理 - 提供多属性动画、属性监测功能、基于物理特性的运动计算等多种管理机制。
- 完善的动画编辑工具 - 专门为设计师或者运营人员打造，无需编程即能设计出美轮美奂的动画效果。

支持多端分发部署

- 统一工程开发 - 一个工程即可支持多类型设备的应用开发，提升代码复用率，避免冗繁的工程项目维护工作。
- 可组合的应用打包 - 将一个应用打成多个小文件包，按设备类型进行组合，支持按需下载。
- 灵活的分发策略 - 一个应用对应一组应用包，按设备类型 / 型号进行匹配管理，为设备分发合适的应用包。

支持多种应用形态

- 应用 - 即点即用，接近原生体验，提供丰富的 UI 组件和能力 API。
- 小部件 - 轻量化桌面卡片，可与应用联动，提供强大、流畅的动画渲染能力。
- 游戏 - 即点即玩，深度优化的游戏性能，支持国内主流游戏引擎，提供丰富的 Game API 和安全机制。

3.4.3 重点特性

3.4.3.1 跨端应用框架

技术介绍

应用模型

应用模型是跨端应用的抽象表示，一个跨端应用由一个 App 和多个 Page 组成，Page 之间可以通过路由进行任意切换，页面栈负责维护 Page 之间的切换关系，遵循先进后出的原则；任何时刻只能显示一个 Page，即位于页面栈顶部的 Page。

与应用模型相对应，一个跨端应用程序由一个配置文件和多个 ux 文件组成。配置文件用于定义应用基础信息、功能权限声明、系统配置和页面路由等数据； ux 文件则记录页面或组件的具体实现代码，包括 UI 模板、样式单、数据定义和回调事件处理等。

MVVM 模式

跨端应用框架采用主流的 MVVM 模式设计，MVVM 是 Model-View-ViewModel 的简写，本质上就是 MVC 模式的改进版本。MVVM 模式有助于将应用程序的业务逻辑与 UI 清晰分离， 使应用程序更易于开发、测试和维护。在 MVVM 模式下，一个应用可以分为 Model、View 和 ViewModel 三个部分，其中 Model 表示业务数据模型、View 表示用户图形界面，ViewModel 则是将二者的关联起来的桥梁，通过数据绑定机制将 Model 的数据转化成 View 所呈现的用户图形界面，通过事件监听回调将 View 上产生的交互事件传递给 Model 来更新数据。

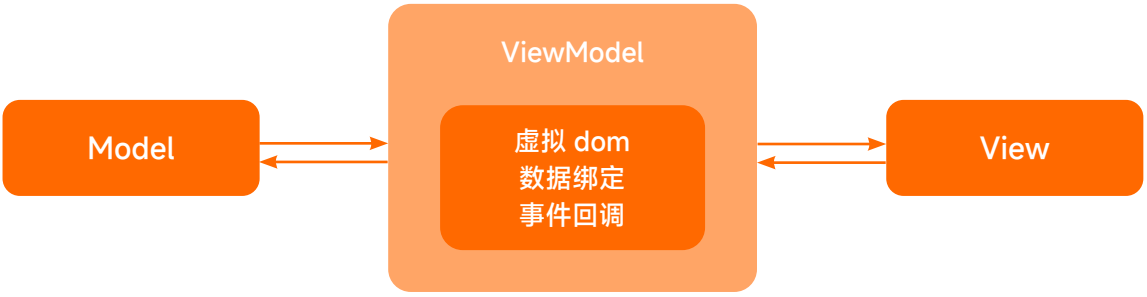


图 3.4-2 MVVM 模式

生命周期模型

跨端应用的生命周期由 App 生命周期和 Page 生命周期组合而成，Page 只能运行在 App 生命周期之内，而不能脱离 App 独立运行。App 生命周期主要包含 onCreate、onShow、onHide、onDestroy 等主要状态，Page 生命周期较为复杂些，主要包含 onInit、onReady、onShow、onHide、onRefresh、onDestroy 等主要状态。当跨端应用运行时，会在应用的不同的生命状态下，触发对应的回调函数，开发者可以精确掌控应用的运行状态并做出正确的响应操作。生命周期状态的具体作用说明见下表。

生命周期状态	说明
onCreate	监听应用创建 , 当 App 创建时调用
onInit	表示页面的 ViewModel 的数据已经准备好
onReady	表示页面的 ViewModel 模板已经编译完成，此时可以获取 DOM 节点
onShow	表示 App/Page 的已经显示
onHide	表示 App/Page 的已经隐藏
onMenuPress	当点击顶部系统菜单时调用
onBackPress	当用户点击返回实体按键、返回菜单等操作时调用
onRefresh	当页面重新打开时调用
onDestroy	当 App/Page 被销毁时调用，此时可以做应用退出前的资源释放操作，比如：取消事件订阅监听等

开发范式

语法规范

跨端应用框架是一套以前端开发技术栈为主的应用开发框架，语法规范的设计上，采用流行的前端开发模式，尽可能贴合主流前端开发者的思维习惯。每个跨端应用的页面由三个部分组成：<Template>、<Style> 和 <Script>，其中 Template 负责描述页面组成结构，Style 负责页面元素的样式描述，Script 负责逻辑处理。

- Template - 采用类似 HTML 的标签语言，利用基础组件、自定义组件、block/slot 等元素构建出复杂的页面结构，还支持列表渲染、条件渲染、数据 / 事件绑定等高级功能。
- Style - 遵循 CSS 语言规范，描述页面中组件的显示样式。
- Script - 遵循 JavaSscript ES6 语法标准，编写代码实现具体的业务逻辑和数据处理。

路由管理

跨端应用框架负责管理整个应用的页面路由，实现页面间的无缝切换，并管理每个页面的完整生命周期。开发者需要将页面在应用配置文件中进行注册，也可以在 Script 中通过路由 API 来控制页面切换。

路由管理除了支持应用内页面跳转之外，还支持应用间的跳转，即从一个应用中打开并跳转到另一个应用的某个页面，比如打开系统设置的 WIFI 管理界面。

数据绑定

数据绑定是 MVVM 模式的重要功能，开发者通过数据绑定，只需少量的代码就可以实现数据与视图的同步。在 Script 中对已经绑定的数据进行修改时，视图就会自动完成相应的更新。对于数组类型的数据，可以通过循环渲染指令简化绑定过程，能够根据数组元素的数目，自动生成相同数量的 Dom 节点完成一一对应的绑定。布尔类型的数据，可以配合条件渲染指令，动态控制页面中组件的可见状态。

系统组件

跨端应用框架提供了一组基础的原生实现的系统组件，系统组件除了支持常用的 HTML5 标签（比如 、 <a>、<input> 等）外，还提供了更多复杂的 UI 组件（比如 <swiper>、 <tabs>、 <marquee> 等）。

由于不同的设备对系统组件的支持能力存在差异，因此可将系统组件分为通用组件和非通用组件：

- 通用组件：可以运行在所有类型的设备上，比如
- 非通用组件：只能运行在特定的一种或几种设备上，比如 <tabs>

在开发跨端运行的跨端应用时，尽可能使用通用组件；如果必须使用设备相关组件时，可以通过条件渲染、动态组件等手段来减少适配工作量。

组件自定义

为了更好的组织逻辑与代码，可以把一个页面按照功能拆成多个模块，每个模块负责其中的一个功能部分，最后页面将这些模块引入管理起来，传递业务与配置数据完成代码分离，这些模块这就是自定义组件。

自定义组件不同于框架所提供的原生实现的系统组件，它是由开发者采用跨端应用语法来实现的，使用方式与和系统组件是完全一样的；自定义组件拥有与 Page 相同的生命周期模型，也具备对数据、事件和方法的管理功能；另外也为组件提供了父子组件通信、兄弟 / 跨级组件通信机制，高效实现组件之间数据传递和协同工作。

原生系统能力

跨端应用框架提供了丰富的原生能力 API，既有通用的系统功能，比如设备信息、屏幕亮度、文件访问等，也有硬件模组相关能力，比如传感器、地理位置、蓝牙、NFC 等，还有第三方服务的对接，比如推送，支付、账号、广告等。这些原生能力 API 可以大大节省开发者工作量，快速开发出跨端应用。

与系统组件类似，原生系统能力也分为通用能力和非通用能力：

- 通用能力：在所有类型的设备上均可用的能力，比如应用上下文、日志打印、页面路由等。
- 非通用能力：只能在特定的一种或几种设备上可用的能力，通常与设备的硬件模组相关，比如地理位置、震动、传感器等。

为了方便跨端开发，跨端应用框架采用能力可用性查询 API 和能力调用降级处理等手段，确保当使用无效的设备相关能力时不会导致应用运行异常。

多语言覆盖

跨端应用框架也提供了多语言支持能力，可以做到让一个跨端应用同时支持多个语言版本的切换，开发者无需开发多个不同语言的源码项目，避免给项目维护带来困难。开发者配置多语言的方式非常简单，只需要定义资源、引用资源两个步骤即可完成；另外也提供标准 API 允许跨端应用在运行过程中修改地区语言。

关键技术

UI 自适应

智能终端设备种类繁多，在屏幕大小、屏幕分辨率、页面结构、交互方式等方面都有差异，因此页面的自适应能力就非常重要，否则会给开发者带来大量的适配工作。

UI 自适应能力通过多种技术手段和途径组合实现，最终保证页面能够在各种设备上呈现出满意的 UI 表现：

- 弹性布局 - 弹性布局的独特之处在于能够扩展和收缩组件元素，以最大限度地填充可用空间，因此弹性布局可以让页面在面对不同的屏幕尺寸时，具备更好的适应能力
- 等比缩放 - 每个 App 定义了设计宽度属性 designWidth，当实际运行的设备屏幕宽度与 designWidth 不同时，采用等比例放大缩小来实现自动适应。
- 非等比例屏幕适配 - 非等比例的适配也是开发中经常遇到的一种需求，比如：同一个页面在大屏和小屏上显示的内容不同。针对这种需求，采用设备独立像素、媒体查询等手段来控制页面上组件的显示状态（比如大小、显示 / 隐藏等）来解决内容适应问题。

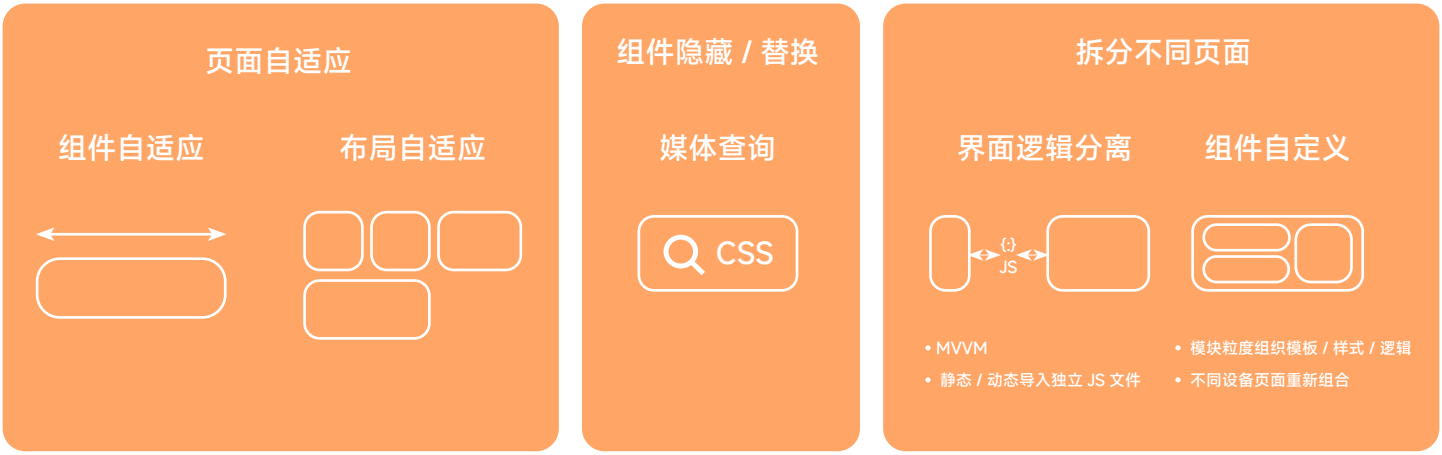


图 3.4-3 UI 自适应技术解决方案

可见即可说

可见即可说能力为应用提供了 2 种开发接入方式，一是语音技能绑定方式，该方式利用强大的后端 AI 服务提供复杂的 NLP 模型和意图识别能力，而且也可以为应用关联更复杂的在线服务和业务逻辑；二是本地意图识别方式，采用框架预置的意图识别引擎，只需简单的配置就可实现本地化的基础语义理解能力，能够大大降低使用门槛。

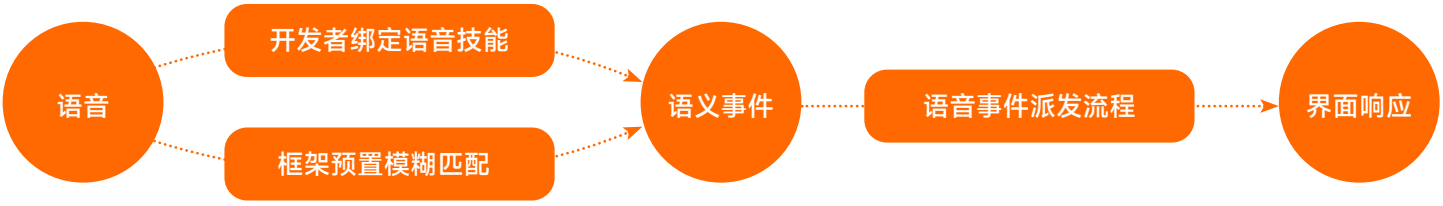


图 3.4-4 可见即可说接入方式

3.4.3.2 小部件

技术介绍

小部件是一种轻量的卡片应用，可以驻留运行在终端设备的桌面应用上。由于小部件采用各种规范尺寸的卡片来呈现，所以界面内容相对简单，通常借助设计工具来辅助开发，其典型开发流程如下所示。

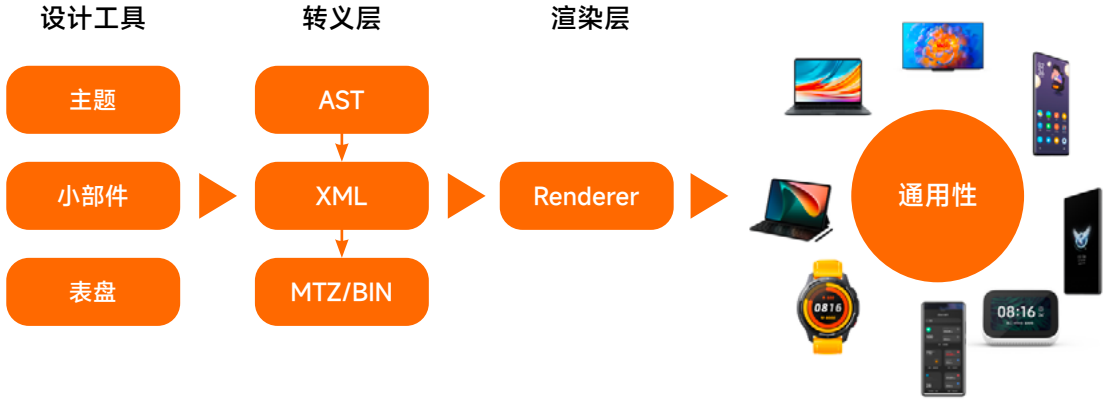


图 3.4-5 小部件典型开发流程

小部件采用自己独有的运行时，其技术特点如下：

- 统一跨端体验 - 面向跨端的运行时设计，采用统一的动画计算模型，为各类终端设备提供一致的交互体验。
- 高效的动画渲染 - 小部件注重动画效果，因此采用高性能的物理动画引擎，帮助开发者实现酷炫的动画表现。

技术架构

小部件运行时主要包含 Amaml 渲染器、动画引擎和卡片容器等核心模块，其整体框架结构如下图所示。

动画引擎提供了丰富的动画接口，支持 Folme 物理动画引擎、关键帧动画、Lottie 等等，Amaml 渲染器底层采用 Canvas 图形渲染，实现了图像、文字、矢量图形等元素的高性能绘制和场景管理；卡片容器集成二者，实现卡片应用的生命周期、VDom 以及交互事件等基础管理功能。

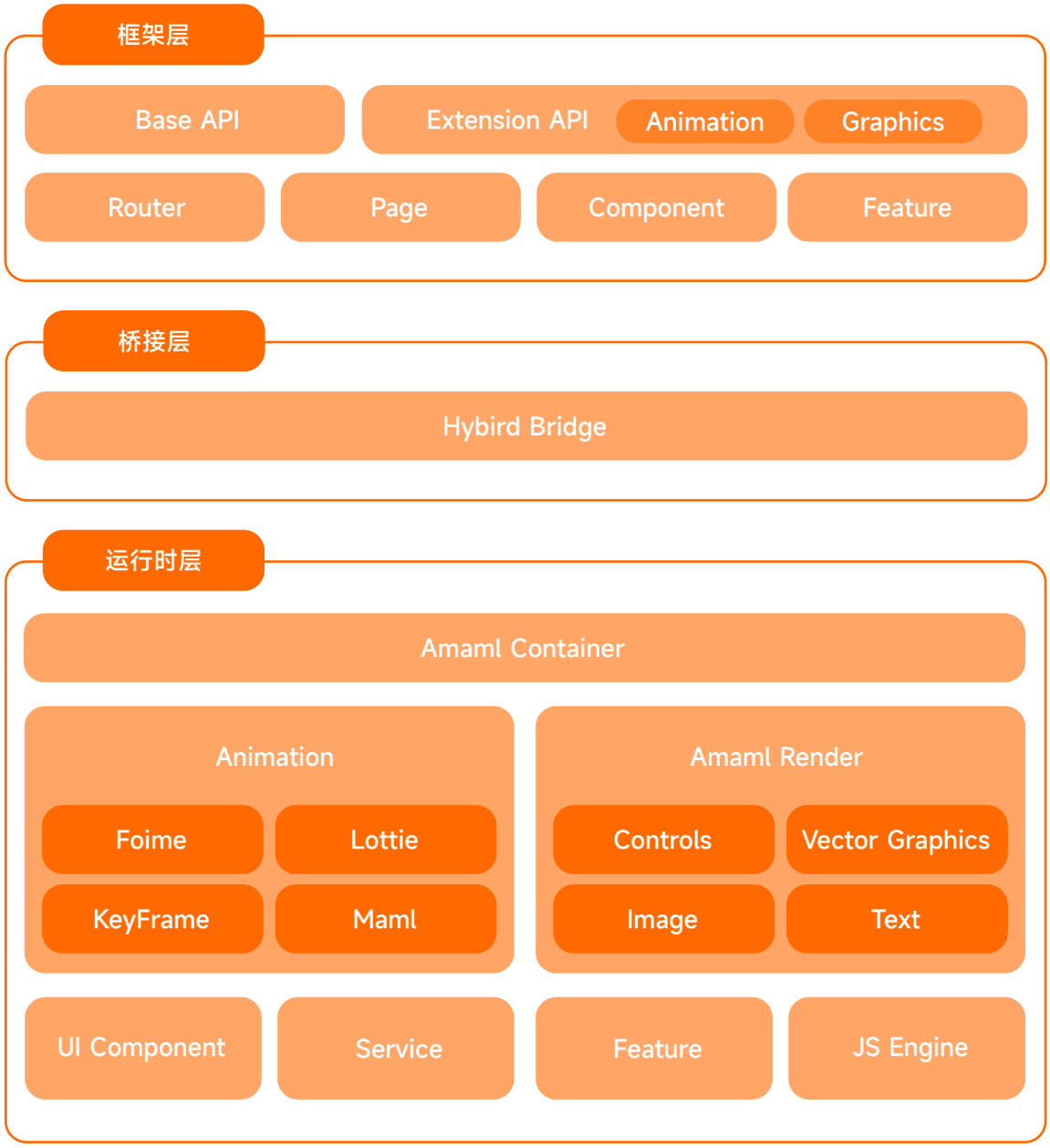


图 3.4-6 小部件架构

关键技术

1. **Folme 动画引擎：**Folme，取名于 Follow Me，是一套设计端与开发端统一思路与接口的跨平台递进式动画 SDK，其保证了设计端和开发端一致的语言、一致的参数得到一致的效果，并大幅提高设计效率与研发效率。

   在以往的动画设计与研发流程中，动画对接都是一个耗时耗精力的过程。原因大致有：

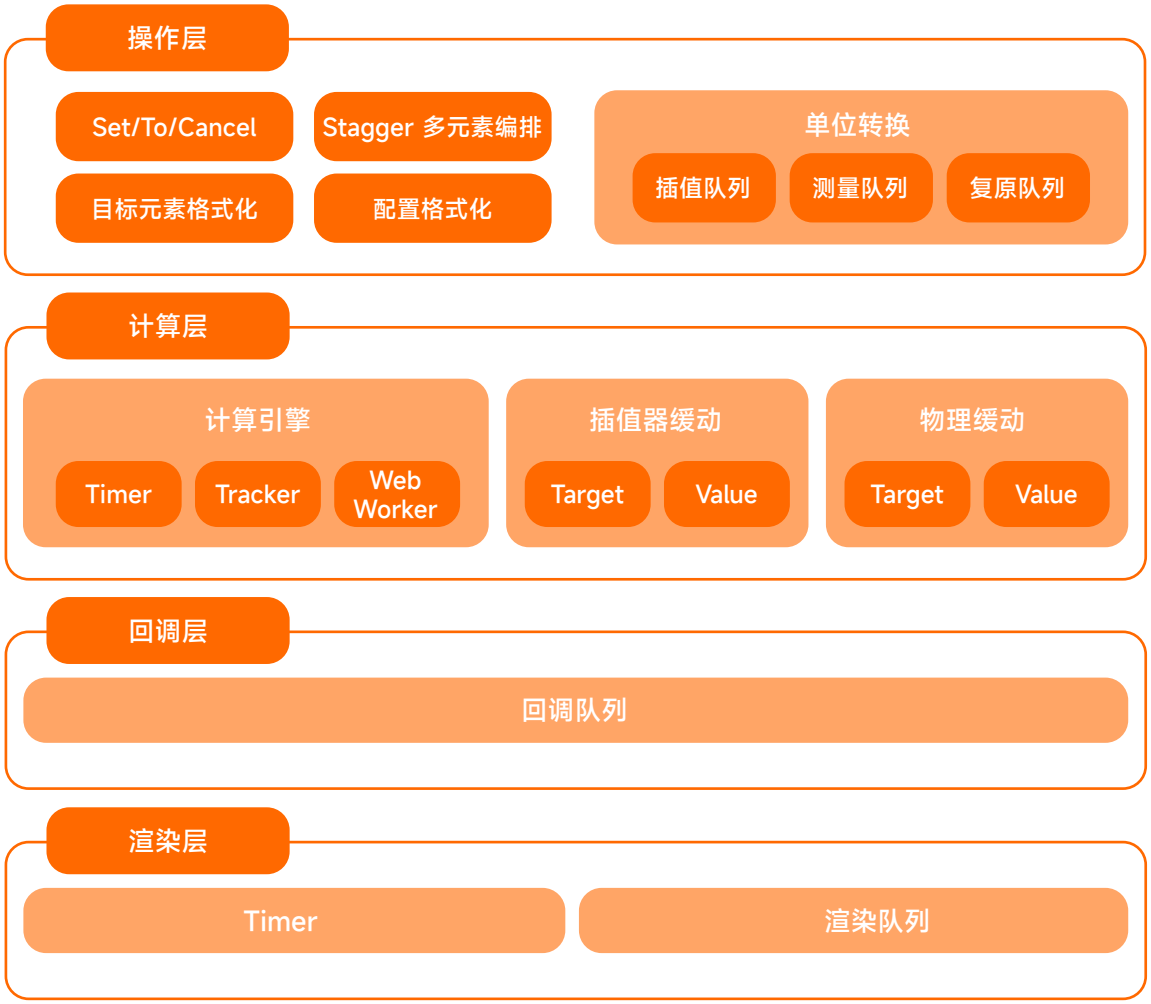


图 3.4-7 Folme 动画引擎架构

- 设计与研发工具不统一，导致动画参数与形态不一致。
- Android 动画框架过于老旧，不支持交互、打断、可变速度等操作。
- 传统的插值器不能支撑现代动画设计。

针对这些问题，我们以“提升动画对接效率，支撑现代化的动画效果”为设计目标研发了 Folme 引擎，其主要特点在于：

- 高性能：Folme 非常注重动画的性能，尤其是在多元素同时运行的场景下，Folme 做了大量的性能优化，保证动画的流畅顺滑。
- 模块化：Folme 层级之间相互解耦并且可以单独调用。开发者可以基于 Folme 提供的基础模块开发可适配不同运行环境的动画引擎。
- 使用简单：Folme 在操作层做了大量的工作，优化了接口调用方式，添加了多元素编排，支持智能单位转换等等。

此外，我们也为开发者提供了 Folme 组件库，将项目中常用的动画效果抽象封装成为开箱即用的组件，这有利于提高小部件的开发效率，也方便技术沉淀和积累。Folme 组件的优势在于：

- 开箱即用：只要在页面中简单引入一个 js 文件即可使用组件，非常方便。

- 跨框架：组件库基于 WebComponent，可以在任何框架下像使用原生组件一样使用 Folme 组件。
- 可扩展：组件库是按需加载的，即便添加再多组件也不会影响到每个组件的加载性能。
- 安全：组件库使用 Shadow DOM，开发者不需要担心引入组件造成样式污染。

2. **动画编辑器**：动画编辑器是一款专门为设计师或者运营人员提供的简易动画编辑应用，旨在解决设计师在设计动画时需要进行大量简单而重复的工作痛点，利用补间动画原理，只需要设计师设计几个关键的动画帧，然后设置好帧与帧之间过渡的缓动曲线函数就可以实现很好的动画效果。

   动画编辑器支持的素材类型有：圆角矩形、椭圆、文字、图片、序列帧、编组，支持的属性有：大小、位置、缩放、旋转、颜色、边框颜色、字体、字体颜色、字体大小、不透明度、蒙版等，同时，还支持所有的缓动曲线函数。

   在设计阶段，动画编辑器采用可视化编辑方案，属性设置立即生效，同时支持高频操作的快捷键及拖拽操作，操作习惯与其他常用的设计软件保持一致，尽可能降低设计师的使用学习成本。此外，该编辑器对动画曲线的支持粒度非常地细，对每个关键帧、每个素材及每个属性都可以分别设置缓动曲线函数及 delay，让动画设计更加自由和灵活，动画效果更加酷炫。

   在预览和产出阶段，动画编辑器提供了两种方案：

- **Web 版** - 提供实时预览、生成 h5 动画并下载、协同与共享等功能。
- **桌面版** - 提供实时预览、工程文件保存、 代码辅助（支持手势、陀螺仪）、ADB 连接手机调试、生成代码文件等功能。

3.4.3.3 开发工具套件

技术介绍

开发工具套件为开发者提供了一个完整的、跨平台的开发环境，包含命令行工具 (AIOT-Toolkit) 和 IDE (AIOT-IDE) 两部分。开发者可以根据需要使用命令行工具或者 IDE 来进行开发，给开发者开发带来更便捷的可视化操作体验。

技术架构

开发工具套件组成结构如下图所示：

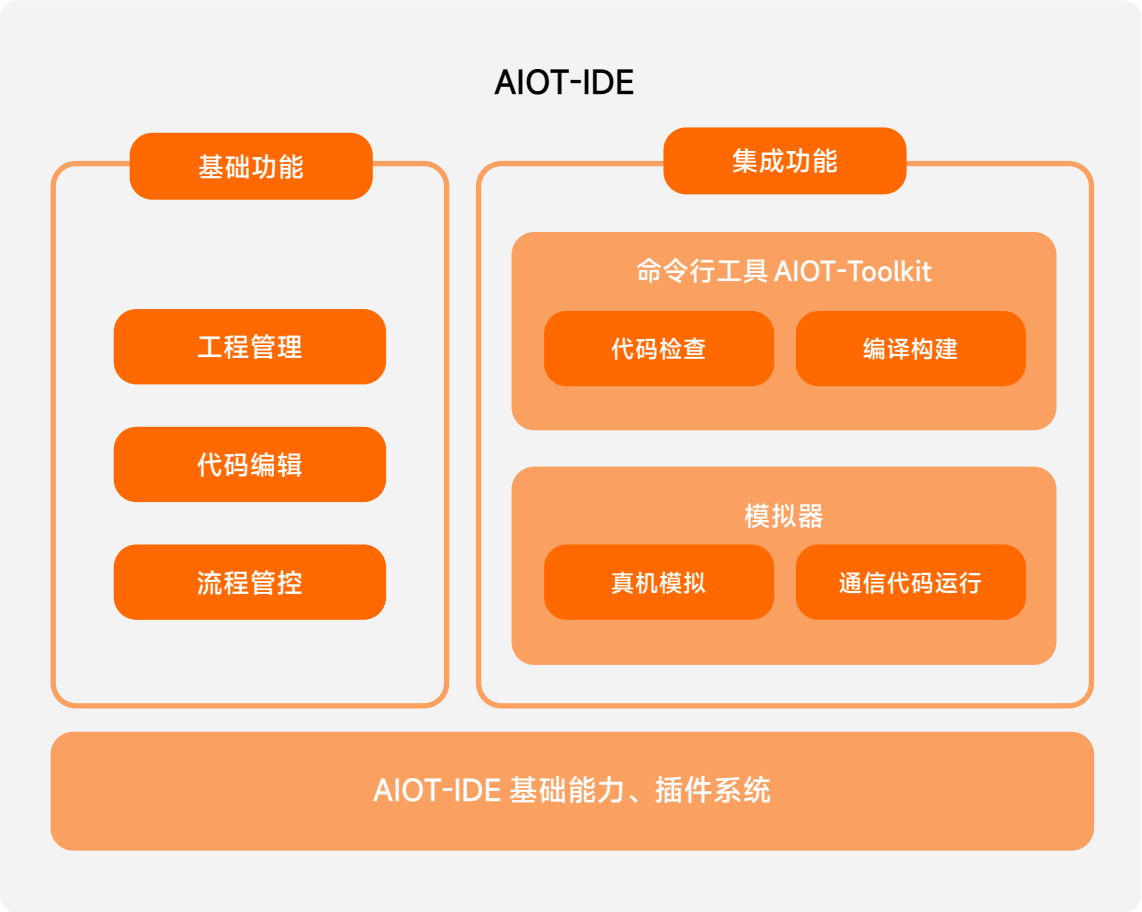


图 3.4-8 开发工具套件

开发工具套件技术特点包含：

功能强大的命令行工具

命令行工具基本覆盖了跨端应用开发的全部流程，包括项目创建、项目打包编译、项目运行。它基于 Nodejs 来开发，是一个跨平台、跨 IDE 的系统级命令行工具，开发者完全可以在喜欢的 IDE 中结合 AIOT-Toolkit 来开发跨端应用。

高度可扩展的插件机制

AIOT-IDE 针对跨端应用开发场景对 VSCode 进行了二次开发，完整继承了 VSCode 的插件能力，同时也对插件接口的能力进行了扩展，使得 IDE 即可以直接使用 VSCode 现有丰富插件，也可以定制开发出针对跨端应用的各种工具插件。

实时快速的语言服务器插件

语言服务器是跨端应用开发必备的代码理解帮助工具，它通过对跨端应用代码上下文进行实时分析，快速准确地完成代码提示、补全、代码诊断等工作，有助于减小代码中的人为错误，提高跨端应用代码开发效率。

关键技术

1. **工程管理：**提供项目管理的能力，包含创建项目，删除项目，以及项目文件创建、删除、项目资源管理等等。开发者可以方便地管理多个项目的文件和资源，提高开发效率。
2. **代码编辑：**代码编辑包含代码编辑器和语言服务器两部分：

- 代码编辑器 - 主要用于编辑代码。代码编辑器可以用于任何一种编码语言或框架。它具有代码词法高亮、自动缩进、自动补全、代码折叠等功能，以帮助程序员编写更高效、更易于阅读和维护的代码。
- 语言服务器 - 能够在开发者在进行跨端应用代码编辑的时候，对代码进行理解，对输入的代码进行实时语法高亮；根据代码的上下文进行代码提示和补全；对于输入完的代码，进行代码校验、诊断，对于存在错误或不合理的地方，给与标记并给出错误提示。

3. **流程管控：**在使用 IDE 进行三方应用开发过程中，流程管控能力提供了一个标准化的跨端应用开发流程管理功能，使得开发者可以清晰得知道自己在项目开发的什么阶段，当下需要处理什么问题；另外，流程管控还以可视化的方式，将整个开发流程串联，让开发者快速上手。

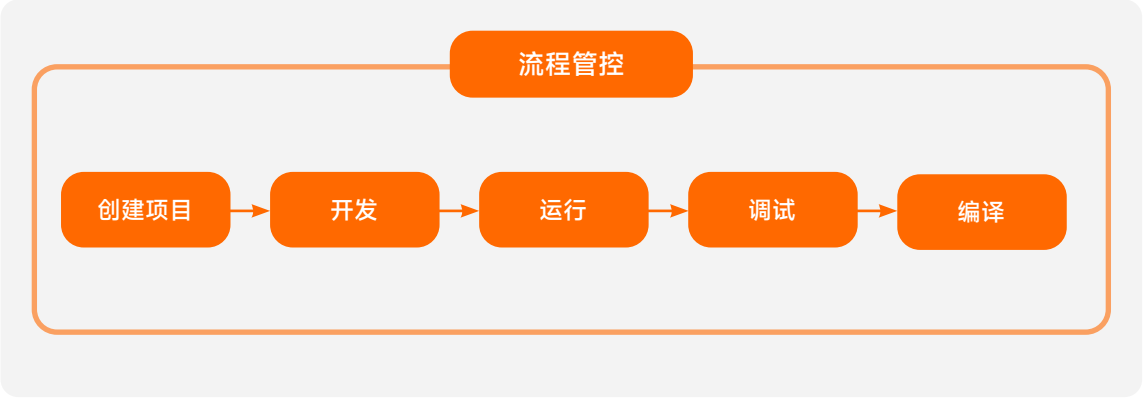


图 3.4-9 开发工具流程管控

4. **代码调试：**在开发跨端应用时，代码调试是使用频率非常高的功能。开发者可以使用代码的断点来跟踪代码中的问题；通过分析代码执行时产生的日志来定位代码中的问题。AIOT-IDE 中提供以下主要调试能力：

- **断点功能**
  在代码的编辑过程中，你可以对代码进行断点操作，这样，在代码运行的时候，运行到断点处时，代码会暂停在断点处。此时，你可以查看代码运行到此处的状态，如变量查看，逻辑分析等等，方便排查问题。
- **日志查看**
  并不是所有的调试都方便用断点的方式查看，比如程序运行过程中一个多次变化的变量，这时，你可以通过日志的方式，输出该变量，可以方便的查看这个变量的变化过程。

- **DOM 树查看**
  跨端应用在显示布局上与 HTML 非常类似，调试工具提供了 DOM 树查看的能力，该能力方便开发者可以准确快速的查看代码布局的情况，选中调试工具中的 DOM 树，该 DOM 树在渲染窗口中的实际渲染产物会被高亮，可以方便开发者定位布局样式等问题。

5. **命令行工具：**命令行工具是基于 Nodejs 的开发工具，它对用户开发的代码进行编译，并将源码打包成三方应用的格式 (RPK)，在打包的过程中，它对应用进行签名。在应用上架和安装过程中，都会进行签名验证，防止应用被恶意篡改。
   命令行工具提供了丰富的命令，可以支持所有类型跨端应用的打包编译。在打包编译的过程中，它还可以对源码进行校验，当源码出现一些错误时，会给出提示并指出打包错误的地方。除此之外，命令行工具还可以调用 Emulator 模拟器，可以让程序在模拟器 Emulator 中运行并查看运行效果。
6. **模拟器：**AIOT-IDE 集成了多种设备模拟器 Emulator，能完整支持跨端应用开发、调试和测试等工作，而且部分性能评估工作也可以在模拟器上完成。另外，模拟器也支持跨平台，可以在 Linux/Windows/MAC 等平台上运行。
7. **插件扩展能力：**AIOT-IDE 针对跨端应用开发场景对 VSCode 进行了二次开发，增加了插件接口的能力方便完成业务需求。基于 AIOT-IDE-SDK，可以开发出功能更强大的各类插件工具。此外，AIOT-IDE 也完整兼容 VSCode 的插件生态，让海量插件在 AIOT-IDE 上随搜随用。

3.4.3.4 IoT 框架

技术介绍

IoT 框架提供了通用的功能模块，使得采用米家 IoT 协议簇的设备可以接入米家平台，并和其他设备以及云服务进行交互。其中基础连接协议层提供多种不同的通信协议与跨端设备进行通信，而通用服务层提供不同的功能模块进行业务开发。

技术架构

图 3.4-10 IoT 框架

小米澎湃 OS 操作系统 IoT 子系统包含协议连接层、中间件、通用服务三层协议框架。

- **连接协议层**

  连接协议层包含 Wi-Fi 和米家 Mesh 协议，以及 PLC 有线协议。

- **中间件**

  中间件包含基于 TLS 的米家云端连接协议 OTS，以及米家局域网控制协议 OTU。同时实现了基于订阅发布的应用间消息总线和局域网端到端的分布式通信总线。

  MIoT Spec 是对于 IoT 设备规范化的功能性描述语言。

  米家自研的 Host 和 Mesh 蓝牙协议栈，扩展了低功耗和 Mesh 组网的 IoT 设备应用。

- **通用服务层**

  通用服务层为米家应用、云端、小爱语音提供应用服务能力。其包含畅快连配网服务，以提供设备的极致配网体验。以及固件 OTA 服务，用以设备固件下载和升级。

与此同时，通用服务层还提供了米家本地化服务集，包括中枢服务、本地控制服务和自动化服务，用以实现设备本地协同、离线控制和设备联动。

关键技术

1. **畅快连：**小米畅快连™是一种自动发现设备并配网、离线自动回连的能力，它在小米智能设备生态之上，基于米家 Wi-Fi 和 BLE 连接协议，提供了以下四大功能体验：

- 靠近配网：靠近即发现，点击即配网。手机靠近设备，系统 / 米家 APP 主动发现新设备，并引导用户快速绑定。
- 一键配网：插电即发现，点击即配网。搭配小米畅快连路由器使用，新 IoT 设备接通电源后，米家 APP 会主动弹窗提示发现新设备，一键点击即可快速进行配网操作，且不需要手动输入 Wi-Fi 账号密码。
- 无感配网：下单预绑定，插电即配网。用户在小米商城下单时选择了“无感添加”服务，新 IoT 设备接通电源后，与小米手机或小米畅快连路由器自动联动，用户全程零操作，是设备配网的最优体验。
- 改密同步：Wi-Fi 改密，设备自动回连。搭配小米畅快连路由器使用，当路由器修改 Wi-Fi 账号密码后，IoT 设备可自动连接新的 Wi-Fi 账号，避免需要重置设备重新配网。

2. **OTA：**固件升级是智能设备的关键能力，它可以为用户提供不断进化的产品体验。对于亿级设备的小米 IoT 平台而言，固件 OTA 能力至关重要。小米 IoT 平台具备行业领先的固件升级能力：

- 设备固件静默升级：取得用户的授权后，设备可按需启动智能升级，且升级过程对用户无打扰。可大幅提升设备升级率，便于产品新体验的推广普及。
- 基于容器化的独立升级方案：设备中的米家连接模块和产品功能模块实现解耦，可独立升级，降低开发者的开发维护成本。
- 实时监控与熔断机制：设备端支持分钟级的实时监控，在监控到升级关键指标异常时自动采取熔断操作，降低可能的用户和平台损失。

3. **网关服务：**网关服务作为一种通用能力，可以将其他协议的设备接入到米家平台，如蓝牙设备、PLC 设备等。通过统一的适配层，网关服务屏蔽了芯片、操作系统、协议带来的不同，为用户带来一致的体验。

   网关服务包括多项具体功能：

- 设备管理：网关统一管理设备的信息和状态，包括设备标识、网络拓扑、在线 / 离线状态等。
- 协议转化：不同协议接入的设备及其行为都会被转化为统一的 MIoT Spec 描述，为设备互通提供基础能力。
- 分组 / 批量 / 场景控制：根据使用场景的需求，提供分组 / 批量 / 场景等多种控制方式。
- 流量管理：屏蔽无效上下行流量、减轻服务器压力。
- 设备漫游：网关自动捕获附近设备，设备可在不同网关间无感漫游。

4. **本地化服务：**本地化服务是若干服务的能力集，包括中枢服务、本地控制服务和自动化服务等，可以在外网离线的情况下对设备进行控制和设备联动。各主要服务的功能如下：

- **中枢服务**

  中枢服务提供了网关设备之间协同能力，包括：

- 协同组网能力：集成不同中枢能力的设备，在局域网内可以组成主、备、从三种角色的中枢拓扑架构。
- 数据同步能力：主备之间可以进行文件同步和设备属性状态的同步，保证设备之间数据的一致性。
- 设备发现能力：通过设备发现协议探测 IoT 设备，并根据设备能力进行消息订阅和总线连接。

- **本地控制服务**
  本地控制服务结合中枢服务，对外提供 IoT 设备的本地控制能力。具备控制能力的应用和终端设备，通过连接主中枢控制其他设备。
- **自动化服务**
  自动化服务提供设备联动的能力，基于有向图规则引擎，可以创建不局限于 When-If-Then 的复杂场景。

3.5 跨端安全子系统

3.5.1 子系统简介

小米澎湃 OS 基于不同的硬件环境使用不同的 TEE 方案构建可信的安全底座，进一步依赖安全底座构建了框架层的设备可信、通用密钥管理和跨端身份认证的能力，对跨端场景提供了可信连接、可信传输和可信对端状态。

3.5.2 子系统架构

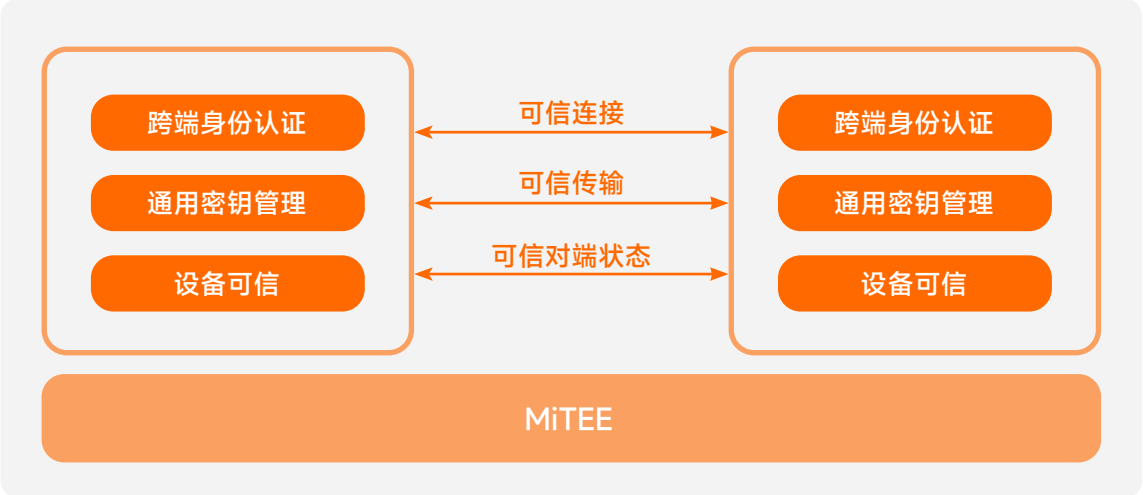


图 3.5-1 跨端安全子系统整体框架

3.5.3 重点特性

3.5.3.1 可信连接

技术介绍

为保证小米澎湃 OS 的连接安全，实现用户数据在多设备场景下各个设备之间的安全流转，需要保证设备之间相互正安全可信，系统提供了安全可信的身份认证和可信连接。

技术架构

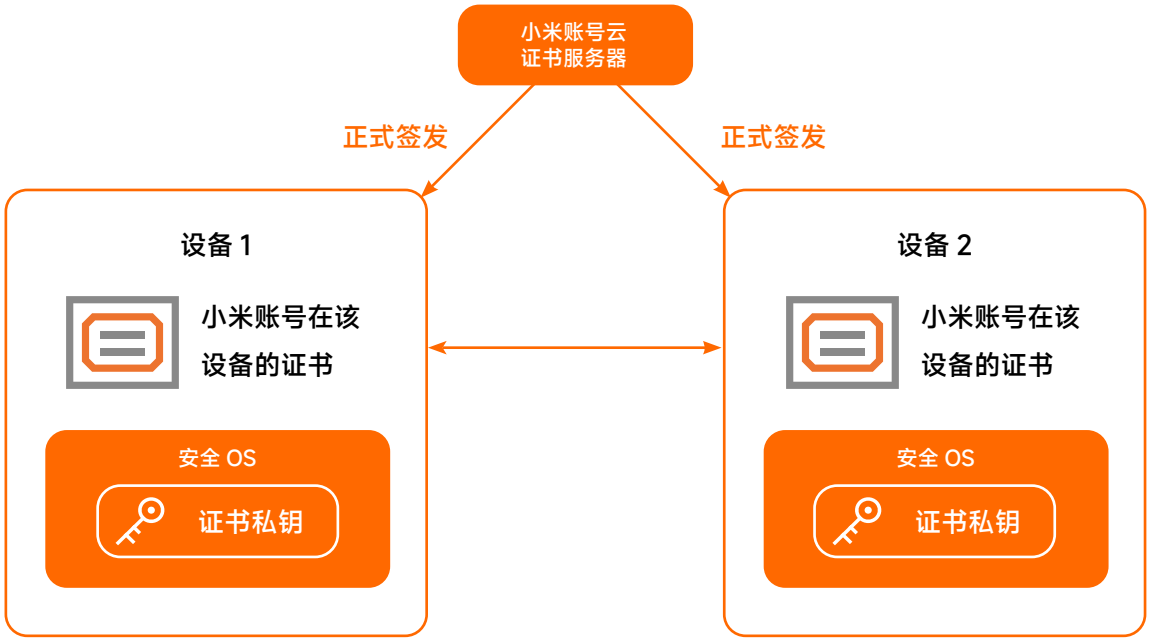


图 3.5-2 跨端可信连接

关键技术

1. **证书私钥：**设备在登录小米账号后，将由小米账号云为该设备签发证书，确保该设备登录了该账号。证书将用于同账号设备之间的相互认证。
2. **私钥保护：**证书的私钥由安全 OS 全生命周期保护，在认证过程中，将校验对端设备证书的合法性、有效性，确保证书是签发给该设备的。即使将认证在另一台小米设备上使用，也将无法通过身份认证。

3.5.3.2 可信传输

技术介绍

提供有身份验证，加密保护的可信通信，确保数据的机密性和完整性。

技术架构

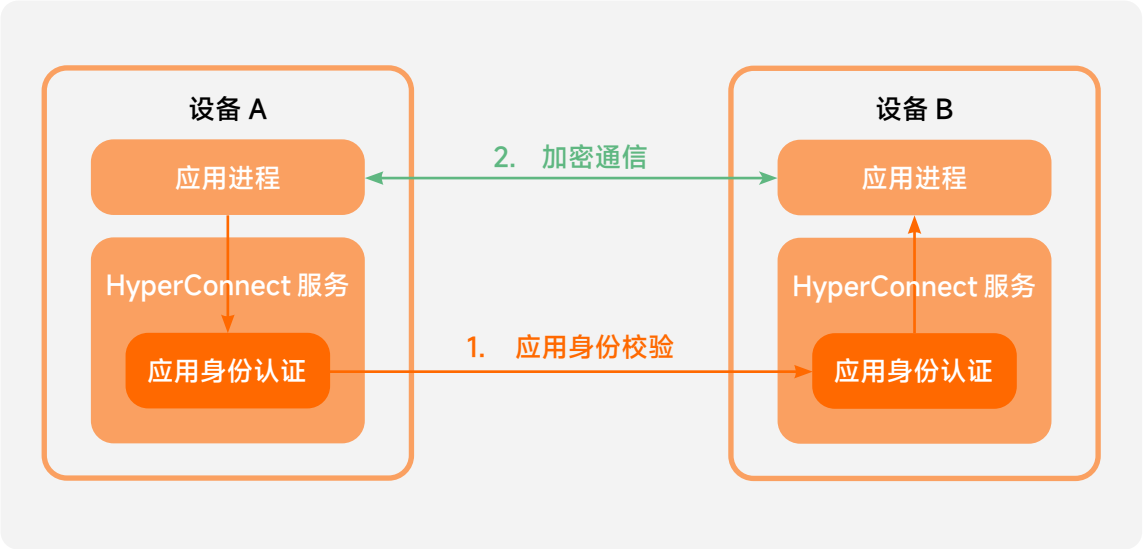


图 3.5-3 跨端可信传输

关键技术

1. **应用身份校验：**在设备级安全连接的基础上，对应用之间的通信连接也会进行应用身份校验，确保跨端应用身份的合法性，避免应用伪装等场景。HpyerConnect 服务将校验应用的签名信息，包名等关键信息，确保两端应用的通信的身份安全。
2. **加密通信：**应用间任一连接、任一会话将使用独立通信秘钥，通信的数据将自动加密，确保数据的机密性和完整性。

3.5.3.3 可信对端状态

技术介绍

小米澎湃 OS 借鉴了可信计算和零信任安全模型等前沿的安全理念和架构，在深度融合了小米 IoT 万物互联的实践经验上，通过动态验证对端的可信状态，来应对万物互联时代带来的新挑战。

关键技术

1. **统一的安全建模：**小米澎湃 OS 基于分布在不同设备上硬件隔离环境，对系统运行状态提取运行特征，并根据统一的安全模型叠加不同设备类别的维度，对设备进行动态度量，用于识别设备是否处于安全状态。
2. **可信的对端度量：**小米澎湃 OS 基于出厂时在不同互联设备上基于硬件执行环境预置的信任根，来达到可信确认对端安全状态的目的。
   处于互联状态中的小米设备，在本端完成动态度量后，度量结果加上互联对端传入的挑战值，组合后由硬件执行环境中的信任根进行加密及签名保护，得以可信任的被对端确认。
3. **持续的动态验证：**在互联状态下，基于可信的互联对端度量，小米澎湃 OS 在系统底层维护了互联对端的可信状态，并开放的提供给互联应用作为参考。在互联应用需要时，可以实时对对端可信状态进行刷新，达到动态验证的目的。

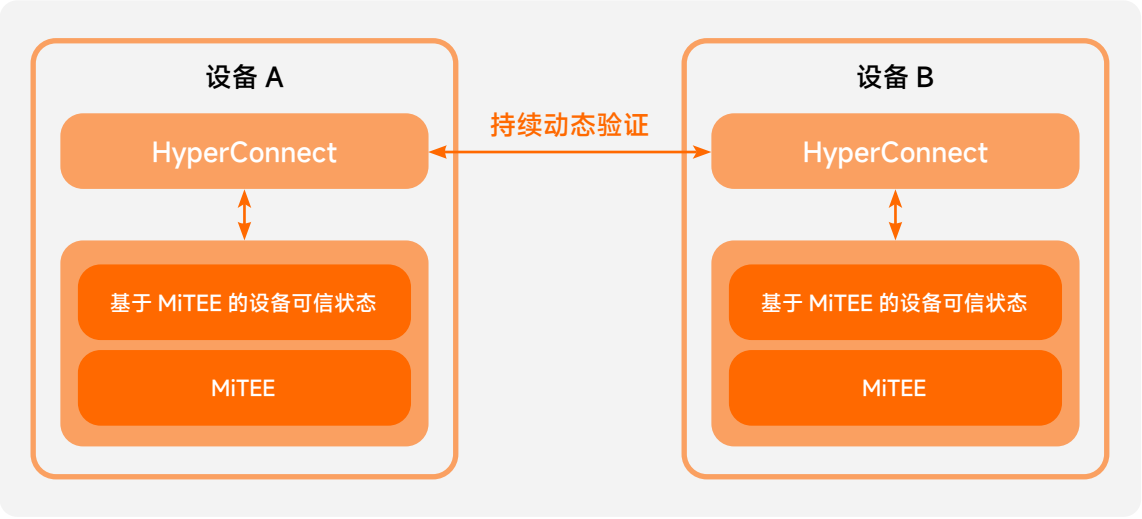


图 3.5-4 跨端持续动态验证

4. **永不信任的主动免疫：**互联状态下的不同小米澎湃 OS 设备实体，秉承永不信任的零信任思想，在确认对端可信的情况下进行互联融合，也会在动态验证对方发现问题时进行主动免疫，避免互联场景引入的单点设备风险。

4 系统服务与框架 (Service & Framework)

4.1 资源调度子系统

4.1.1 子系统简介

资源调度子系统主要围绕系统资源来进行工作，聚焦任务如何在硬件上进行管理和调度，主要提供三个关键能力，即：抽象协同芯片硬件，释放硬件能力；统筹管理资源供需，提升全局能效；使能应用高效运行，提供生态入口。

资源调度子系统重点组成部分为 CPU 资源子系统（存储、GPU 会在其他章节介绍），CPU 是对手机的所有硬件资源进行控制、运算和管理的核心硬件单元。随着 CPU 制作工艺的提升，手机所能提供的算力也越来越强，游戏和相机等高负载应用，在高算力的支持下，运行的也越来越流畅。但算力增长的同时，续航却变的越来越差，发烫也日益严重。因此 CPU 的自身性能和调度方式，是用户体验是否满意的一个主导因素。

传统的 CFS 调度，主要是面向桌面系统 / 服务器的调度策略，可以更好的兼顾交互应用与系统吞吐量之间的平衡。属于性能优先的调度策略，通过把任务近乎平均的分配到系统所有可用的 CPU 上，最大限度地提高系统的吞吐量，不会过多的考虑到系统的耗电问题，设计时主要针对的是 SMP 系统。CFS 并没有充分利用各个核的能效比、频率差来平衡性能和功耗，自然也不适用于有交互界面、场景复杂的手机操作系统。2012 年前后，ARM 推出了 big.little 处理器架构，来优化手机日常使用中的 CPU 能效。随后，Linaro 与 ARM 联合提出了 EAS (Engry Aware Schedule) 调度算法，通过对原有的调度策略（选核，负载追踪，均衡等）做重新设计，打通了 CPU idle 管理子系统，CPU freq 子系统与调度之间的关联，以调度为核心，让 CPU 状态管理相关的几个模块紧密协作在一起，目标是在保证系统性能的前提下尽可能地降低功耗。

因此，CPU 调度子系统做为操作系统中的核心子系统，对任务资源调度平衡起着关键作用，需要通过一系列先进的技术来管理、调度和优化 CPU 资源，以提供高效的计算和响应能力。小米澎湃 OS 资源调度子系统目前主要包括关键任务的识别染色、焦点计算和 SOC 一体化调频等技术，从而实现了对 CPU 资源的能效管理和优化，有助于确保系统的稳定性和性能，满足各种算力需求。

4.1.2 子系统架构

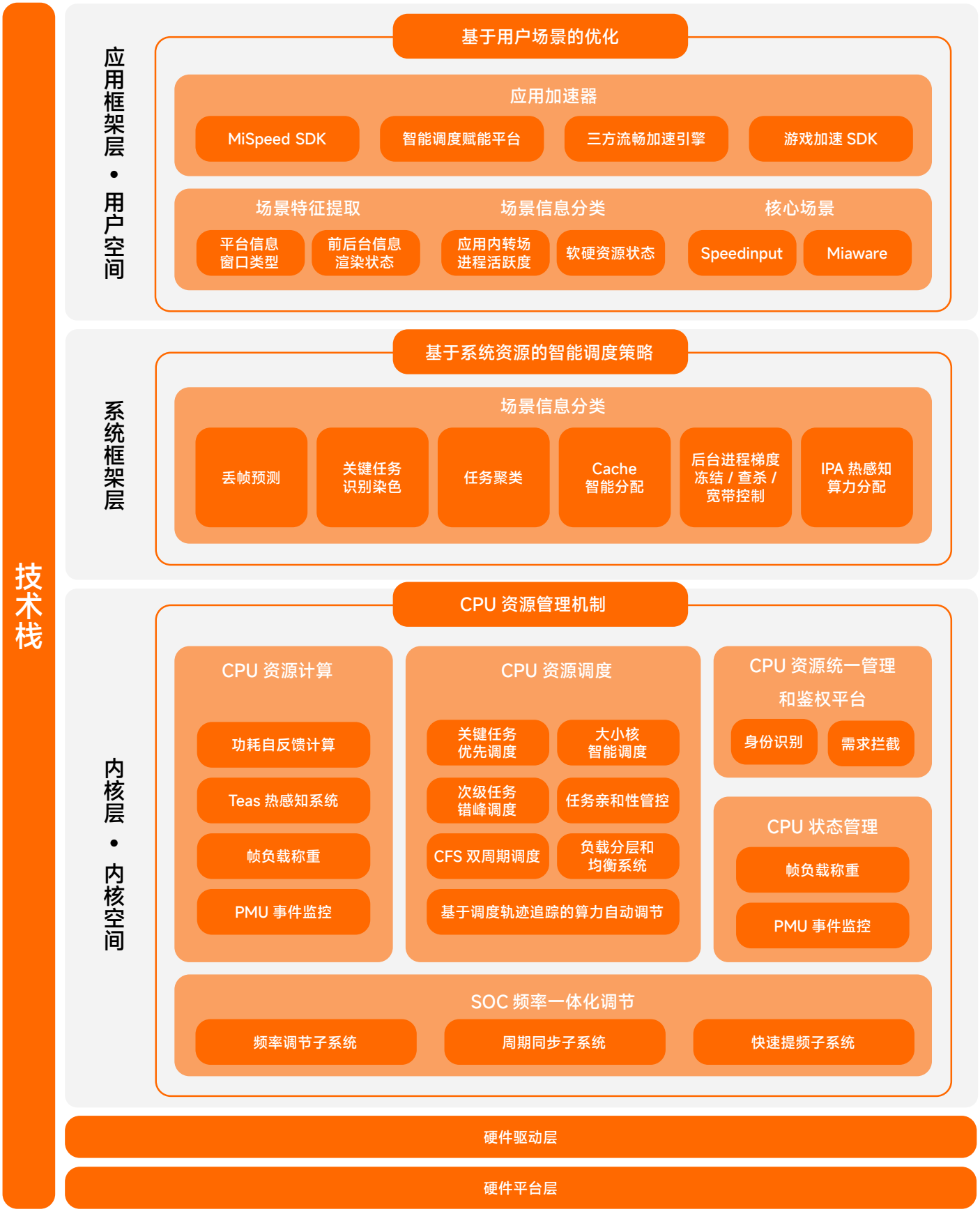


图 4.1-1 调度子系统整体架构

4.1.3 重点特性

4.1.3.1 关键任务识别染色

技术介绍

手机在使用过程中，会有成千上万个任务处于活跃状态，这些任务当中，有些是前台任务，有些是后台任务，将前台任务中会影响到渲染绘制时长的任务识别成关键任务，通过对关键任务标记，根据任务的阻塞状态和阻塞原因，对关键任务依赖的普通任务进行追踪染色处理，从而实现链式优先调度，带宽控制，错峰对齐等操作。

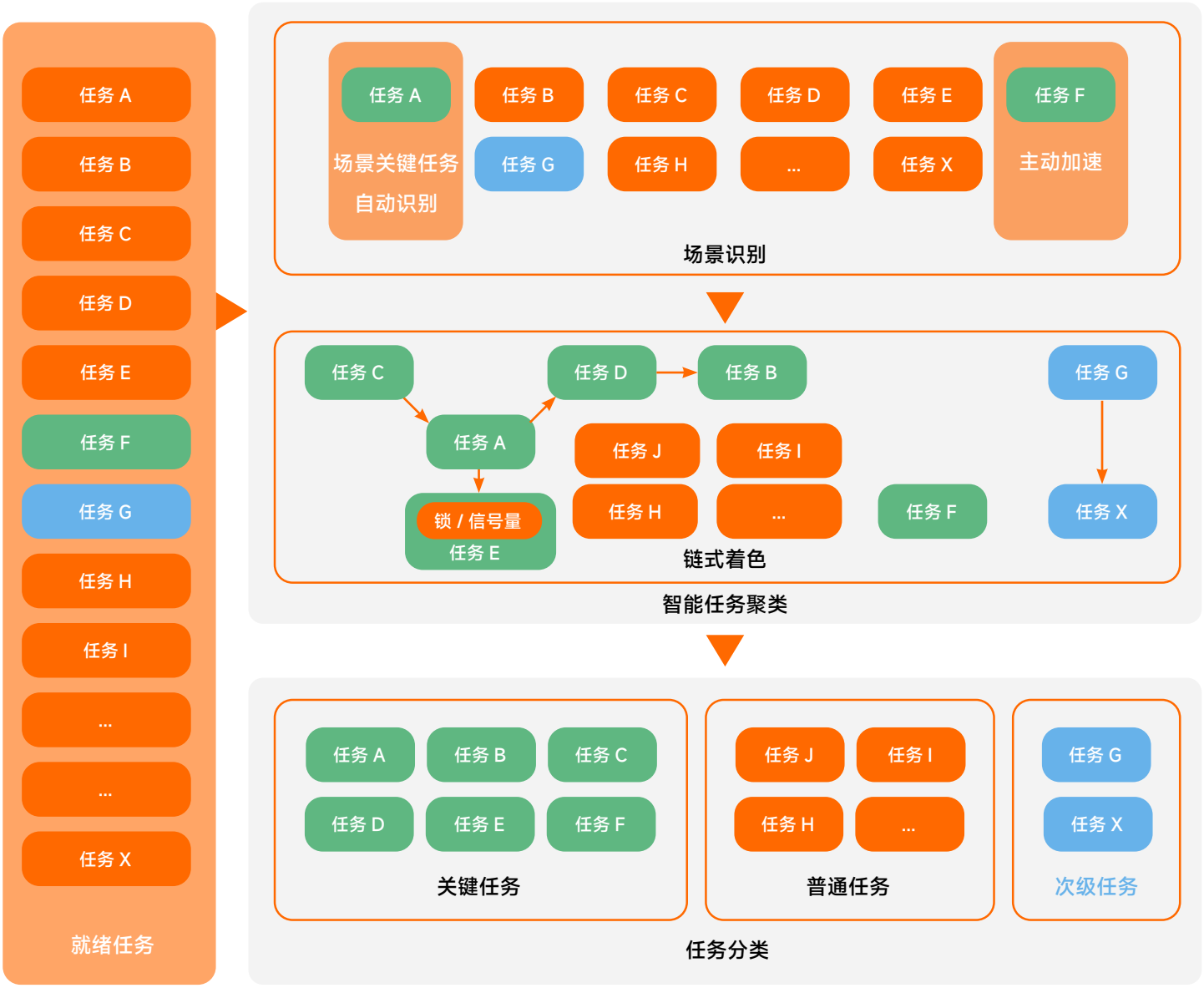


图 4.1-2 关键任务识别染色机制示意图

技术架构

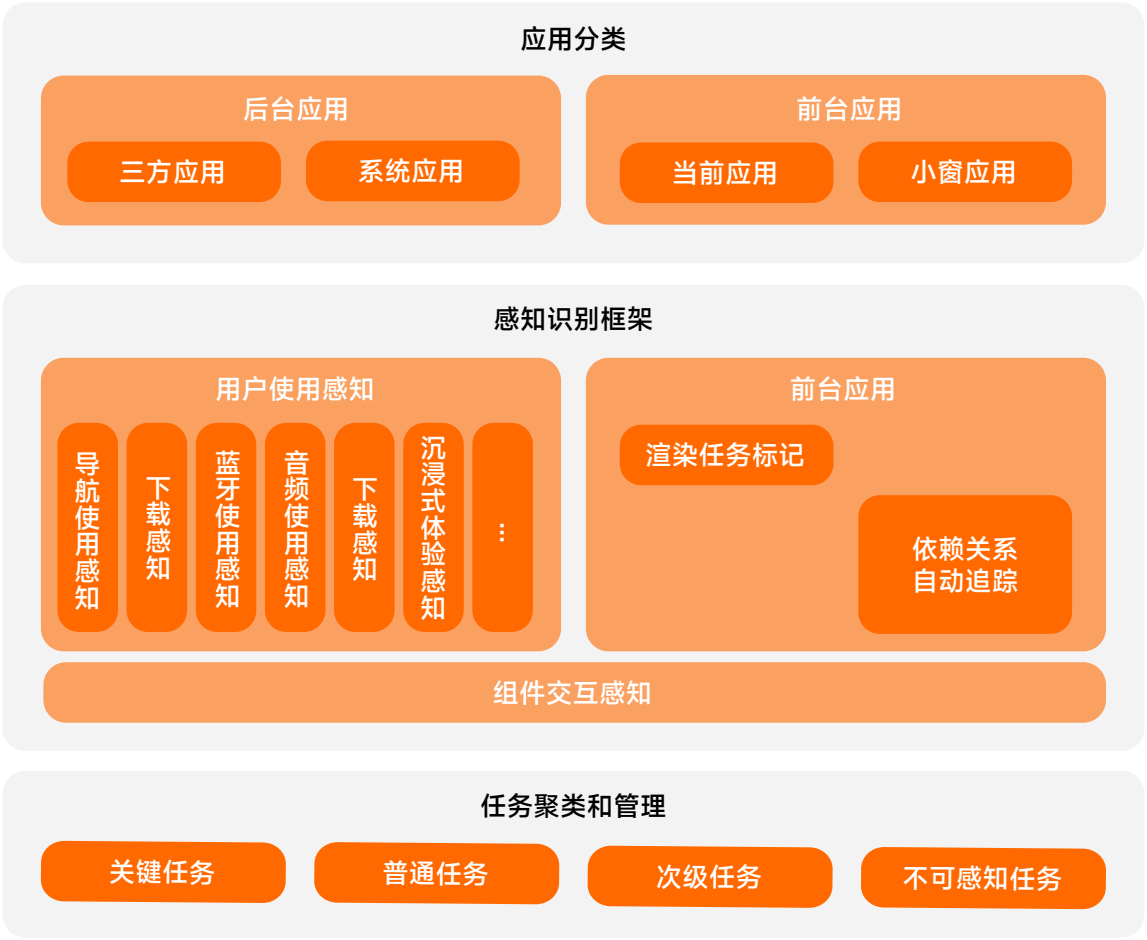


图 4.1-3 关键任务识别染色技术架构

关键技术

1. **关键任务的链式识别：**资源调度子系统能够通过任务的阻塞状、阻塞原因识别出渲染过程中的焦点任务转移路径，动态标记当前影响渲染的任务为关键任务，并优先供给系统资源，达到渲染全链路加速的效果。资源调度子系统针对不同的依赖类型进行不同的标记传递。不局限于静态的管控单一任务，而是通过追踪任务依赖，动态的管控整个渲染链路。
2. **次级任务识别和管控：**资源调度子系统针对剩余的普通任务进行进一步的精细化识别，识别出其中对用户感知以及对于系统体验影响较小的次级任务。并针对次级任务进行带宽控制，错峰调度。
3. **不可感知任务识别：**通过对用户场景的识别和对系统资源使用状态的探测，评估每个任务的用户感知程度。通过组件交互信息的拦截和梳理，确定活跃应用的关联任务范围，从而识别出系统中的用户不可感知任务，进行资源使用管控。

4.1.3.2 丢帧预测模型

技术介绍

丢帧预测模型主要是基于渲染链路各关键节点的负载变化差异以及核心应用丢帧特征进行丢帧预测，为了适用更多应用，需要对更多常见的丢帧特征进行学习和训练，以此提高丢帧预测的准确性。

技术架构

图 4.1-4 丢帧预测模型技术架构

关键技术

1. **渲染耗时感知模块：**资源调度子系统收集画面渲染过程中的事件、场景、帧率和画面信息。通过采集到的信息能够勾勒出渲染的整体情况，并且统计出渲染各个过程的耗时情况以及资源消耗情况，从而能够识别出渲染过程中的资源瓶颈。
2. **丢帧预测模块：**通过采集到的渲染信息、耗时以及资源消耗情况，预测出下一帧发生丢帧的概率，针对高概率场景，提前做系统的资源的预先供给或算力补偿。
   基于经验学习的应用丢帧特征识别，通过对高丢帧特征的实时监控以尽可能早的预测出任务的负载变化，进而提高预测精度。

4.1.3.3 帧负载称重

技术介绍

传统的 CPU 负载计算方式，无法准确评估每一帧之内关键任务的真实负载，因此可能会导致负载评估不准而引起的 CPU 频率调节不精确的问题，进而导致性能不足或者功耗的浪费。通过研究发现，即使限制 CPU 频率，仍可以保证游戏帧率满帧，说明当前系统平台上直接计算得到的频率并不准确，存在算力供给过剩的情况。进一步分析发现，当前系统采用的负载统计方法中，某些计算进入负载的任务对用户感知影响并不明显，但会极大的拉高 CPU 的频率。例如对于游戏来说，只要保证游戏的渲染相关线程在当前帧结束前运行完成即可，这样就可以保证游戏流畅度。

为了能够解决这些问题，资源调度子系统自研了帧负载称重方案，在精细化管控和识别精度上更近一步，以降低系统不必要的功耗为基础，提高 CPU 日常使用过程中的能效。

技术架构

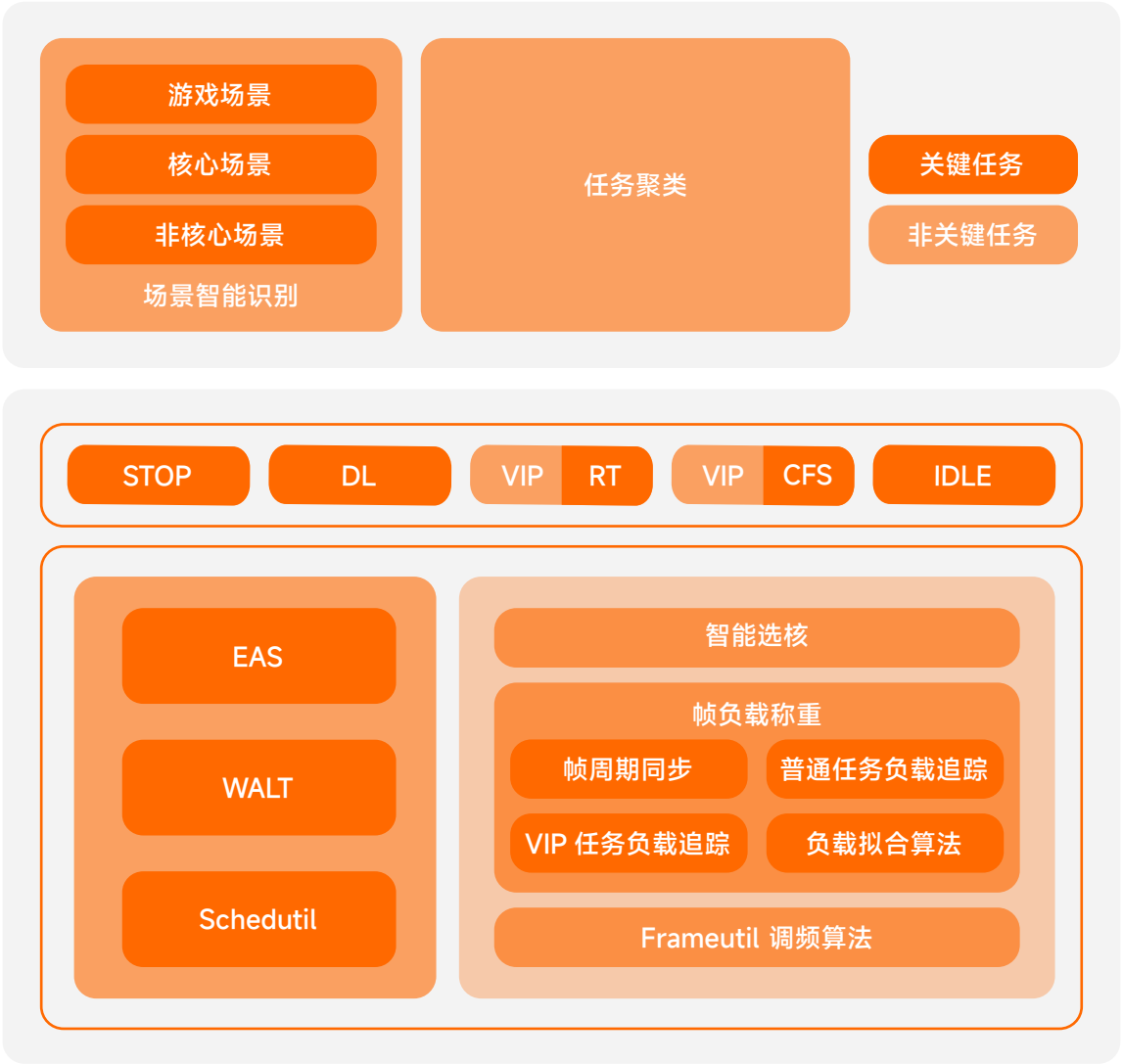


图 4.1-5 帧负载称重技术架构

关键技术

1. **渲染链路上的负载追踪技术：**一帧的绘制过程，是由多个任务协同完成的。而任务在协同过程中，会产生前后依赖关系。因此，渲染链路上的负载追踪技术，主要是识别任一时刻从事绘制事件的焦点任务程序片段，并计算该段时间的负载。从而根据依赖关系，精确的跟踪渲染链路中的实际负载。
2. **普通任务负载：**与帧绘制流程无关的任务或程序执行片段，所产生的负载将单独计算。
3. **负载拟合算法：**对不同种类的负载进行拟合，确定 CPU 的算力需求，从而发起按需调度请求。

4.1.3.4 焦点计算

技术介绍

随着安卓应用的发展种类呈现出多样化的趋势，现在越来越多的应用为了提高自己的响应速度或者达成自己的商业目的，会在后台进行未经授权的运行，并且存在滥用系统资源的风险。

图 4.1-6 应用的特立独行行为

技术架构

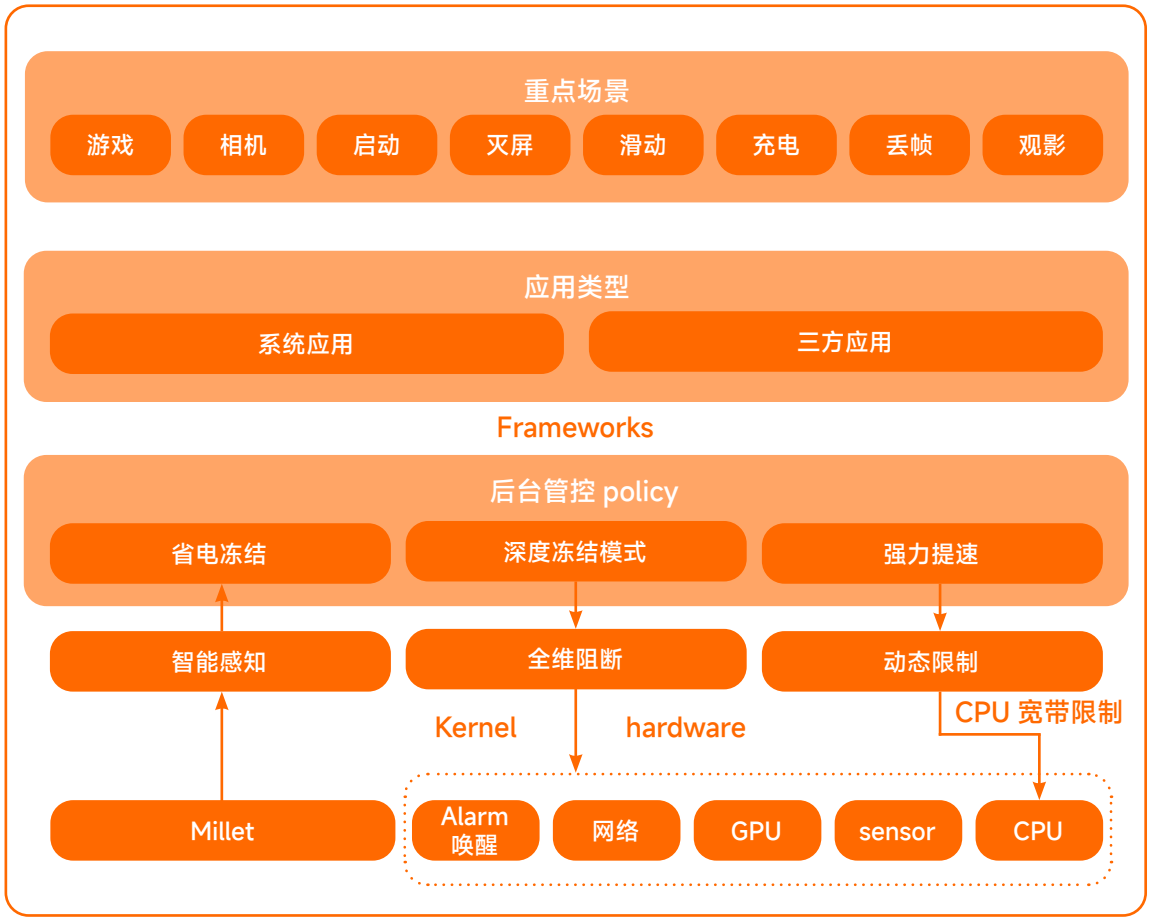


图 4.1-7 焦点计算技术架构

Linux 原生的进程冻结机制，可以将活跃状态的任务进行冷冻处理，暂停并保留其执行上下文，以便后期解冻时快速恢复。该方式可以增强系统对后台应用的行为约束和管控，是一种强有力的系统管控策略。但由于系统中，任务间的依赖关系非常复杂，进程一旦被冻结后无法接收或处理外部消息（比如，手机实时通讯 apk 在后台运行中被冻结，会导致无法接收网络信息包，最终令客户无法收到通讯消息），会给用户体验带来非常多的负面影响。

为了限制后台应用对系统资源的占用，减少其运行过程中对前台应用分配资源的争抢和干扰，同时不影响用户对后台应用的功能使用，设计了一套全新的冻结解决方案。该方案以后台三方应用资源管控为中心，对后台任务进行冻结，降低其活跃度，同时针对各路试图唤醒被冻任务的子系统进行拦截和管控，是一个典型的围点打援类的系统级大型优化方案。同时，抑制活跃度，降低其对 CPU 带宽的占用，也会使前台应用收益，从而增加其流畅度。

新一代冻结技术在更多的细分场景中进行策略差异化定制来提升优化效果和减少冻结异常概率，同时要将冻结方案的范围从 CPU 端扩展到传感器，音频等子系统，通过各子系统资源的后台联合管控和优化，提升和改善用户体验。

关键技术

1. **极限续航模式：**极限续航模式下的深度冻结方案是独立的后台管控方案，且只在极限续航模式下才会生效，非极限续航模式下使用默认冻结方案管控后台应用。极限续航模式深度冻结：首先通过极限模式上层监听到的设置变化来通知冻结控制中心，系统已进入极限续航模式，需切换管控方案以进行深度冻结模式，执行更严格的管控。其次，针对极少数必要场景条件进行解冻放行，而对于频繁解冻的应用采取查杀策略，其余绝大多数非必要条件下的解冻需求，则推迟至极限续航模式退出后再解冻。
2. **前台服务的可感知性探测和管控：**对影响用户使用的可感知的前台服务，比如音乐、导航、下载等，不进行限制；而用户不可感知的前台服务，统一进行后台管控限制。比如用户在全屏观影或者上网课的情况下，属于前台高度关注场景，此时对后台的 app 的行为进行深度冻结限制，只保留必要的解冻唤醒，降低后台功耗的同时减少后台应用对前台应用的运行干扰。
3. **性能强力提速：**一些高负载的手机使用场景，CPU 负载大，如果此刻后台应用频繁的唤醒和执行，会抢占前台应用的 CPU 带宽，加重前台运行时的丢帧。通过对后台应用进行短时间的严格冻结，来保证前台应用对 CPU 的占用不受后台应用的争抢，提高前台流畅度。

4.1.3.5 SOC 一体化调频

技术介绍

一帧的绘制完成，是需要多个硬件部件相互配合完成的。其中，CPU 是主控中心，GPU 和 DDR, Cache 等属于配套资源。当 CPU 绘制完成，将顶点 / 纹理等数据提交到 GPU 进行渲染或合成时，焦点将转换到 GPU 上。当 CPU 发起文件读写请求时，I/O 就会成为影响流畅度的主角。因此，多部件之间如何高效一体配合来减少掉帧，是该方案的优化主题。

现有的 DCVS (Dynamic Clock and Voltage Scaling) 和 DVFS (Dynamic voltage Frequency Scaling), 都是采用基于周期采样的负载预测调频算法。试图用前一周期的负载，去预测下一周期内的负载变化，从而导致频率变更需求通常要有一个采样周期的滞后性。而通过大量的帧绘制流程分析，我们观测到 CPU 作为帧绘制流程的主控方，往往能提前预判该帧绘制过程对其它从设备的资源使用需求，从而指导相应设备做出快速的频率变更响应，减少丢帧。

技术架构

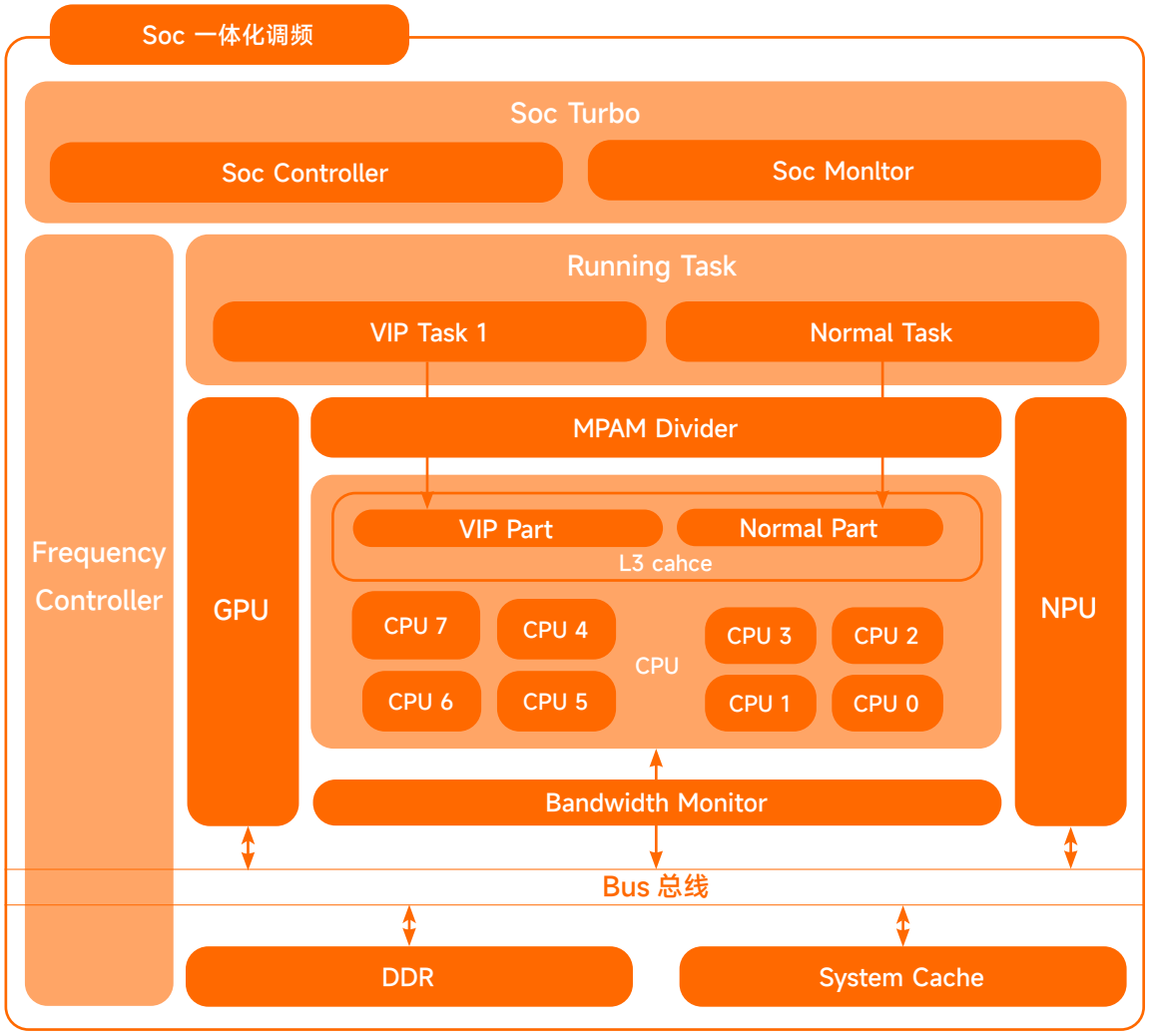


图 4.1-8 SOC 一体化调频技术架构

关键技术

1. **帧负载同步技术：**协调 CPU/GPU/L3 cache/DDR 等子系统的调频周期，使其与帧绘制周期同步。
2. **算力补偿和快速提频：**当丢帧等行为发生时，联合诊断丢帧原因，对产生资源需求瓶颈的子系统做快速的算力拉升和调配。
3. **SOC 一体化频率调节：**通过精准追踪 CPU 的流水线指标动态变化趋势，及时预估负载变化对 CPU/GPU/L3 cache/DDR 等子系统的算力新需求，并做出统一部署。通过精密计算不同策略的能效，选择最优算力调配方案，提升系统能效。

4.1.3.6 智能调度赋能平台

技术介绍

为了将目前小米在 SoC 调度领域的能力拓展到更多场景，并且帮助应用开发者根据实际需求来提升其在小米澎湃 OS 上的性能体验，基于目前资源调度子系统的能力，开发了智能调度赋能平台。

技术架构

图 4.1-9 智能调度技术架构

关键技术

1. **关键任务优先调度：**传统的调度策略，对外主要提供了 CFS 和 RT 两种调度策略，实时线程用于处理一些实时性要求高的系统级短时任务，普通任务用于承载应用的绝大多数业务逻辑。但普通任务数量庞大，其中部分任务用于渲染绘制，其执行时间的快慢会影响一帧的绘制时长，从而影响到用户感知。与此同时，其它的任务运行时间的快慢用户感知并不明显。所以，为了对普通任务中的关键任务执行调度加速，故提供了关键任务优先调度功能。资源调度子系统打破了原有的 Linux 内核公平调度机制，针对标记的关键任务进行优先排队，使关键任务能够在系统极度拥塞的场景下也能及时获取 CPU 资源，从而让系统保持流畅运行。

图 4.1-10 关键任务优先调度机制示意

2. **次级任务低功耗调度：**应用开发者，可以根据业务自身的特点和开发过程中任务角色的划分，将一些用户感知不明显的任务主动接入次级任务低功耗调度接口，从而减少其对 CPU 资源的过多占用。
3. **更加安全的调度加速解决方案：**关键任务优先调度，相比于原有 RT 调度，更具有普适性。关键任务优先调度功能既不会影响系统中低优先级实时任务的运行，也可以在关键任务发生异常时及时中断其优先调度功能，实现异常探测和调度异常自恢复。同时，资源调度子系统还集成了频率异常监控机制等，防止系统出现过热或 CPU 资源异常无法释放的情况。
4. **基于任务调度轨迹追踪的精准提频方案：**智能调度避开了原生的系统提频机制，独立实现了一套精准提频机制，能够快速响应提频和取消提频的指令。避免了原有提频机制可能会被阻塞而推迟提频的问题，并且追随关键任务的调度轨迹，只对其所在的 CPU 做算力提升和动态迁移，从而做到精准提频。

4.2 AI 子系统

4.2.1 子系统简介

AI 子系统承接小米澎湃 OS 中包含大模型能力在内的各类 AI 能力、以及感知 / 理解 / 推荐（即智慧能力）的提供，为整个系统的“智能大脑”，是小米澎湃 OS 的最重要组成部分之一。AI 子系统不仅可以让单设备实现极强的端侧 AI 能力，同时赋予整个生态智能能力。

4.2.2 子系统架构

AI 子系统分为：AI SDK，AI 能力中心、AI 智慧中心、AI 融合部署框架四个部分，如下图所示：

图 4.2-1 AI 子系统技术架构

其中，

- AI SDK 是系统 AI 能力集中管理框架，并提供智慧能力服务，为应用程序提供统一的调用接口。
- AI 能力中心将包含大模型能力在内的各类 AI 相关能力集中至系统侧，完成对 AI 能力的统一管理与发布。
- AI 智慧中心主要围绕用户的高频刚需场景构建建议能力，通过学习用户在小米澎湃 OS 各端的使用习惯，识别用户当前所处场景及需求，从而进行信息或服务的推荐。
- AI 融合部署框架用来支持能力中心和智慧中心中的各类 AI 算法在多端设备上的运行。

4.2.3 重点特性

4.2.3.1 AI SDK

技术介绍

AI SDK 是 AI 子系统的重要组成部分，是包含大模型能力在内的系统 AI 能力集中管理框架，并且提供智慧能力服务。框架内部采用模块化设计，使得 AI 能力提供方能够灵活地接入，同时为 AI 能力需求方提供统一、可靠、高效、丰富的 API 接口，同时也提供了智慧能力的系统服务，助力应用程序实现 AI 相关的复杂功能。本框架可分为 API 层、引擎层、管理中心、能力中心和智慧中心等核心组件，后续章节会有更详细的阐述。

技术架构

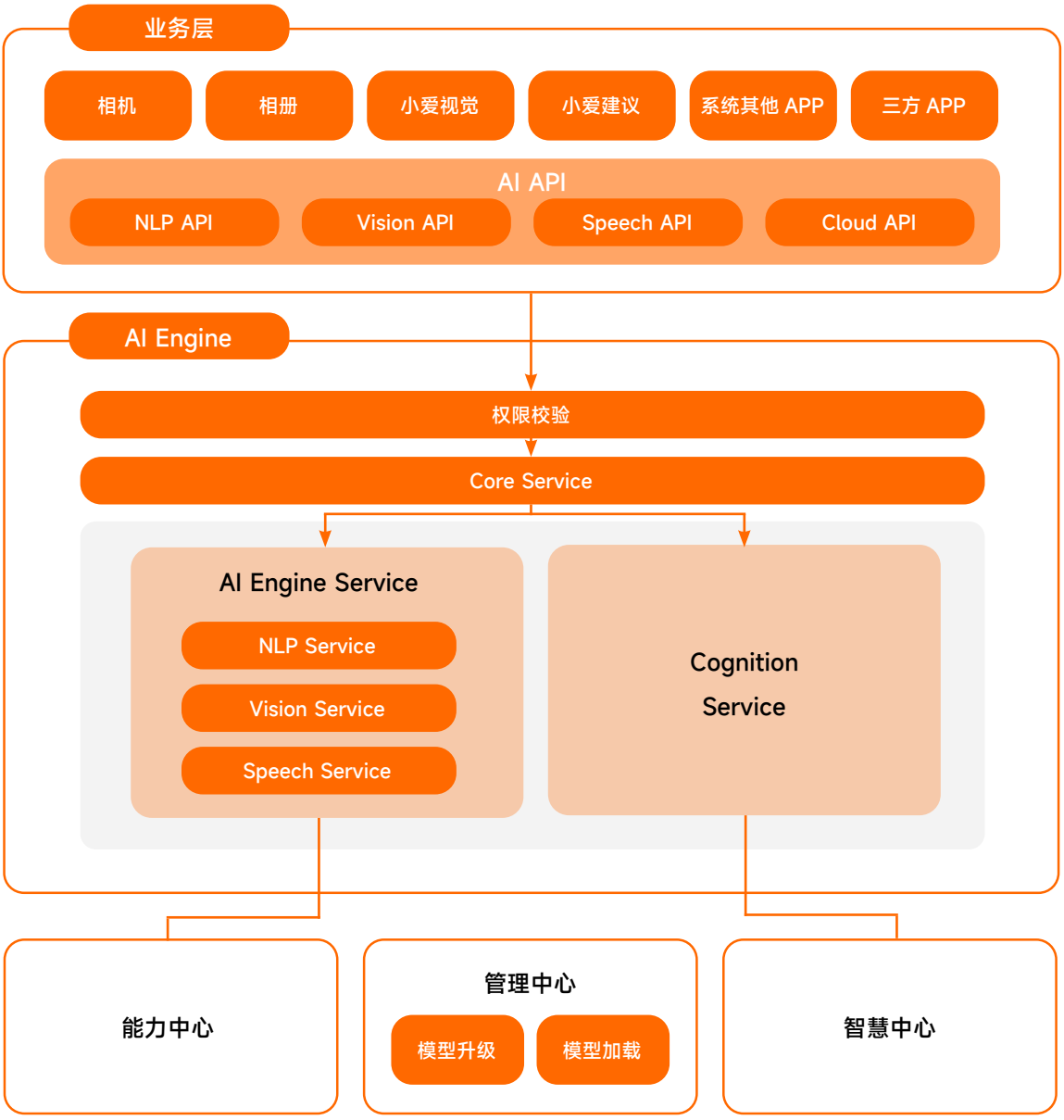


图 4.2-2 AI SDK

AI SDK 的主要组成部分如下：

- **业务层 API**
  包括用于与 AI Engine 进行通信的一组 API。这些 API 提供了开发者与 AI Engine 交互的接口，使他们能够请求不同的 AI 功能和服务。
- **AI Engine**
  AI Engine 是一个独立的系统服务，负责实现 AI 功能的处理和计算。
  AI Engine 包含以下关键功能：
  - 功能处理：接收来自业务层 API 的请求，执行相应的 AI 任务，如图像分析、语音合成等。
  - 结果返回：生成处理结果，并将结果返回给业务层 API，以便应用程序使用。
  - **AI Engine 中包含以下部分：**
  - NLP Service：自然语言处理服务，负责加载 AI 能力中心里自然语言处理模块中的能力。
  - Vision Service：图像处理服务，负责加载 AI 能力中心里视觉图像模块中的能力。
  - Speech Service：语音处理服务，负责加载 AI 能力中心里语音处理模块中的能力。
  - Cognition Service：认知模块，负责加载智慧中心的智慧能力。
- **管理中心**
  - 模型加载：负责加载所需的 AI 模型，这些模型用于不同的 AI 任务。
  - 模型升级：具有升级模型的功能，以确保应用程序使用最新的 AI 模型，从而提高性能和精度。

AI API 承接应用请求与 AI Engine 进行交互，在 AI Engine 中找到对应的 Service 并加载 AI 能力中心及 AI 智慧中心对应能力实现进一步的处理和计算。

关键技术

1. **实时性**
- 低延迟： 快速响应用户请求，如实时翻译、虚拟现实交互等。

2. **安全性**
- 数据隐私： 保护用户的个人数据，确保其不被滥用。
- 模型保护： 防止模型被恶意修改或盗用。

3. **可扩展性**
- 多平台支持： 跨 Android 和 Vela 等多个平台使用统一的 AI 能力。
- 模块化设计： 允许应用程序选择性地集成和使用 AI 功能。

4. **自适应性**
- 学习能力： 支持模型更新，以适应不断变化的用户需求和环境。

4.2.3.2 AI 能力中心

技术介绍

AI 能力中心是一个集成在智能设备上提供多种 AI 能力的服务中台。它将系统中散布各类 AI 相关能力集中至系统侧，完成对 AI 能力的统一管理与发布，这些能力包括：大模型、视觉图像、自然语言处理等多个领域的能力。AI 能力中心能够优化用户与设备的互动，增加设备的自动化能力，并提供更丰富的功能。

技术架构

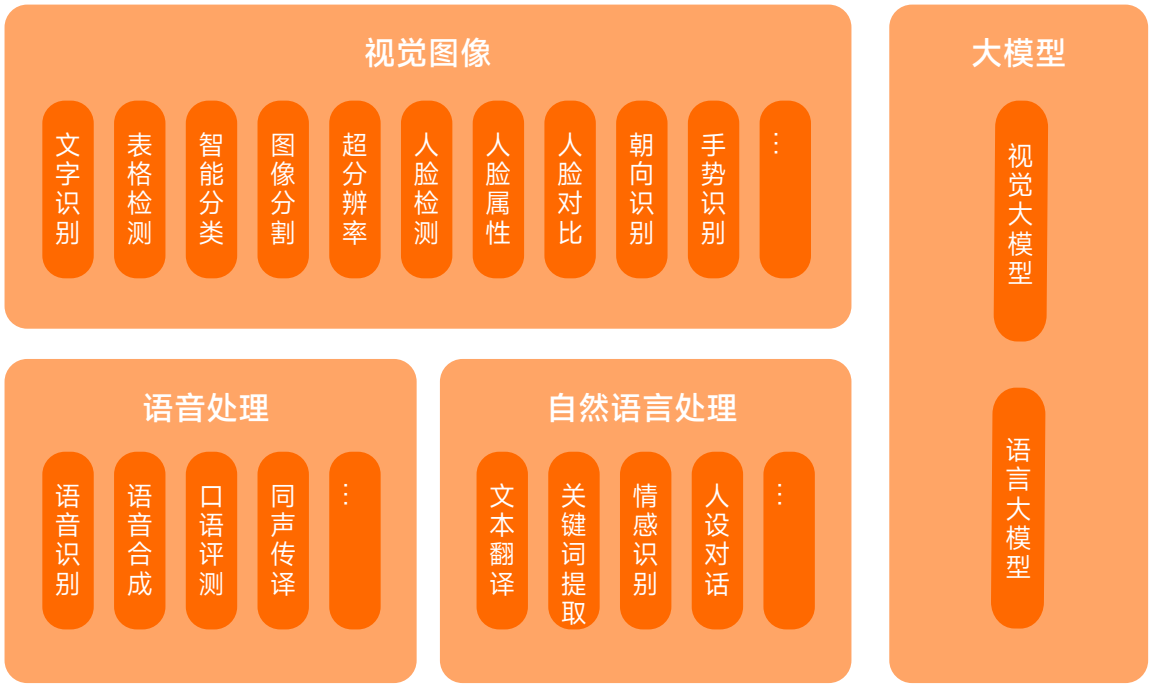


图 4.2-3 AI 能力中心架构

AI 能力中心包含大模型、视觉图像、语音处理、自然语言处理等模块：

- **大模型**
  整合了视觉大模型和语言大模型能力，能够将用户输入的图片、视频、文字等信息进行精细化理解和整合。
- **视觉图像处理模块**
  提供超分辨率、图像分割、人脸检测等基础能力，可实现图像质量改善、增强特定功能、识别人脸或场景等一系列视觉识别、增强功能。
- **语音处理模块**
  处理和分析用户的语音输入，以识别语音指令、转录文本、实现语音识别、合成语音和同声传译等功能。
- **自然语言处理模块**
  提供各种处理和分析文本和语言数据的基础能力，以使智能设备能够理解和生成自然语言文本。

关键技术

AI 子系统封装并提供大模型能力，包括：

1. **视觉大模型：**

- AI 写真：根据用户提供的个人图片生成各种不同场景的用户个性化图片，创造出各种有趣和独特的效果，满足用户的多样化需求。
- AI 妙画：根据用户的涂鸦和输入的文字描述作为创作材料，生成各种风格和主题的精美图像，让用户能够将自己的创意转化为图画作品，无需专业绘画技能。
- AI 搜索：帮助用户通过文本描述来查找其手机中相关内容的图片，用户能够使用文字输入来快速找到智能设备中的图像资源，提高图像管理和检索的效率。
- AI 编辑：实现帮助用户扩展图像内容、替换图片中的特定物体等功能，改进图像质量或满足用户特定需求。

2. **语言大模型：**

- 意图理解：根据用户交互的上下文，包括对话历史和语境，更好地理解用户意图和情感倾向。这使它们能够提供更具连贯性和准确性的回应。
- 知识问答：基于广泛的知识库进行了训练，涵盖了多个领域的信息。它们可以回答各种主题和领域的问题，从历史事实到科学知识。并实时从互联网获取信息，以回答当前事件和趋势相关的问题。
- 工具调用：它们可以用于自动执行各种任务，如发送电子邮件、创建日程事件、提供天气信息、翻译文本等。用户可以通过自然语言自动化地调用工具，以完成日常任务。

4.2.3.3 AI 智慧中心

技术介绍

AI 智慧中心是一个多层次、多模态的智能系统。基于终端设备丰富的传感器和强大的计算能力，AI 智慧中心可以感知用户身边发生的事件，以及设备操作记录。旨在为用户提供更加智能化、个性化的服务。它包含四个关键模块：原子化感知、多模态融合感知、智慧理解以及应用服务。每个模块都扮演着不可或缺的角色，共同构建了终端设备的 AI 智慧生态系统。

技术架构

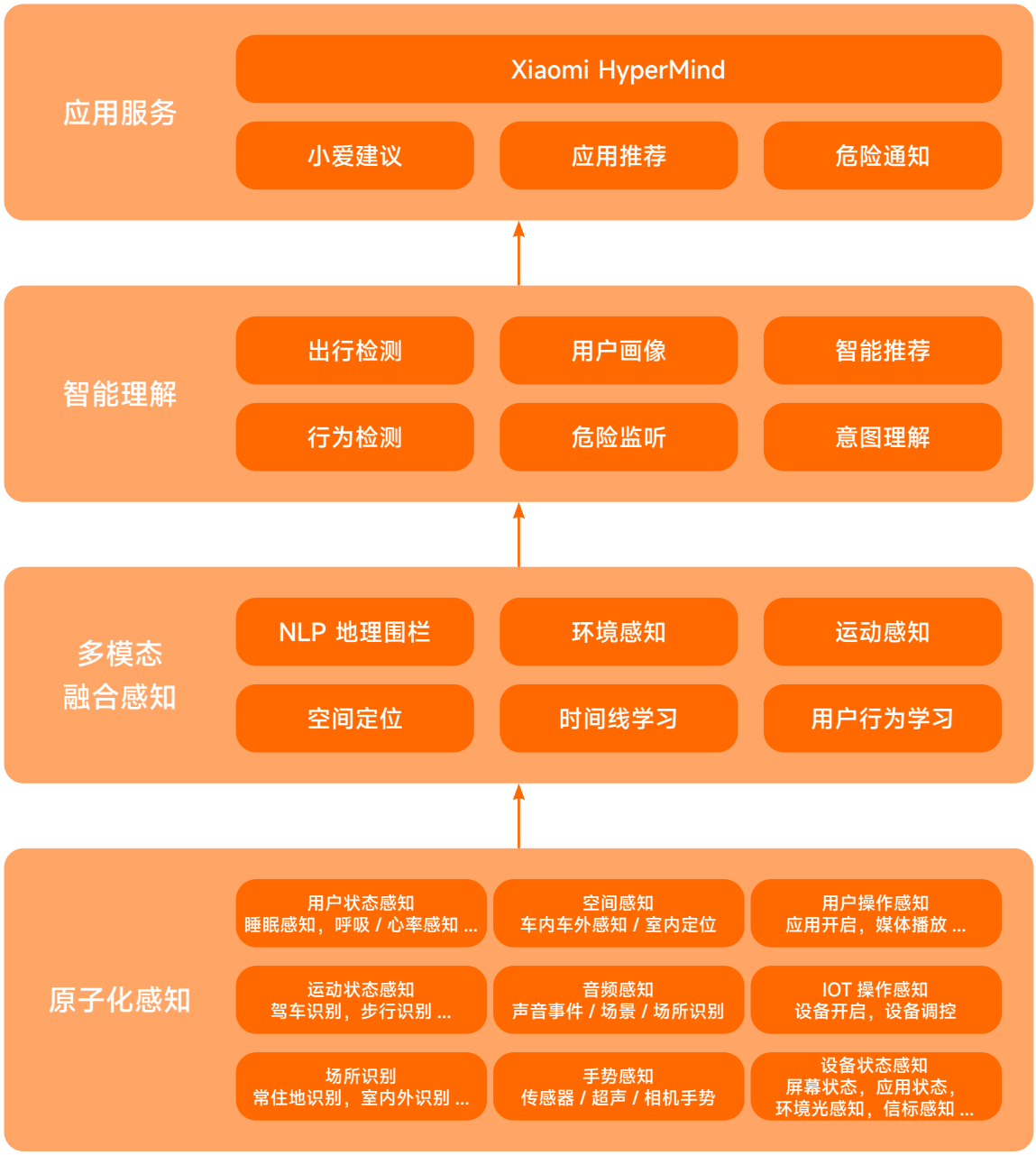


图 4.2-4 AI 智慧中心技术架构

AI 智慧中心分为原子化感知，多模态融合感知，智慧理解，应用服务四个模块：

1. **原子化感知**
   原子化感知是整个系统的基础，它涵盖了一系列手机传感器和数据源的感知能力。这包括用户状态感知，空间感知，操作感知，音频感知和设备状态感知等。这些感知数据为系统提供了关键的环境信息，以便更好地理解用户与其周围世界的互动。
2. **多模态融合感知**
   多模态融合是基于原子化感知的能力，将不同感知源的数据整合在一起，创造了更全面、丰富的感知能力。这包括地理围栏，环境感知以及用户行为学习。多模态融合允许系统更全面地理解用户的上下文和行为，为更高级的智能服务提供了关键信息。
3. **智慧理解**
   智慧理解模块通过学习和分析多模态融合感知数据，构建了用户画像，实现出行检测，提供智能推荐，以及实现对用户意图的理解。这个阶段的技术使系统能够更好地理解用户需求和喜好，从而为用户提供更个性化的服务。
4. **应用服务**
   最终，AI 智慧中心的目标是为用户提供有价值的应用服务。这包括小爱建议、习惯推荐、以及行为习惯学习与自动执行（Xiaomi HyperMind）。这些服务是系统的最终输出，旨在提高用户的生活质量和满足其需求。
   通过这个技术架构，AI 智慧中心能够实现更智能、更个性化的用户体验，提供广泛的应用服务，从而满足用户的不断变化的需求和期望。这个架构为智能终端的 AI 生态系统的未来发展奠定了坚实的基础。

关键技术

AI 智慧中心的技术架构具有以下重要特性，这些特性为其提供卓越的智能能力和用户体验：

1. **多模态感知和融合：**该架构强调了多模态感知的能力，通过整合各种传感器和数据源，使系统能够理解用户的多重上下文，包括位置、环境、行为和声音等。这为更深入的智能服务提供了丰富的数据基础。
2. **个性化智慧理解：**通过学习用户的行为习惯、喜好和意图，智慧理解模块构建了个性化的用户画像，从而为用户提供智能推荐和个性化建议。这个特性提高了用户的满意度和服务质量。
3. **Xiaomi HyperMind 行为学习和执行：**Xiaomi HyperMind 是这个架构的关键特性之一，它使系统能够学习用户的行为习惯并在用户授权的情况下自动执行任务。这为自动化生活和用户便捷性提供了新的可能性。以下是一些示例说明 Xiaomi HyperMind 如何在各种情境下提供便捷的自动化：a）音乐播放场景：当用户打开音乐应用时，Xiaomi HyperMind 可以识别用户的位置和已连接的音响设备，然后自动将音频流切换到附近的音箱，以提供更好的音乐体验。b）电话通话场景：当用户接听电话时，Xiaomi HyperMind 可以感知到用户的通话状态，并可以自动调低周围媒体设备的音量，确保用户可以清晰地进行通话，而不会被嘈杂的声音打扰。c）回家的情境：当用户回到家时，Xiaomi HyperMind 可以根据用户的位置数据和时间，自动触发一系

列操作。它可以打开家里的智能灯光系统，并将灯光调整到用户喜好的亮度。这不仅提供了方便，还创造了一个愉悦的家居环境。

4. **用户授权和隐私保护：**AI 智慧中心强调用户授权。用户可以明确授权系统执行任务，并保护他们的隐私。这为用户提供了对他们个人数据的控制，在提供智慧服务能力的同时确保其数据的安全性和隐私。

综合来看，AI 智慧中心的技术架构具有多模态感知、个性化智慧理解、自动学习和执行等关键特性，这些特性使其能够提供高度智能化和个性化的服务，同时保护用户隐私和授权权利。这为用户提供了更便捷、智能和个性化的手机体验，将来将继续发展以满足不断变化的用户需求。

4.2.3.4 AI 融合部署框架

技术介绍

AI 融合部署框架提供了一个可以应用于各种终端设备场景的统一 AI 部署框架，旨在解决业务中碎片化的框架部署、优化无法通用以及资源浪费等问题，通过软硬件深度融合、实现统一、高效的 AI 模型部署，赋能手机、平板、汽车、家电、IoT 设备、可穿戴设备等各类终端设备。

技术架构

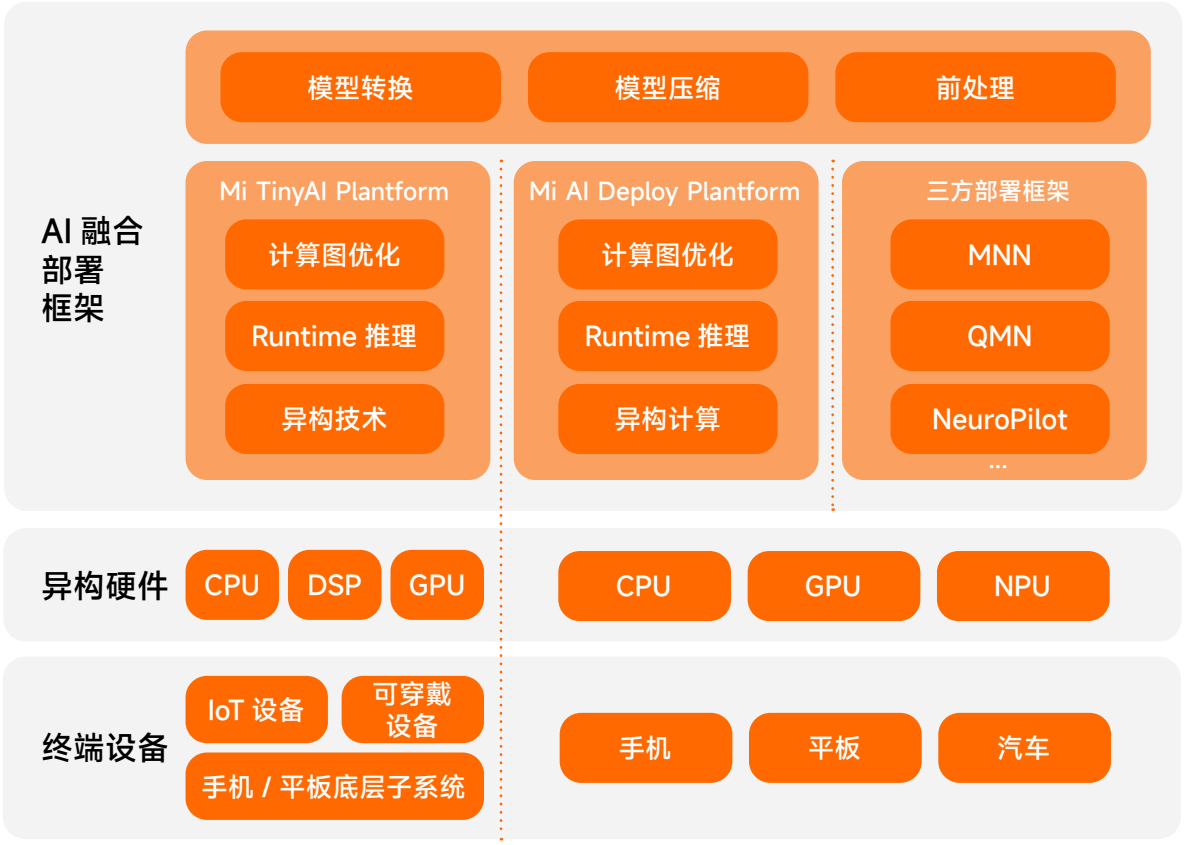


图 4.2-5 AI 部署框架技术架构

AI 融合部署框架包括通用工具和 AI 推理框架两部分：

通用工具

- 模型转换：支持 pb、ONNX、TFLite 等模型转换。
- 模型压缩：支持量化、知识蒸馏、剪枝等功能。
- 前处理：支持常用的音频、图像、视频信号处理。

AI 推理框架

- 面向小米澎湃 OS 标准设备的 Mi AI Deploy Platform，支持 CPU、GPU、NPU 异构计算，可部署在手机、平板、汽车等终端设备。
- 面向小米澎湃 OS 轻量设备的 Mi TinyAI Platform，支持 CPU、DSP、GPU 异构计算，可部署在手机底层子系统及 IoT 设备、可穿戴设备等终端设备。
- 三方部署框架，包括高通 QNN、MTK NeuroPilot、开源推理框架 MNN 等。

关键技术

AI 融合部署框架可以更灵活地选择适用于其硬件设备的最佳 AI 部署框架，提高通用性和性能。

1. **通用部分的提取和共享：**为了实现通用性，AI 融合部署框架可以提取和共享通用部分，包括模型转换工具、模型压缩算法、前处理功能等。这些通用部分可以跨不同的 AI 部署框架共享，从而减少了冗余开发和维护工作，同时确保了一致性。
2. **多硬件支持：**

   AI 融合部署框架具有广泛的硬件支持，包括但不限于：

- 面向小米澎湃 OS 标准设备，可部署在手机、平板、汽车等高性能设备平台，支持 CPU、GPU 和 NPU 等，以便在高算力平台实现高性能的模型推理。
- 面向小米澎湃 OS 轻量设备，可部署在资源受限的嵌入式系统和物联网设备，包括嵌入式 CPU、DSP、GPU，甚至边缘 AI 处理器。
- 针对小米澎湃 OS 轻量设备，AI 融合部署框架在保留核心框架和算子的基础上最大化优化了框架的空间占用，并对每一种异构硬件架构进行指令级算子优化，确保 AI 能力在这些平台上高效运行。

3. **动态硬件检测和智能选择：**AI 融合部署框架的核心功能是在部署时动态检测当前硬件设备的状态。这包括 CPU、GPU、NPU、DSP 等硬件资源的可用性和性能。根据这些信息，框架可以智能地选择最适合的系统调度策略，以确保在特定硬件设备上获得最佳性能。

4.3 图形子系统

4.3.1 子系统简介

图形图像子系统是澎湃 OS 重要组成部分，作为系统与用户间最重要的交互界面。多年来小米持续在该领域进行重点投入，从底层根技术到用户体验进行深层拆解，在多个核心方向进行自研攻关，成功打造自研视效引擎、游戏图像渲染引擎等领先技术，整体体验处于行业领先水平。

4.3.2 子系统架构

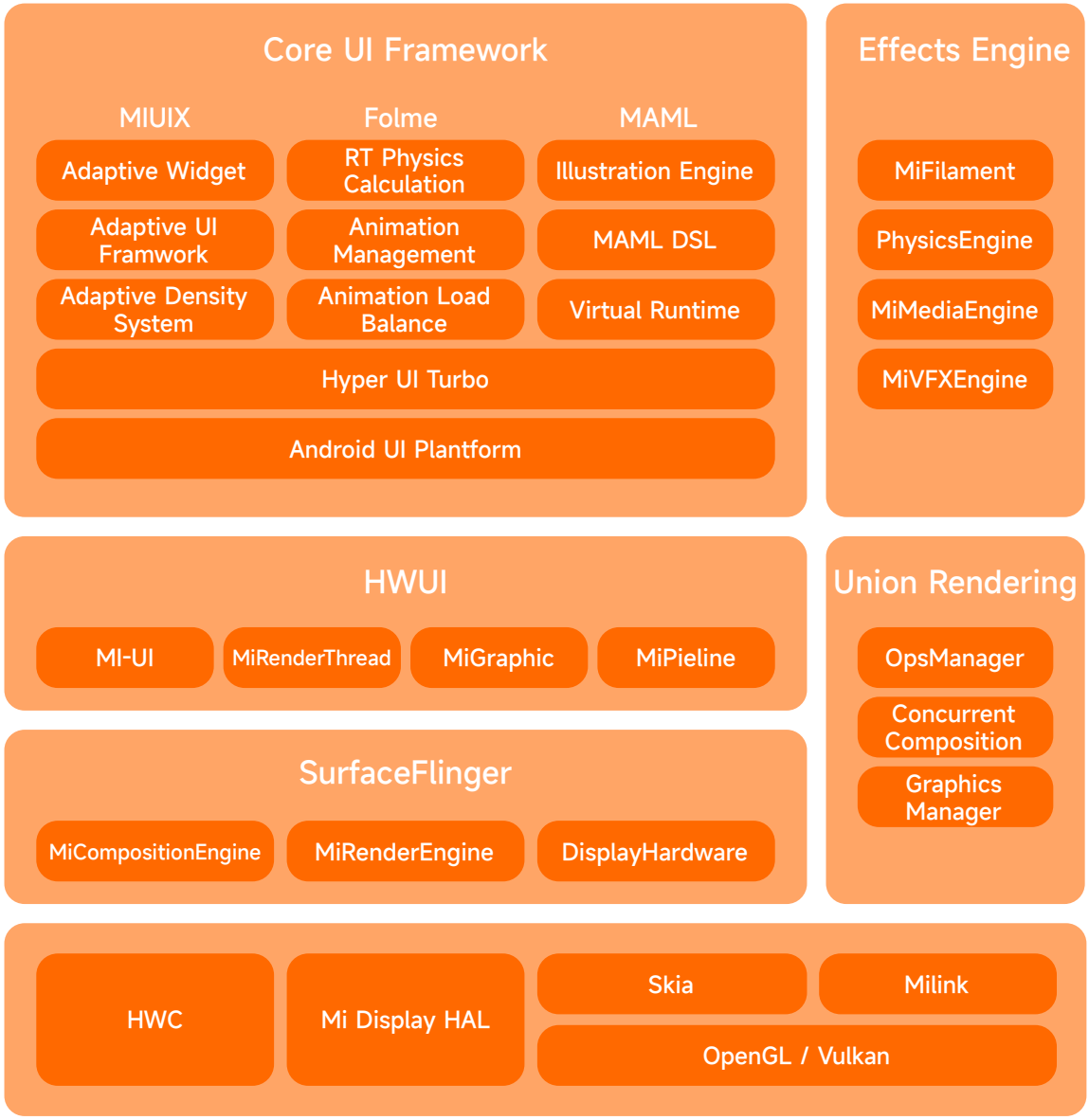


图 4.3-1 图形子系统架构

4.3.3 重点特性

4.3.3.1 视效引擎

技术介绍

视效技术是关乎系统体验最重要的因素之一，小米澎湃 OS 基于用户体验需求，结合多年技术沉淀打造视效引擎，在“系统 GUI 高效绘制”和“2D 和 3D 相融合的 GUI 特效绘制”两个方向取得长足的进展和较大的突破。

关键技术

1. **系统级统一 UI 和动画框架：**实现一次开发，多端适配的核心能力，随着应用窗口的变化，UI（布局与控件）不断的响应变化，动态适配不同类型的终端设备。
- 通过响应式规范 定义不同终端设备的 UI 展现形态
- 通过响应式支撑框架，通知 MIUIX 控件 / 布局 / 业务做形态调整
- 通过响应式 控件 / 布局库，减少业务适配代码

2. **系统动效引擎：**全局系统动效体系，动效稳定性和连续性全面领先，实现 2D 元素的 3D 空间层级渲染，多种滤镜透视效果，还原真实空间玻璃质感。
- OpenGLES 后处理（充分利用 GPU 和 VPU 计算能力）
  基于已解码数据，通过 OpenGLES 进行特效处理，从而减少动画所需的运算量
- 倒播、无级变速、逐帧播放等等能力
- 定制化场景

3. **天气特效引擎：**在体积云实时渲染领域，效果最好的是 Ray Marching 方法，它利用 3D 噪声纹理（堆叠起来的 2D 纹理），搭配使用逐像素光线步进的方法进行实时立体云形状、亮度计算。澎湃 OS 利用体积云和光照特效，还原真实天气视觉渲染，2D 文字和 3D 气象空间融合渲染。

4. **系统联合渲染：**突破应用层级限制，实现真实空间光影渲染效果。
- 设计 HWUI 新的渲染流程
  联合渲染架构
  离屏渲染并行加速，多层级 Layer 合并计算，高效完成空间计算绘制。
  渲染管线全链条优化
  CPU/GPU 动态调度策略，针对模糊混色场景合理的分配硬件资源，保证模糊混色的实时渲染。
- 设计新的合成策略
  SurfaceFlinger 合成时，根据需要对指定的窗口做相应的渲染效果，针对不同的场景满足效果的同时不影响合成效率。

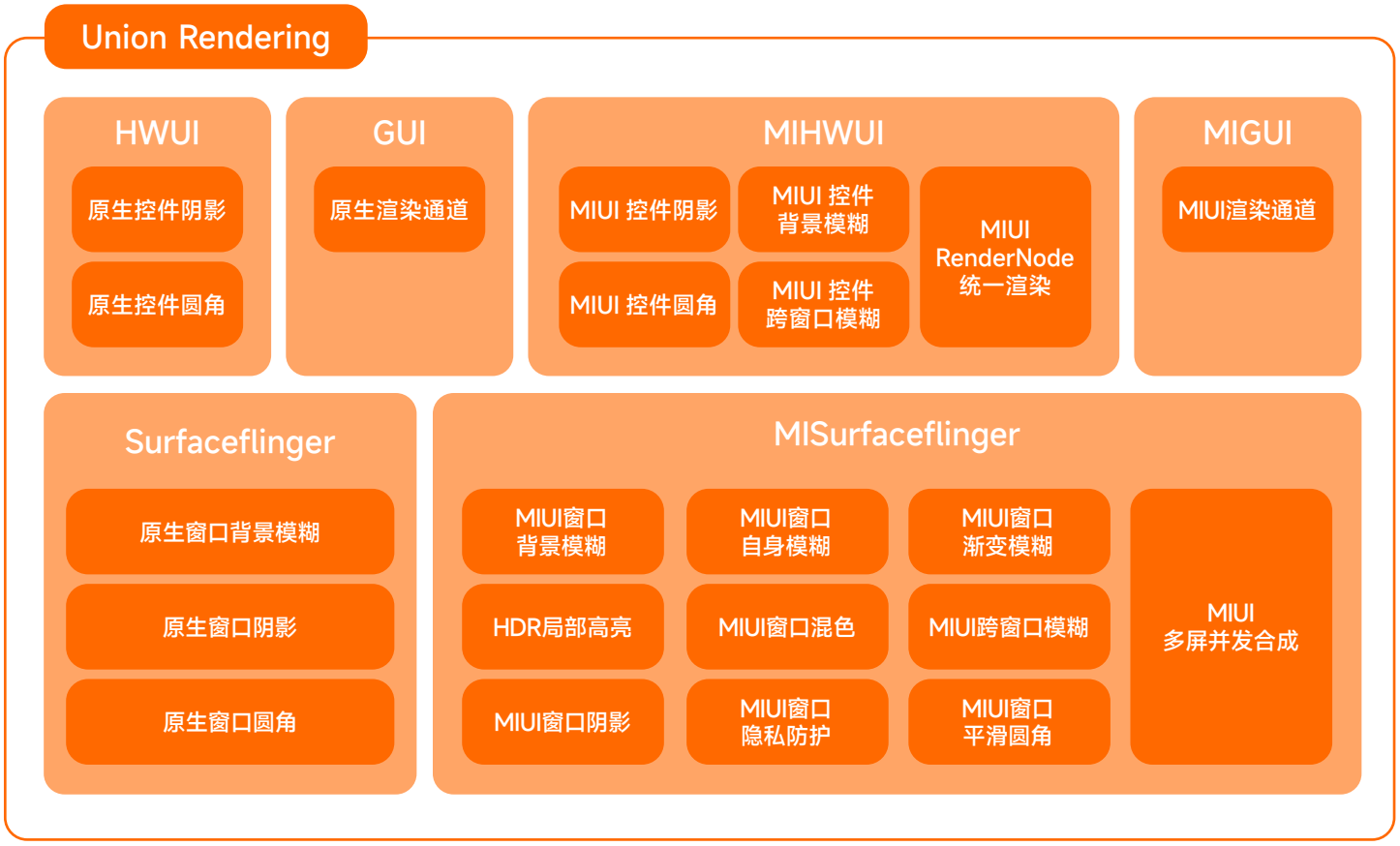


图 4.3-2 联合渲染技术架构

5. **自研图形绘制算法库 MiInk**：全局平滑圆角算法，深度模糊算法，弥散投影算法。

4.3.3.2 游戏图像渲染引擎

技术介绍

基于 Android 原生渲染架构，在 OpenGL 驱动层增加 wrapper，实现渲染指令的拦截和替换，通过对原有渲染逻辑的修改，达到优化游戏渲染负载，支持多种分辨率切换，提升画质等功能。

技术架构

图 4.3-3 游戏图像渲染引擎技术架构

关键技术

1. **GPU 增效算法方案**
   - 自研超分 / 插帧算法
   - 动态渲染分辨率。同性能的场景下，功耗优化 10%
   - 游戏渲染负载优化。通过对游戏原生 shader 进行优化，同性能场景下，功耗优化 5%
   - 游戏画质增强。大型游戏支持更高分辨率，画质体验更好
2. **自适应调度方案**
   - 创新自研动态稳帧技术，实现 CPU/GPU/DDR 联动降载
   - 与第三方合作游戏厂商合作，推出自适应调度框架，满足实时降载需求，与游戏厂商实现联合调优
3. **平台调度优化方案**
   - AI 负载预测，通过 GLES 实现游戏内主动场景识别（加载，团战，传送等），实现自动加速
   - 联合硬件平台厂商实现芯片能效的整体调优
4. **生态推进与合作**
   - 支持 Vulkan 标准及推动提升 Vulkan 覆盖率，优化内容，实现降低游戏负载
   - 与联盟推动 64 位游戏生态普及
   - 自研 Kite Profiler Tool，支持多维度游戏性能数据实时抓取和分析

4.4 网络子系统

4.4.1 子系统简介

网络子系统是通信网络中的核心部分，在小米澎湃 OS 操作系统中，网络子系统由短距子系统和蜂窝子系统组成。它们在操作系统中扮演着至关重要的角色，负责数据的传输、接收和通信。

短距子系统主要负责设备之间近距离的通信，例如 WiFi、蓝牙、UWB、NFC 等。这些技术通过空气以电波或光波的形式传递信息，覆盖范围相对较小，但可以非常高效地传输大量数据。它们常被用于无线耳机、个人电脑、智能手机和打印机等设备的连接，以及在智能家居和工业自动化等环境中的设备间通信。

蜂窝子系统则负责远距离的通信，通过无线电信号在蜂窝网络中传输信息。常见的蜂窝网络包括 5G、4G、3G 和 2G 等。这类网络覆盖范围广，可容纳更多的设备同时通信，但数据传输速度相对较慢。它广泛应用于手机、平板电脑和移动设备等，为人们提供了无处不在的通信连接。

在网络子系统中，短距和蜂窝子系统各自发挥优势，互相补充。短距子系统负责设备间高速、高效的通信，而蜂窝子系统则提供了更大范围的覆盖和稳定的连接。这样的组合使得网络子系统能够满足各种通信需求，从短距离的高清视频流到远距离的语音通话和数据传输，都能得到良好的支持。

4.4.2 子系统架构

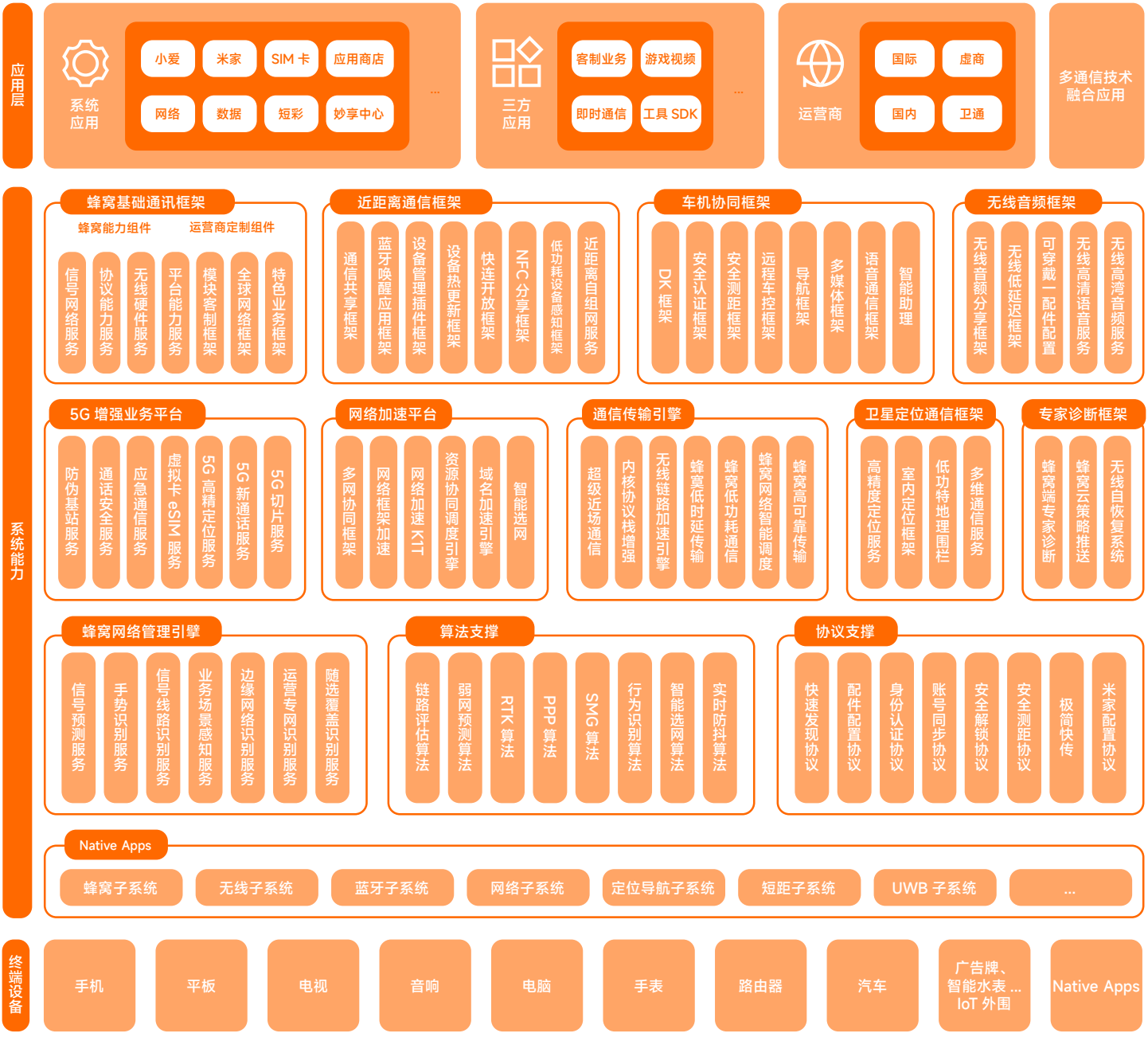


图 4.4-1 网络子系统整体架构

- **全场景信号协同加速**
  - 四网协同、弱网优化、域名加速、资源协同调度、游戏加速、微信通话加速 …
  - 超级近场、数据包聚合加速、文件互传、加速引擎、极简协议 …
  - 蜂窝全覆盖通信：边缘网络优化、运营专网加速、随选覆盖感知 …
  - 全景协作通信：通信共享、多网融合、全域通信 …
  - 智慧蜂窝专家：弱信号预测、端云网络专家
  - 增强泛 5G 通信： 5G 高精定位、5G 切片、5G-XR、5G 新通话 …
- **极速客制 KIT**
  - 蜂窝能力组件、特色安全通信、全球运营商客制 …
- **无线数字音频**
  - 高音质：192KHZ(LHDC)，900Kbps
  - 低延迟：55ms(LEA)，aptX(85ms)
  - 高清语音：32KHZ，16bit
  - 音频分享：2 套 + 高清 TWS 播放
- **近距离通信**
  - 快连弹窗、米家配置、耳机自动切换、耳机流转、小米秒享、靠近发现、小米钱包 …
  - 插件 OTA、配件固件 OTA、低功耗设备感知、近场 (NFC) 分享、近距离自组网
- **车机协同**
  - 数字车钥匙：无感开锁，无钥匙启动，钥匙分享，远程车控
  - CarWith：导航，多媒体，语音通信，智能助理认证
- **定位导航**
  - 地理围栏、高精度定位
- **卫星通信**
  - 短消息、通话、应急通信

4.4.3 重点特性

4.4.3.1 Network X

技术介绍

Network X 为短距通信领域网络加速的统称，主要包括协同加速引擎、智能选网、域名加速引擎等功能，为用户提供快速便捷的网络体验。

技术架构

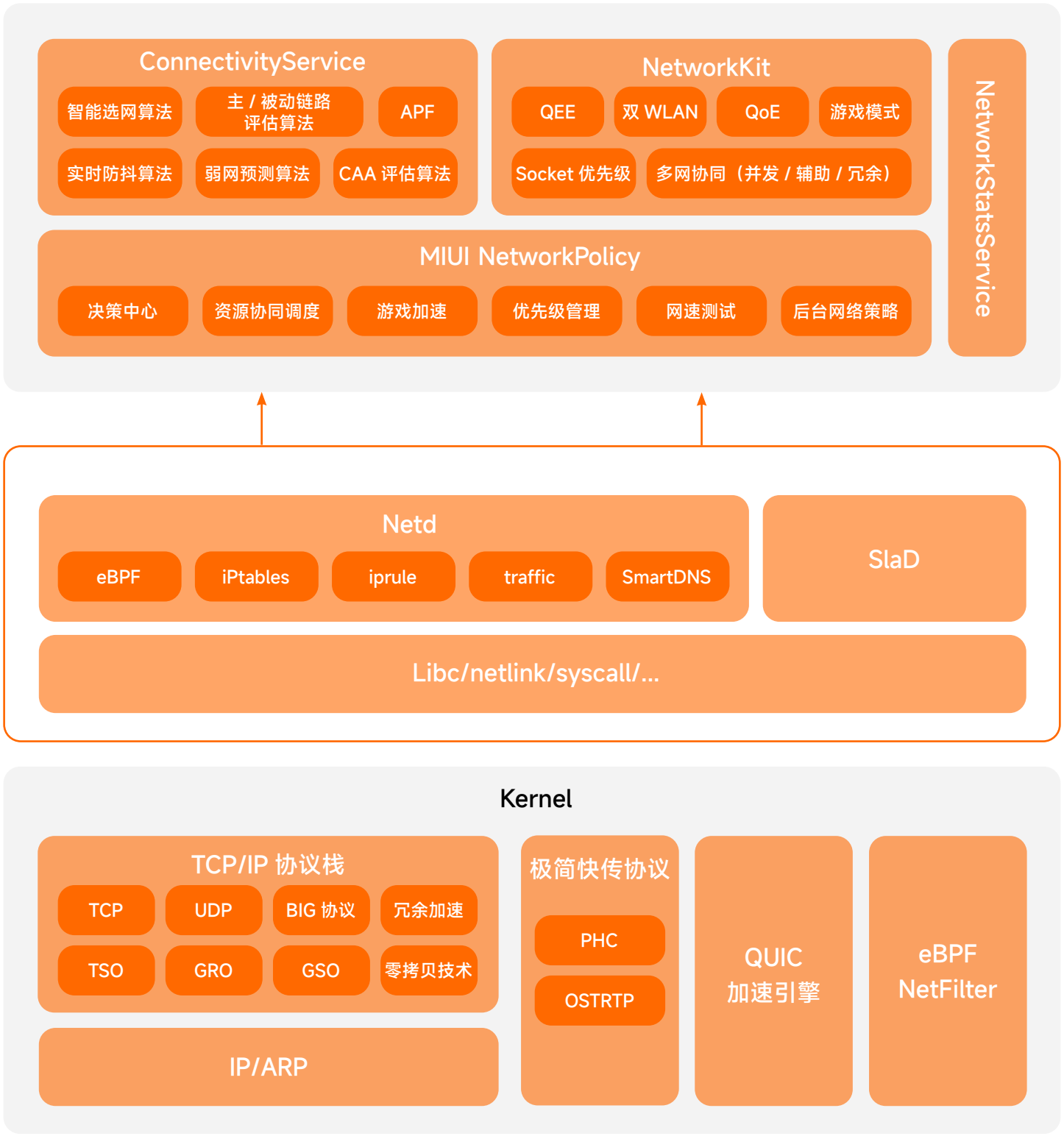


图 4.4-2 Network X 技术架构

关键技术

1. **协同加速引擎 (Matrix Engine)**
- 基于多网协同并发技术，高延迟降低 62%，最大并发带宽可达 7Gbps+
- 实时链路质量评估引擎
- 网络实时防抖算法
- 链路并发 / 冗余加速引擎
- 传感器数据运动状态进行辅助识别

2. **智能选网 (Smart Link Handover)**
- 采用主动和被动网络探测方案，实时进行链路质量评估，进入离开范围进行精准识别
- CCA 物理信道质量评估算法，精准环境拥堵和干扰信息评估，为决策中心提供依据
- 低延迟漫游，不同 SSID/ 不同密码 AP，芯片级低延迟无感漫游
- 跨子网 AP 间地址分配协议优化，加快网络配置速度
- 传感器数据运动状态等场景进行辅助识别

3. **网络资源协同调度 (Network Synergy Engine)**
   手机和路由间标准协议，路由可以依据手机发送给路由的数据优先级，设定路由发送给手机的数据优先级，保证延迟敏感数据包优先传输。在游戏、VOIP 场景可以提供更好的信道和网络竞争能力。
- 根据应用服务进行分类，提高游戏 / 语音等关键业务优先级，并与路由器实时同步优先级协同工作。
- 根据手机应用前后台进行分类，提高前台应用优先级，并限制后台带宽，并与路由器实时同步优先级协同工作。

4. **域名加速引擎 (Smart DNS)**
- 多线程并发查询：加快查询速度
- 基于网络 ID 的 DNS 缓存，充分利用历史查询结果
- 连通性检测，剔除无法连接的 IP 地址，选择最快的结果
- 提升应用程序网络连接速度 20%
- 避免双栈网络（IPv4 和 IPv6）返回的解析结果无法连接的问题

5. **内核协议栈增强 (Data-Path Passthrough)**
- 内核 TCP 零拷贝技术，UDP 零拷贝技术
- 重点应用场景优化，下载管理、应用商店、小米互传、一键换机、SpeedTest、iPerf

4.4.3.2 Network Kit

技术介绍

Network X 为短距通信领域网络加速的统称，主要包括协同加速引擎、智能选网、域名加速引擎等功能，为用户提供快速便捷的网络体验。

技术架构

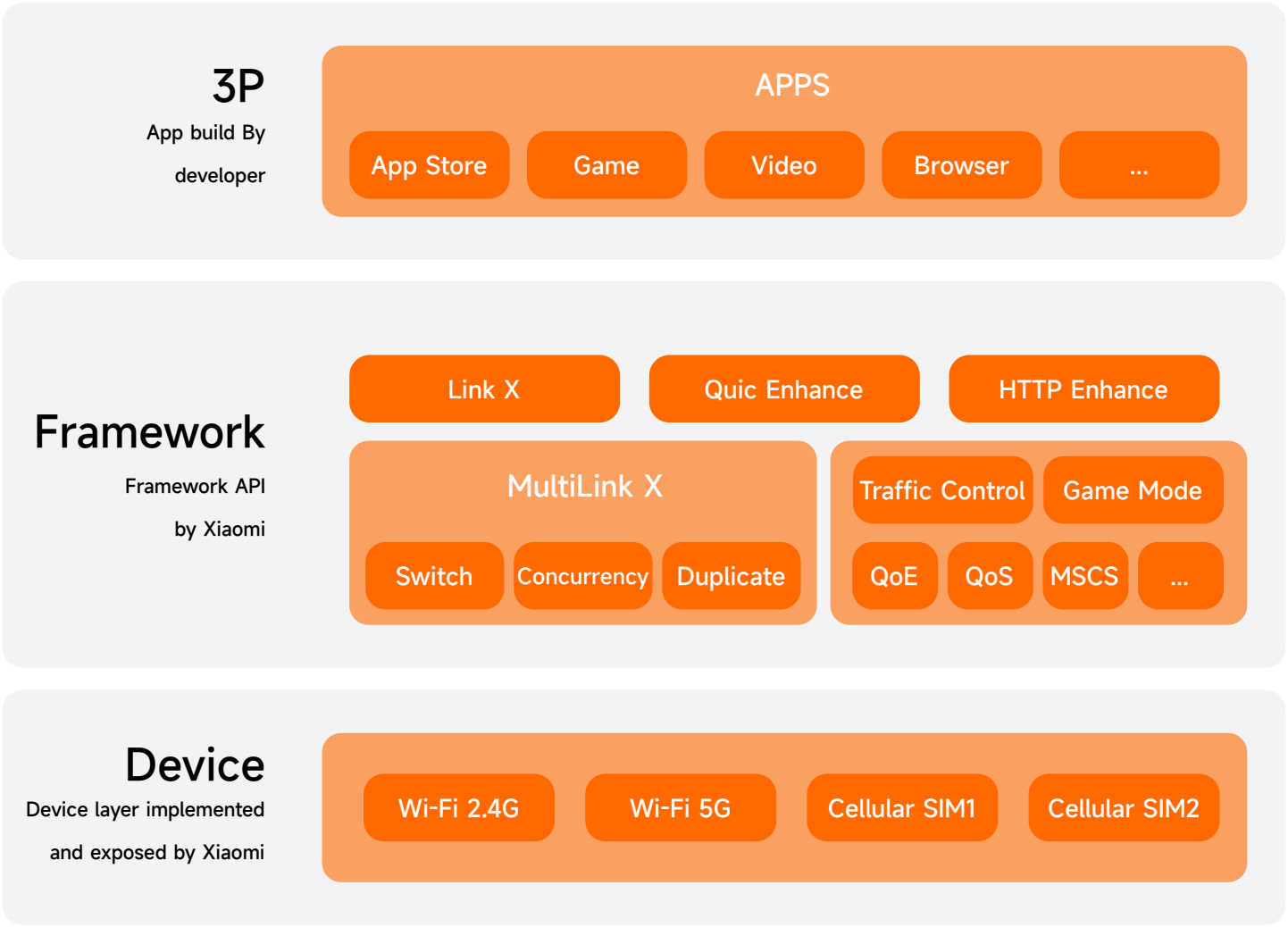


图 4.4-3 Network Kit 技术架构

关键技术

1. **标准的框架 API**
   - Socket 级别的网络控制（网络优先级、网速、延迟等）
   - 更精准的业务场景检测
   - 多网协同（并发、辅助、切换）能力
2. **提供标准网络协议框架加速**
   - QEE: QUIC 系统级加速框架
   - OKHTTP/Netty 加速框架
3. **提供更准确的网络质量监控数据，比如：**
   - 链路的质量参数（占空比、SNR、AMPDU 聚合程度，CCA 等）
   - 网络上下行统计数据（某个时间窗口内的上下行网络数据）

4.4.3.3 无线网络自恢复系统

技术介绍

提供一套集网络质量监测、网络诊断、网络恢复、云修复、大数据于一体的无线网络自恢复系统，实现在用户端自动分析故障并自愈的功能，提升用户体验。并根据大数据监测，全周期监控产品质量。

技术架构

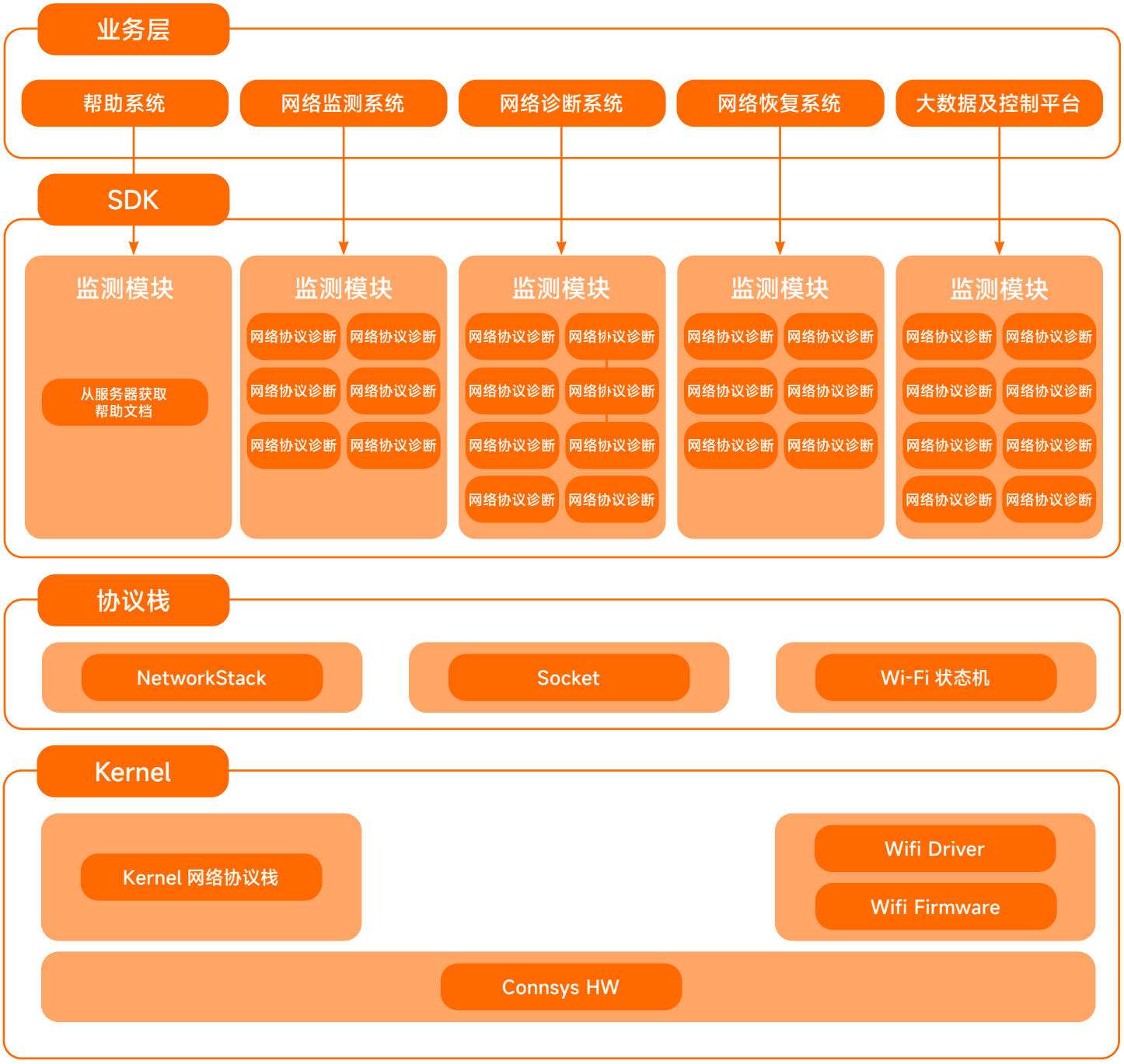


图 4.4-4 无线网络自恢复技术架构

关键技术

1. **网络监测：**监测网络质量指标，分析网络异常类型和原因；
2. **网络诊断：**实现私有的网络诊断模块，精细化诊断故障；
3. **自恢复：**根据异常类型，自动 / 手动执行定制的恢复策略；
4. **大数据：**统计终端设备组网类型，故障类型，恢复成功率，游戏延时等；
5. **云修复：**线上实时修复故障，无需升级 OTA。

4.4.3.4 超级近场通信

技术介绍

针对 Wi-Fi Direct 通信的业务场景和信道特征，优化协议参数，精简传输协议，自研网络旁路方案，自研极简协议 PHC 和 OSRTP。

技术架构

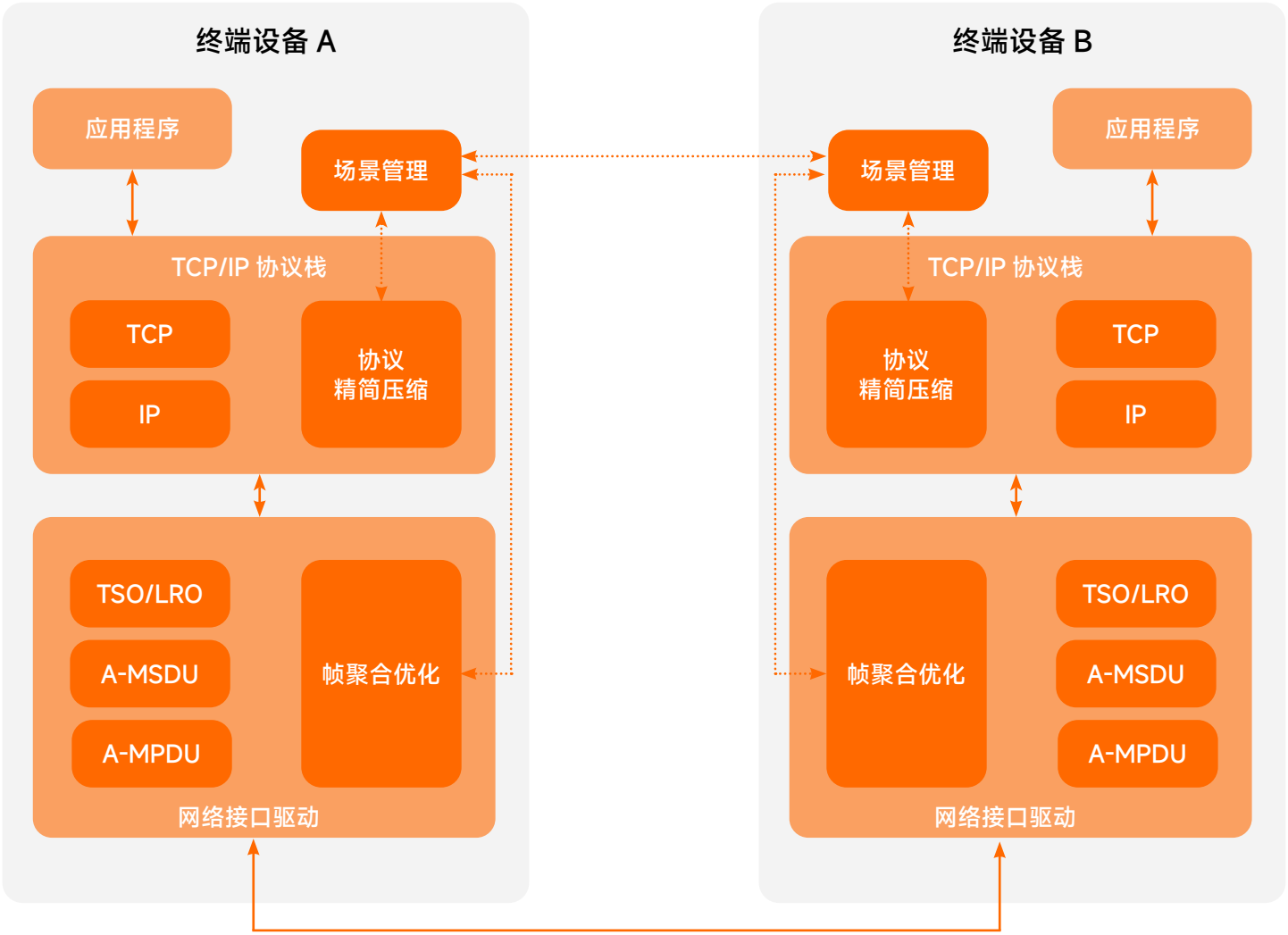


图 4.4-5 极简快传协议（PHC）技术架构

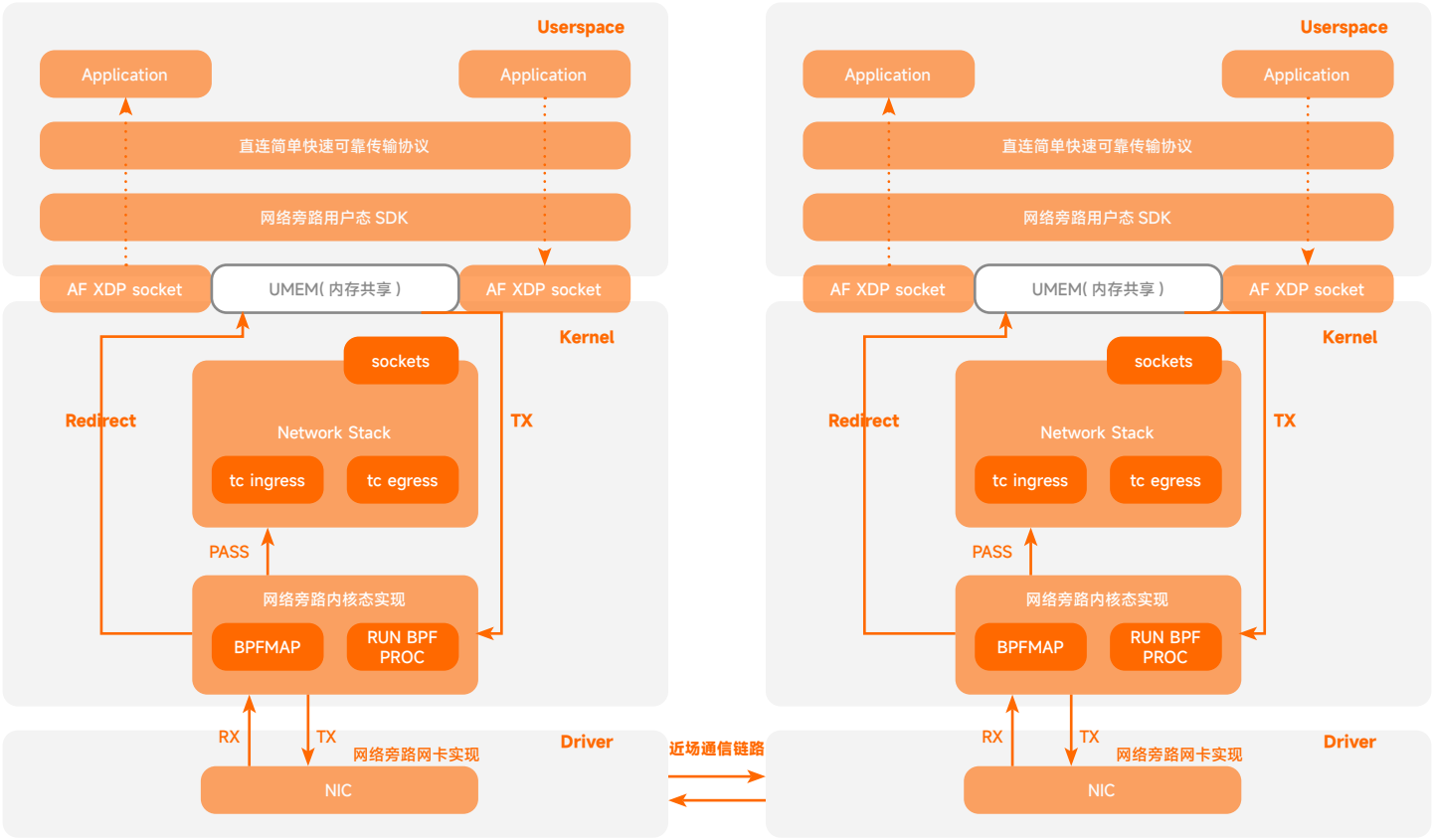


图 4.4-6 极简快传协议（OSRTP）技术架构

关键技术

1. **极简快传协议 -PHC**：PHC 协议 (IP/TCP/UDP 精简），帧聚合优化，可以使功耗降低 5%
2. **极简快传协议 -OSRTP：自研极简快传协议，** 吞吐提升 5%-15%， 功耗降低 5%
3. **多连接高频宽 (HBW)：160M 能力，MLO 能力，互传速率提升 50%**
4. **CastBoost(MTK)**：MCC 时隙动态优化，P2P 吞吐提升 8.9% ~ 121%，投屏场景 P2P 时延降低 2% ~ 86%，双向时延降低 22% ~ 79.6%
5. **直连连接速度优化 (HDL)**：W-HDL、B-HDL，**直连速度提升 30%**
6. **蓝牙加速引擎：互联互通蓝牙连接速度提升 50% 以上，快连弹窗速度提升 20% 以上**

4.4.3.5 数字车钥匙

技术介绍

小米数字钥匙基于 PKI 体系的非对称密钥架构，具有高安全、高隐私的特点。支持在线开通、离线开通两种方式，具有高度的灵活性。支持灵活的跨平台钥匙分享方案。支持蓝牙无感解闭锁，UWB 无感解闭锁，为用户带来丰富多样的选择。

技术架构

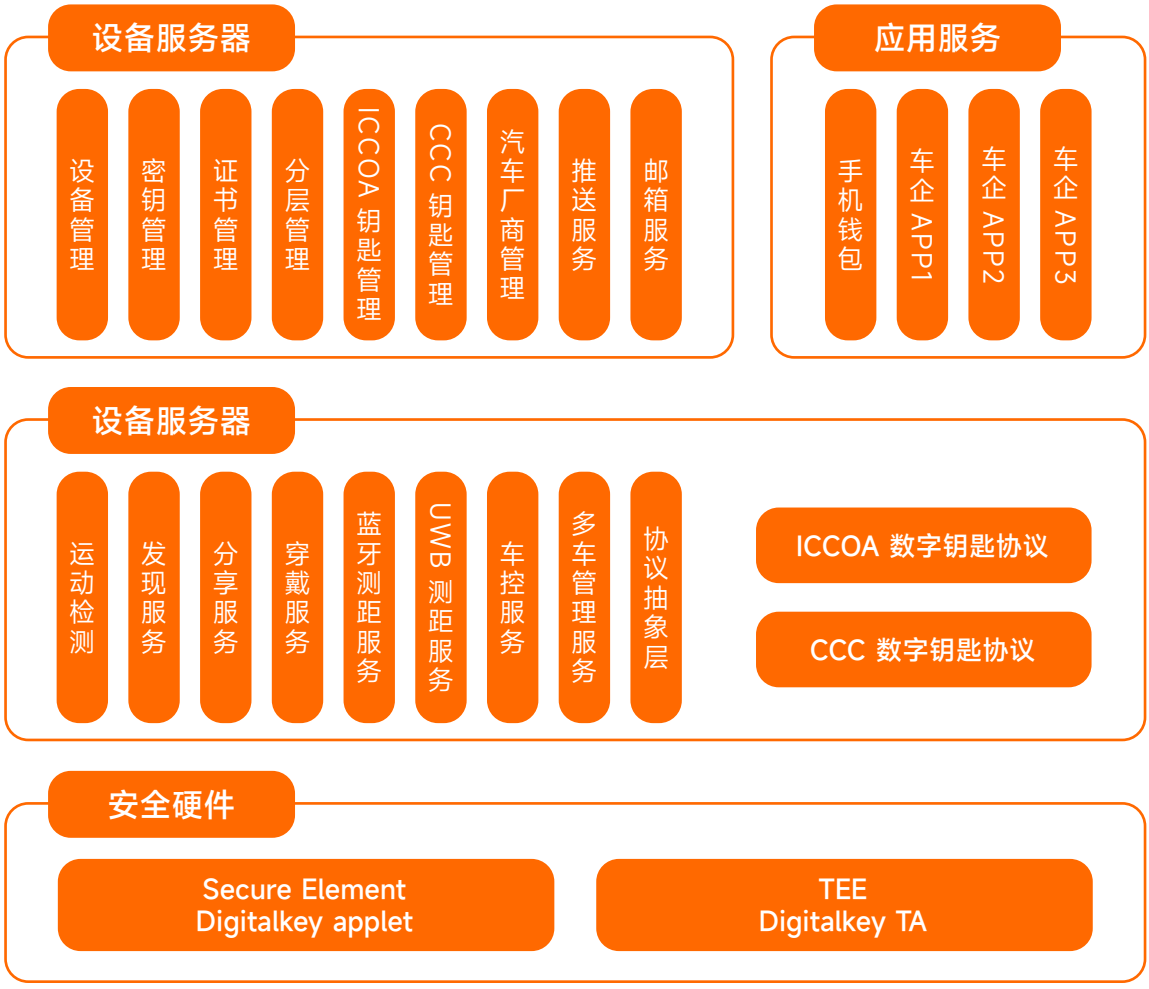


图 4.4-7 数字车钥匙技术架构

关键技术

无钥匙进入，无钥匙启动，RKE，跨品牌钥匙分享，分享至穿戴。

4.4.3.6 高精定位

技术介绍

高精定位服务包含 RTK、PPP、基于机器学习的高精度定位算法，可用于高精度导航、运动健康以及各种位置服务的应用。

技术架构

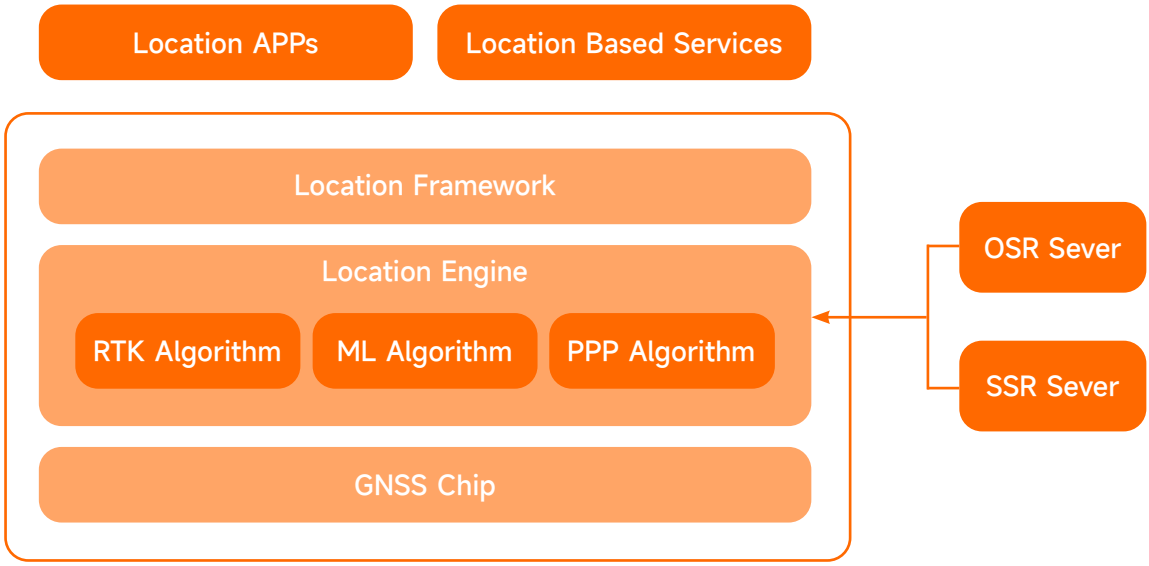


图 4.4-8 高精定位技术架构

关键技术

1. PPP 技术基于高精度的卫星轨道和钟差数据，采用载波相位观测值，综合考量并采用误差模型对误差项进行精确的改正，进行精密定位解算。支持卫星播发和网络获取 SSR 数据。
2. 基于深度机器学习的信号筛选算法，显著提高弱场环境下的定位精度。

4.4.3.7 无线音频服务

技术介绍

无线音频服务主要包含蓝牙无线高清音频、音频低延迟、高清语音通话、音频分享、助听器等行业最全面与领先的蓝牙编解码技术和协议支持，搭配无线音频接收终端和小米无线音频配件配置框架，全方位覆盖用户无线音频体验场景，从基础的蓝牙媒体音频和通话场景，进阶到高清无损音频、音频分享和助听无障碍领域。

技术架构

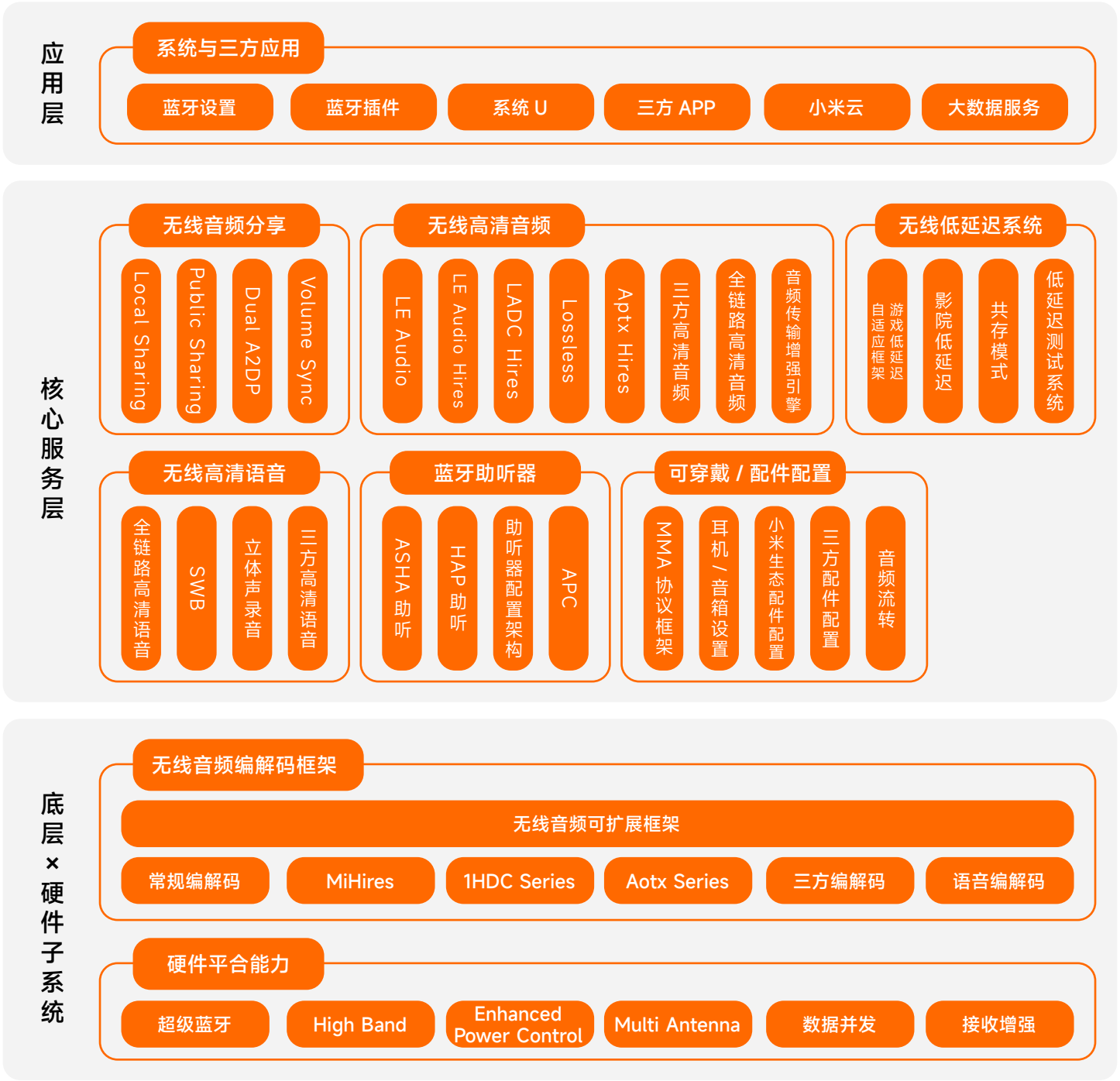


图 4.4-9 无线音频服务整体技术架构

关键技术

1. **无线高清音频：**小米领先支持 LHDC 高清音频 96K/192K 采样高清音频；领先支持新一代 LE AUDIO 蓝牙音频；并在积极布局三方无损和自研无损编解码方案。
2. **无线音频分享框架：**支持传统蓝牙耳机 SBC 的音频分享方案；支持新一代 LE AUDIO Auracast 音频分享方案。
3. **蓝牙低延迟：**蓝牙低延迟持续追求极致领先行业，已发布的几代 K76/L76/M75A 持续保持领先。
4. **蓝牙高清语音**：率先支持 aptx-tws 32K 高清蓝牙语音方案；率先支持 LE Audio 32K 高清语音方案。
5. **蓝牙助听器：**率先发布支持 Google ASHA 助听器方案；支持 LE AUDIO HAP 助听器方案，关注听力障碍群体。
6. **可穿戴配件 / 配置框架：**支持小米自研 MMA 协议，配件配置、音频自动流转算法，打造极致无线音频互联互通体验。

4.4.3.8 蜂窝信号地图

技术介绍

利用海量的通讯大数据，基于小区维度，聚合终端现网的电话、数据、网络 、短彩信异常事件的权重、频次、时间段，在云端通过 AI 决策树算法，实时生成全国各个基站小区质量评分。当前手机匹配异常基站小区，快速切换并驻留最优小区。

技术架构

整体技术方案：

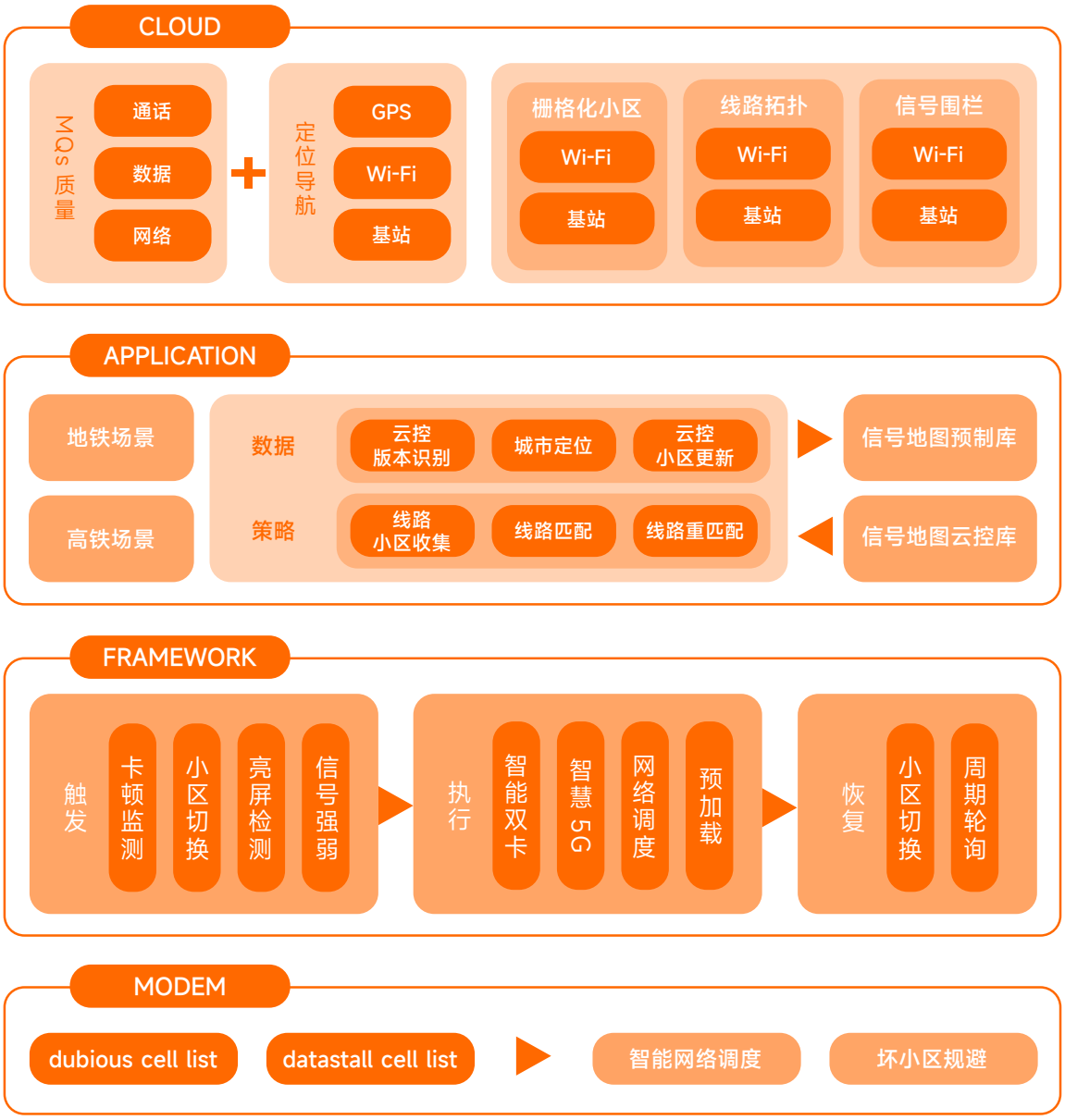


图 4.4-10 蜂窝信号地图技术架构

关键技术

1. **基站定位技术：**通过 GPS 与 Wi-Fi 定位技术采集用户的地理位置信息与扫描到的基站信息，对单个基站的多个 GPS 位置信息采用 DBSCAN 中心聚类方法提取到基站的定位信息，供上层使用。
2. **基站评分体系：**对于 UE 侧发生的通话、数据、网络的异常行为进行采集、加工、清洗、过滤，提取部分区域的最大值，采用自有评分体系对单个基站的网络性能进行评估。
3. **栅格化小区：**融合基站定位与基站评分多维数据源，采用二维空间索引技术将单个 GPS 点位信息转换为二维空间中的栅格小区，对栅格小区进行评分，形成栅格化网络基础能力。
4. **线路小区采集：**使用地图开放平台轨迹提取服务提取线路起终点信息，通过【位置与时间】标签，筛选出在特定时间内（如一个月之内）在特定高铁线路起始站坐标出现

过的用户的轨迹信息，并从该部分用户（量级为一万 / 日）的轨迹信息中提取设备当时连接的基站信息，最终仅将特定高铁线路沿线的基站信息传输到端侧。

5. **线路匹配算法：**端侧将每个线路的所有小区信息提取到 bitmap 中，进入高铁模式后，顺序采集用户驻留小区，匹配 bitmap，匹配到具体线路后进行线路小区加载。

4.4.3.9 弱信号预测

技术介绍

基于实时小区参数和用户场景，通过小区学习实现对常用线路的蜂窝信号质量 / 数据质量的学习，并生成线路小区数据集；通过小区预测实现对未来小区的信号质量 / 数据质量的综合预测。在弱信号预测中，通过打通 AP 和 Modem 之间的链路，实现了更加多元更加精细的关键参数搜集。同时，将 Modem 底层参数与 AP 场景结合，实现基于用户场景的小区质量评判标准；预测过程将结合用户实时场景和学习结果，实现对下一小区的信号预测。

技术架构

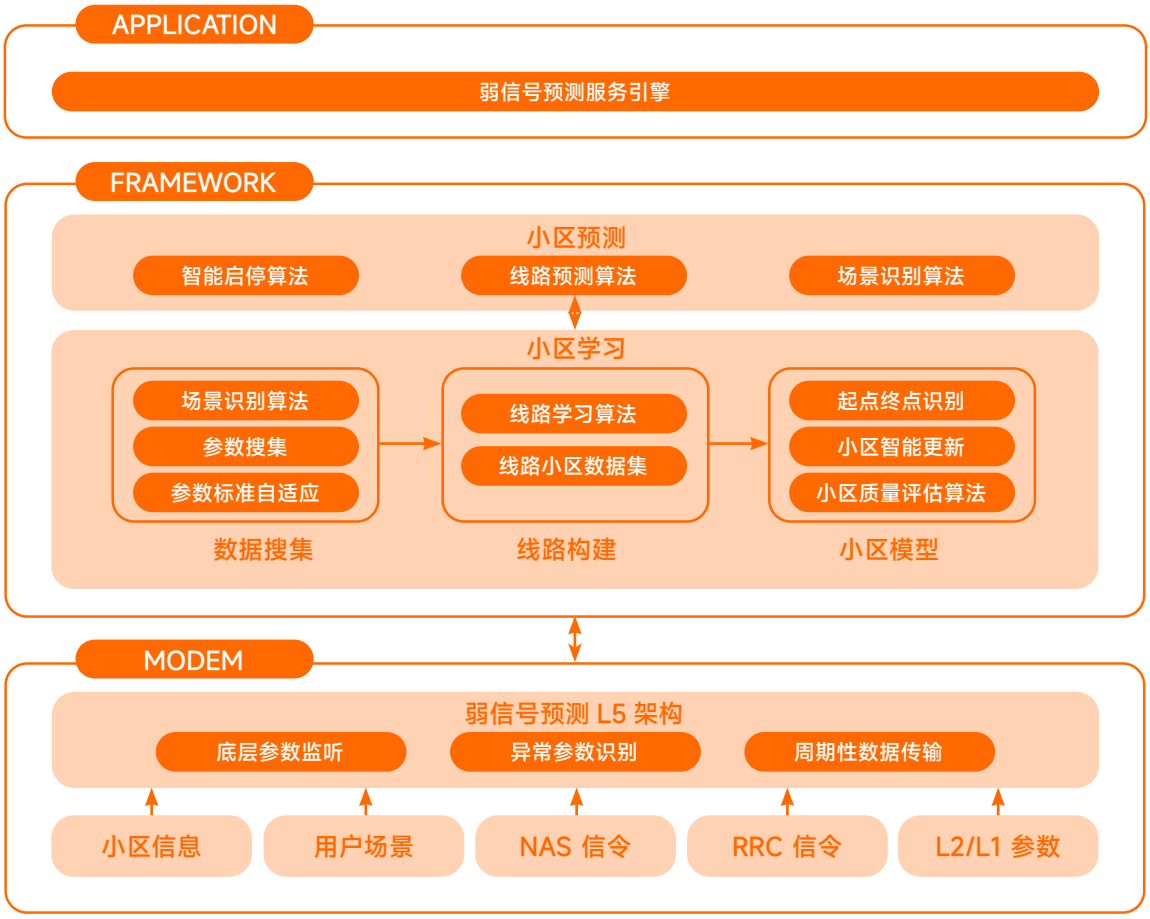


图 4.4-11 弱信号预测技术架构

关键技术

1. **预测下一小区的信号质量 / 数据质量。**

- 基于当前服务小区的信息，实时匹配线路，并对线路下一个小区的预测。
- 支持应用注册相关服务。

4.4.3.10 智能网络调度

技术介绍

智能网络调度是指智能调度双卡上网能力（双卡并发 / 切换），解决主卡无服务 / 无法上网 / 上网过程中卡顿 / 下载慢 / 延时高等问题，通过动态建立双数据 / 动态选择双卡里面通讯质量最好的网络，来减少无网或网络环境差导致的数据卡顿，缩短下载时间，降低时延。

通过监控网络能力 / 信号质量 / 设备状态 / 链路状态 / 业务场景 / 双卡资费套餐 / 开关状态 / 底层 Modem 推荐算法，智能切换双卡并发和切换的状态。

技术架构

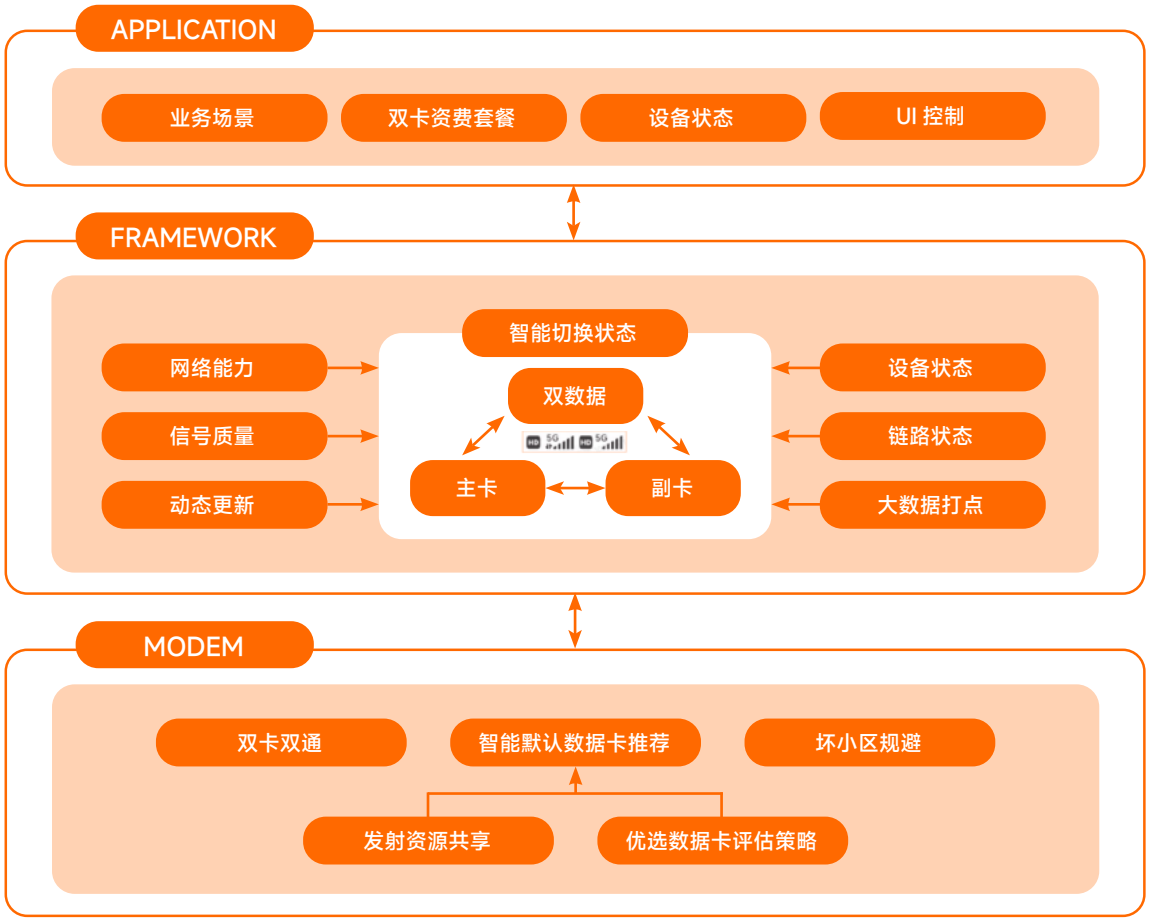


图 4.4-12 智能网络调度技术架构

关键技术

1. **双卡无缝切换**

- **正常情况：**
  当一张卡信号不好，自动切换到另一张卡（中间过程 :Telephony 断开一张卡的数据链路，然后发请求给 Modem 建立另一张卡的网络连接，Modem RRC 建立 /PDP 激活，返回给 Telephony，Telephony 根据 Modem 携带参数重新建立路由）然后通知三方应用链路切换。
- **优化情况：**
  游戏时同时建立双链路，即形成 SIM1/SIM2 双链路共存，一路信号质量不好时自动切换到另一路（无中间过程），然后通知三方应用链路切换。

2. **智能调整并发和切换**

- 监控双卡信号质量，智能决策并发 / 切换。
- 动态场景推荐使用 SIM1+SIM2 的数据并发，双链路备份，提供可靠上网服务。
- 静态场景推荐使用 SIM1/SIM2 智能协同，优选最优链路，提供最佳上网服务。

4.4.3.11 蜂窝通信共享

技术介绍

利用互联互通技术，将有通讯能力设备上的**电话、短信、蜂窝数据能力无缝接力**到无通讯能力的设备上。

技术架构

- **通讯互联子系统**
  - 优点：代码精简、跨层传输少、高扩展、低耦合、无平台限制。
  - 效果：支持多设备多平台共享，可以实现信令消息同步几乎 **" 零时延 "**。

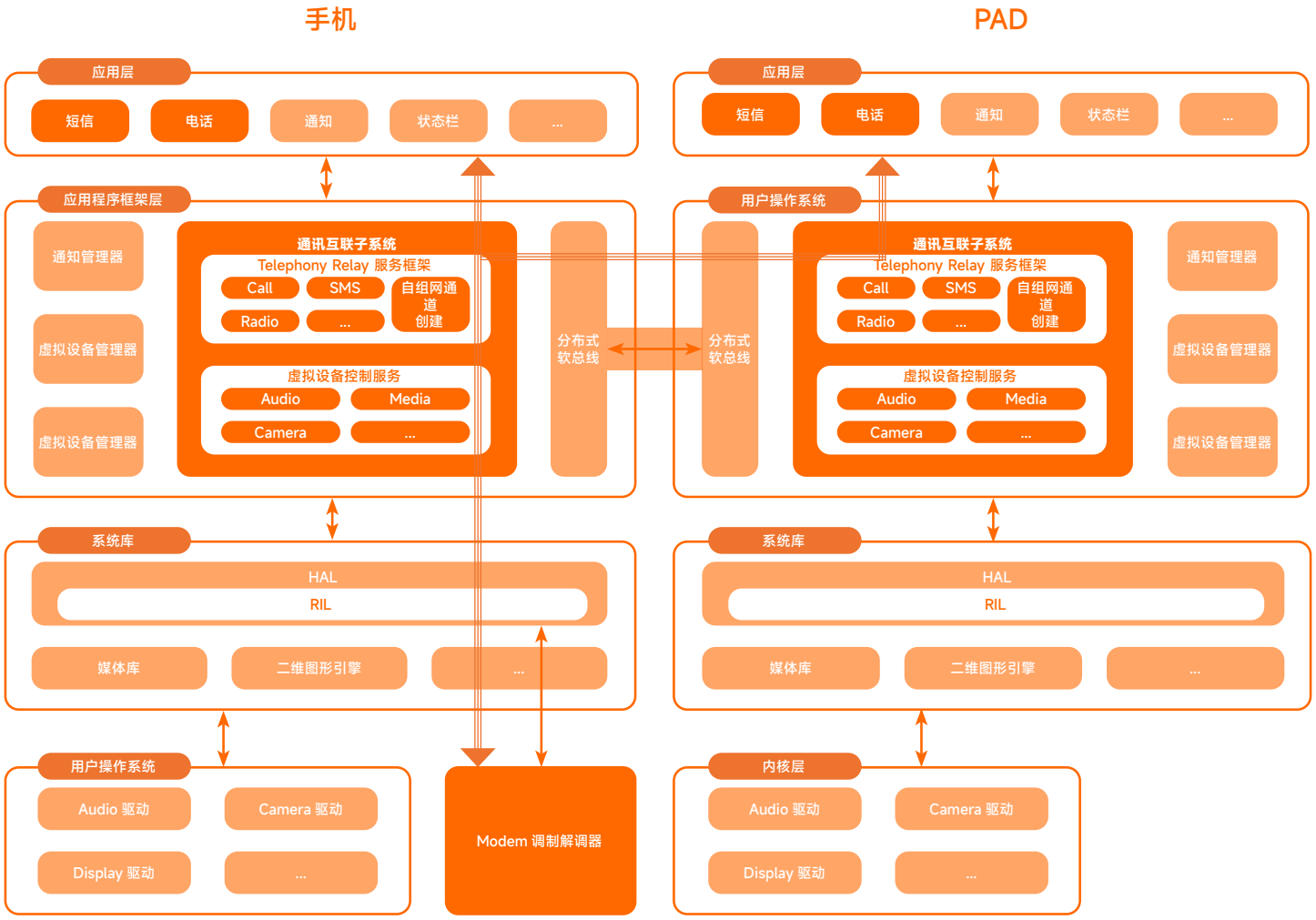


图 4.4-13 蜂窝通信共享技术架构

关键技术

1. **语音通话低延时接力**
   - 精简封装语音电话实时消息，并发传输
   - 建立电话自组网传输通道
   - 建立音频虚拟设备控制通道
   - 实现 InCallUI 页面切换显示
2. **短信零延时接力**
   - 精简封装短信实时消息，并发传输
   - 建立短信自组网传输通道
   - 实现通知栏 Notification 消息显示
3. **蜂窝数据低功耗流转**
   - 精简封装蜂窝数据实时消息，并发传输
   - 建立蜂窝数据自组网传输通道
   - 重构蜂窝数据自组网传输通道
   - P2P 实现底层链路

4.4.3.12 智慧出行

技术介绍

智慧出行（蜂窝场景化定向提优）依托用户出行的特定场景（e.g 高铁、地铁等），通过快速感知用户痛点信号网络问题，结合用户设备状态，使用决策树算法在方案矩阵里寻找最优解，静默解决用户信号网络问题，持续改善用户出行的通讯体验。

技术架构

- 5G 智能出行：地铁 / 高铁 / 郊野 / 机场 / 高速公路
- 高频弱场场景优化：电梯 / 地库 / 居民楼

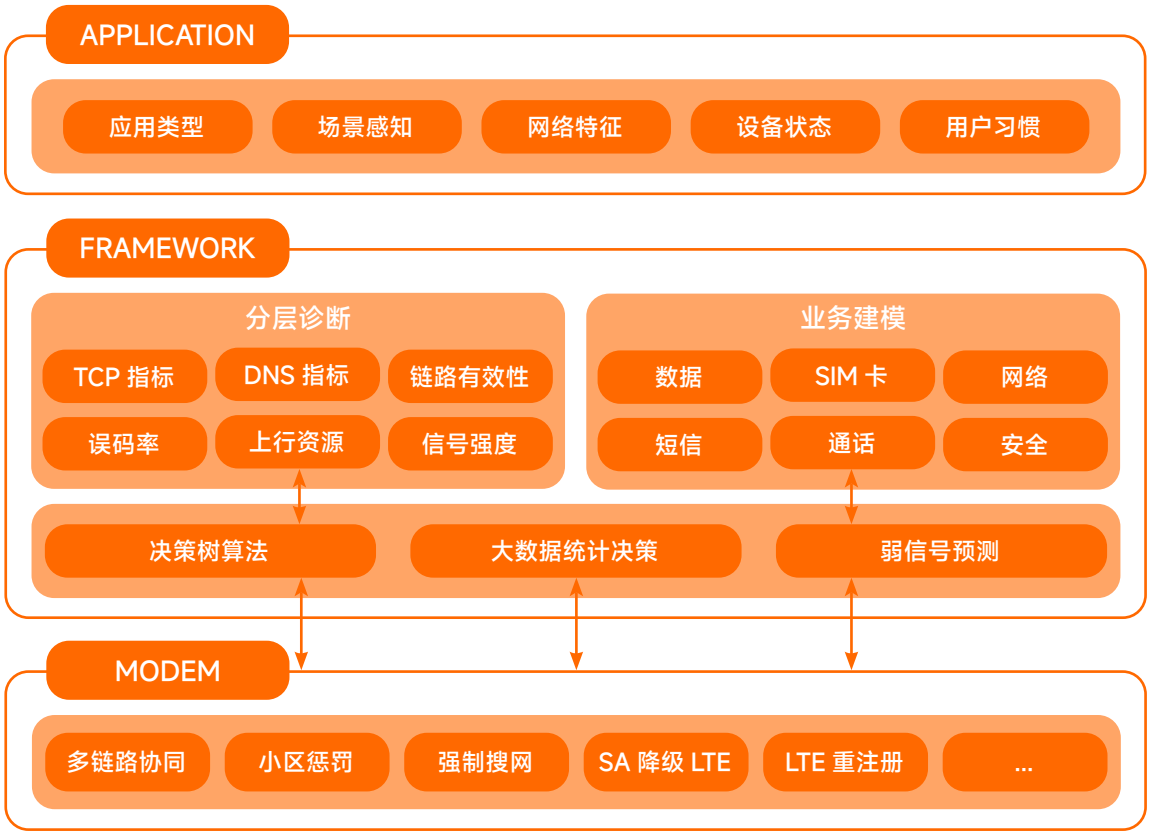


图 4.4-14 智慧出行技术架构

关键技术

1. **场景感知：**结合网络转弯特征和 sensor 来动态识别不同场景（高铁、地铁、电梯、地库、机场等）
2. **小区优化：**针对频繁乒乓的小区进行小区评估和打分，对驻留时间短信号质量差的坏小区进行规避优化；同时针对 datastall 引起的坏小区可实现快速惩罚和切换
3. **快速漫游 / 快速回网：**针对海外场景可实现频点预置，可实现快速驻网；针对 2/3G/ 脱网场景可实现快速回网
4. **弱信号预测预加载：**固定通勤线路上，可提前感知弱信号即将到来，提前做三方预加载或者降分辨率等相关优化措施，来提升用户体验

4.4.3.13 信号智选引擎

技术介绍

通过核心场景识别，核心参数输入，核心算法决策当前底层射频天线行为策略，提升基础信号能力。

技术架构

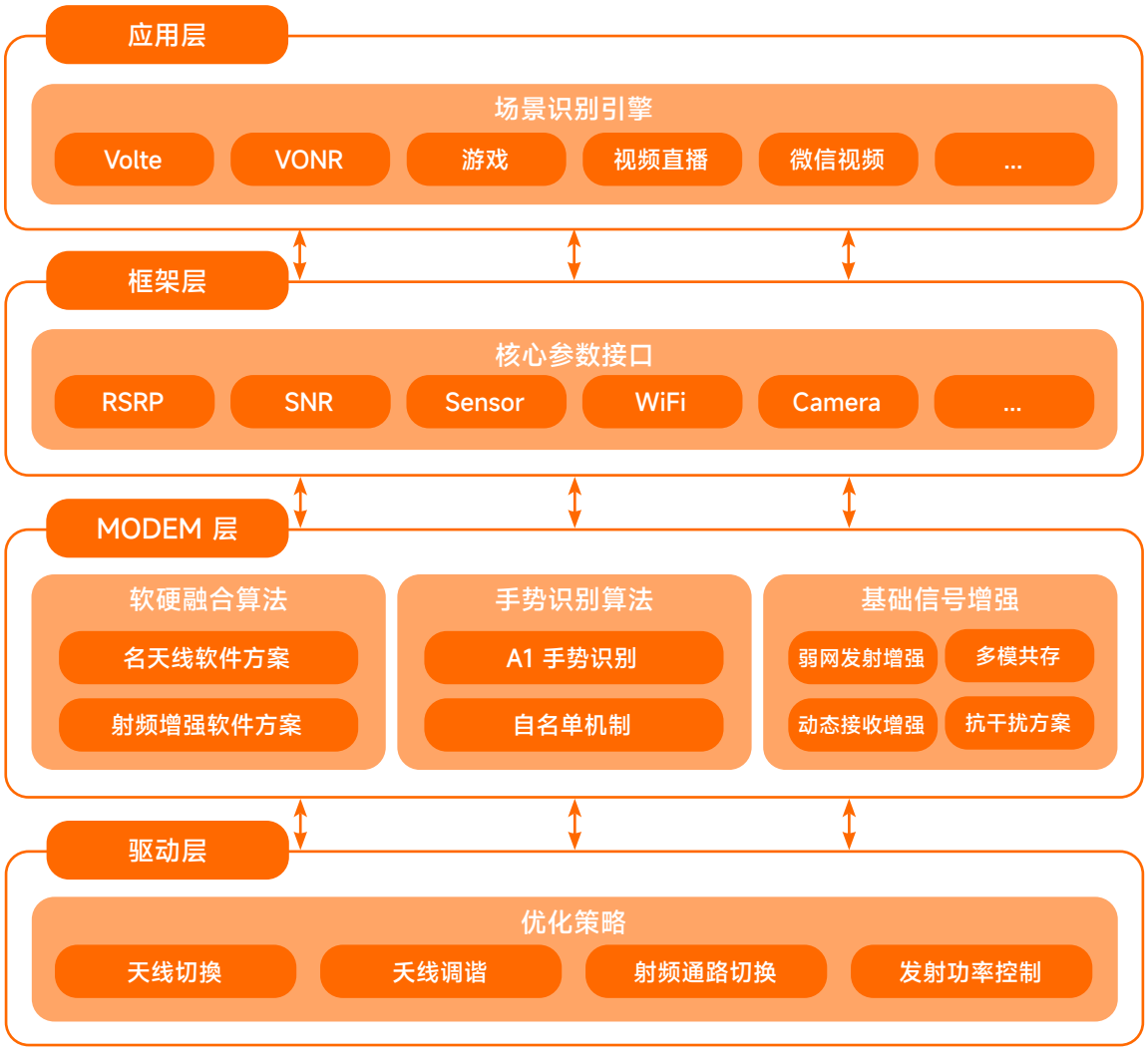


图 4.4-15 信号智选引擎技术架构

关键技术

1. **软硬融合算法**
   - 多天线软件方案，软硬融合，不同场景进行动态天线切换，提升上行和下行性能
   - 射频增强软件方案，软硬融合，不同场景进行动态天线调谐，提升上行和下行性能
2. **手势识别辅助增强**
   - 基于 AI 模型，识别用户各种握姿
   - 通过白名单机制，控制核心场景

- 天线切换和调谐，优化核心场景信号

3. 基础信号增强

- 弱网发射增强，识别弱网，动态调整发射功率
- 动态接收增强，智能判别接收通路质量，动态切换接收通路
- 多模共存，识别多模共存场景，动态进行天线调谐
- 抗干扰方案，识别多种干扰场景，动态进行天线切换和天线调谐

4.4.3.14 双卡双通双数据

技术介绍

- 随着移动通信技术的不断发展以及 5G 技术的普及，越来越多的用户选择 5G 的上网套餐。由于 5G 套餐资费贵、流量耗费快的特点，很多用户会选择大王卡，物联网卡这类流量卡，但是由于这类套餐往往都限制应用，用户在使用起来需要频繁切卡，为使用带来了很多的不便。
- 双卡双通双数据模式依赖于底层 DSDA 能力，建立两条数据链路，对 tcp 连接建立两条 scoket 通路，通过 iptables 技术将两条链路的数据包聚合；
- 通过两张网卡发送相同的数据包，两个数据包竞速，先到达的数据包先被接收。最终达到应对复杂网络场景挑战，双管齐下，无惧断流。

技术架构

对于双卡用户，基于手机的 DSDA 能力，使用负载均衡相关技术对双卡的数据流量进行平衡。用户在复杂网络环境下，手机可根据当前的网络环境，自动调整两张 SIM 卡上的流量；针对流量卡这种特殊卡，手机也可以通过识别对应的 APP 选择合适流量卡进行上网。项目整体架构设计如下图所示。

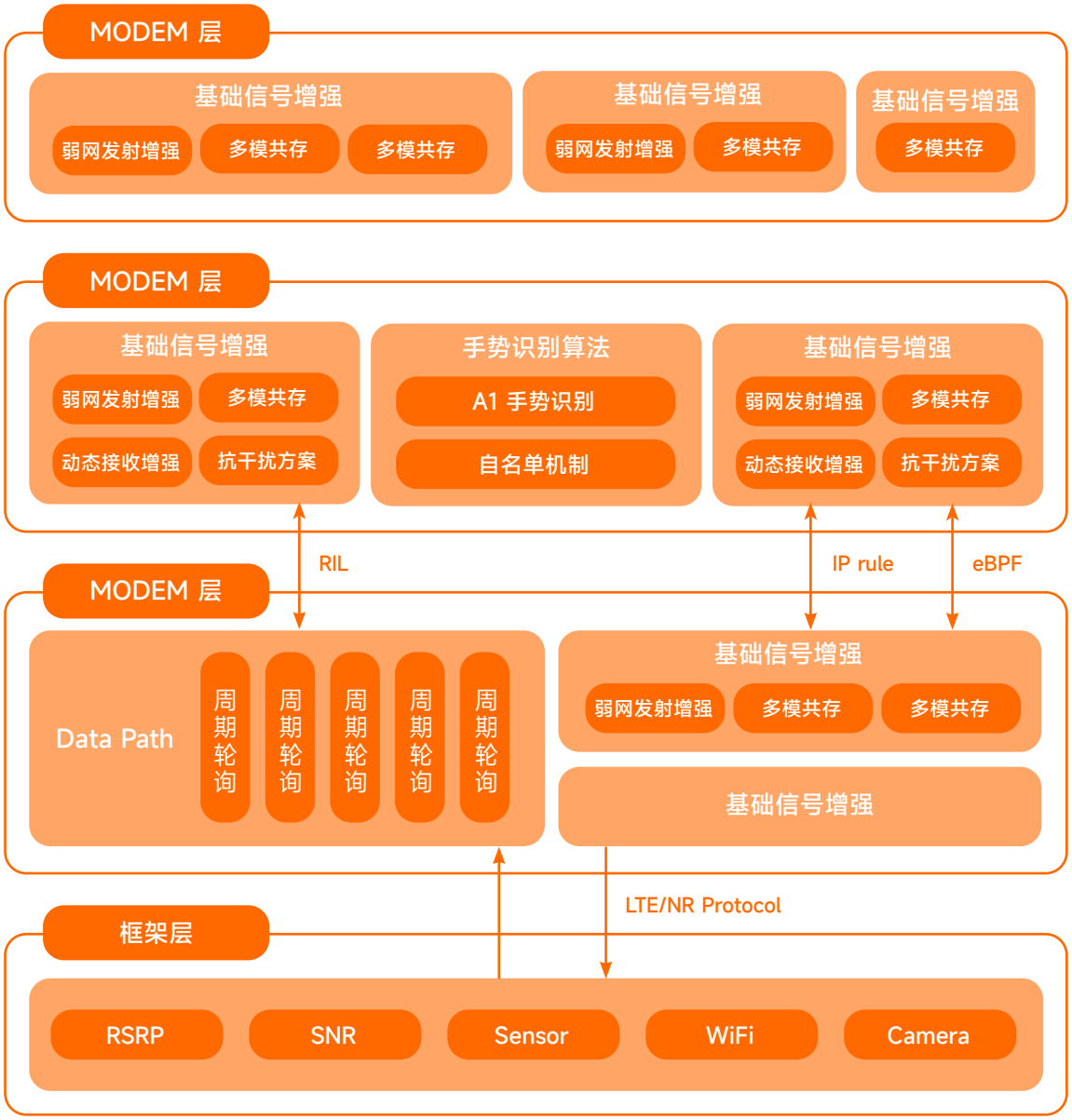


图 4.4-16 双卡双通双数据技术架构

关键技术

1. **副卡数据连接托管，增加 Android 双网卡的 bringup 以及对副卡的数据管理**

- nDDS 基本的 BringUp/TearDown 功能
- nDDS 数据图标指示
- nDDS 和双卡切换容错，按需快速重建
- 并发模式和主从模式的优先级设定

2. **双数据网络数据包融合，完成两张网卡的数据聚合，从而实现增大带宽的效果**

- 双数据网络路由管理：通过 ip rule 对两张网卡上的数据流进行策略路由，将打 Mark 的数据包路由到对应的网卡
- 双数据网络负载均衡：通过 iprables 技术，将服务端传到手机上的数据包按照一定的算法动态分配到两张网卡上，从而达到两个数据链路的负载均衡，增大带宽

4.5 多媒体子系统

4.5.1 子系统简介

多媒体子系统是澎湃 OS 的一个重要组成部分，是处理音频、视频、影像等多媒体数据的功能 / 服务集合，对外提供丰富的 API，使应用程序能够高效地处理音频、视频以及执行多媒体相关的任务。本子系统包括**音频系统、视频系统、影像系统**等诸多组件。

音频系统是澎湃 OS 中负责音频流管理、音频设备管理、音频数据处理等功能的软件系统。

视频系统是澎湃 OS 中负责视频播放、视频录制、硬件编解码、缩略图、媒体库、视频文件封装解析、视频后处理、视频转码等功能的模块。影像系统也是澎湃 OS 多媒体系统的重要组成部分，为用户提供诸如美颜、夜景、防抖等诸多影像功能。

4.5.2 子系统架构

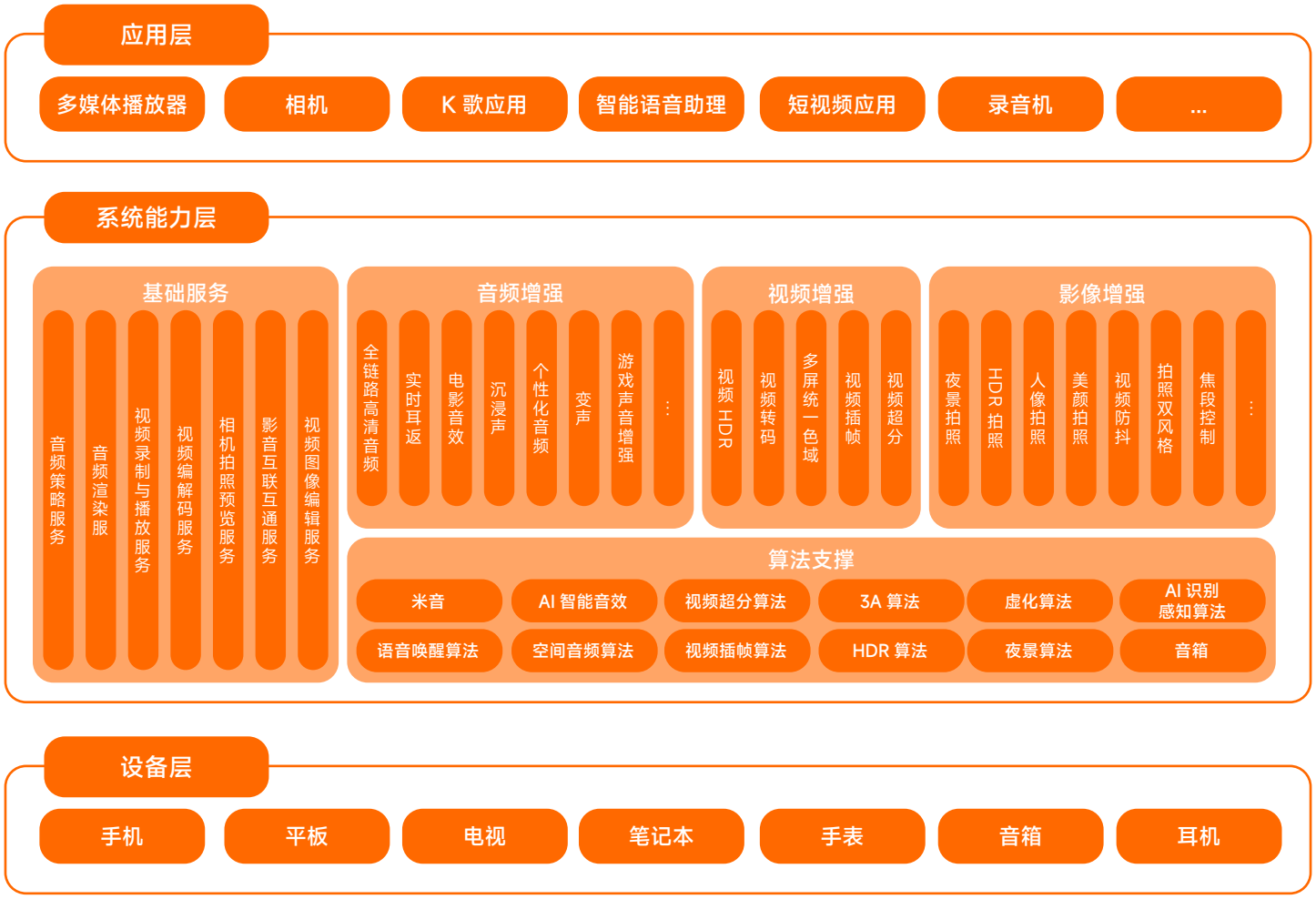


图 4.5-1 多媒体子系统架构

4.5.3 重点特性

4.5.3.1 音频系统

技术介绍

音频子系统是澎湃 OS 中负责音频流管理、音频设备管理、音频数据处理等功能的软件组合，可分为音频 API 层、服务层和硬件抽象层等核心部分。以下是音频子系统中主要组件，例如音频管理器、音频录制 / 音频播放，音频策略服务、音质音效等组件。澎湃 OS 音频在高品质音频、沉浸感、互通互联能力等方面进行了专项优化，为用户呈现极致音频体验。

技术架构

图 4.5-2 音频系统架构

关键技术

1. **高清音频播放：**在澎湃 OS 中，对于扬声器 / 有线耳机场景，实现了全链路位深 32 比特通路播放，使音频数据动态范围更大，精度更高，量化误差更小，噪声降低近 30db，有效地还原了声音细节。

高清音频播放

通过对音频系统各个层级进行优化实现了有线耳机 / 扬声器场景的全链路位深 32 比特播放通路

30db
量化噪声

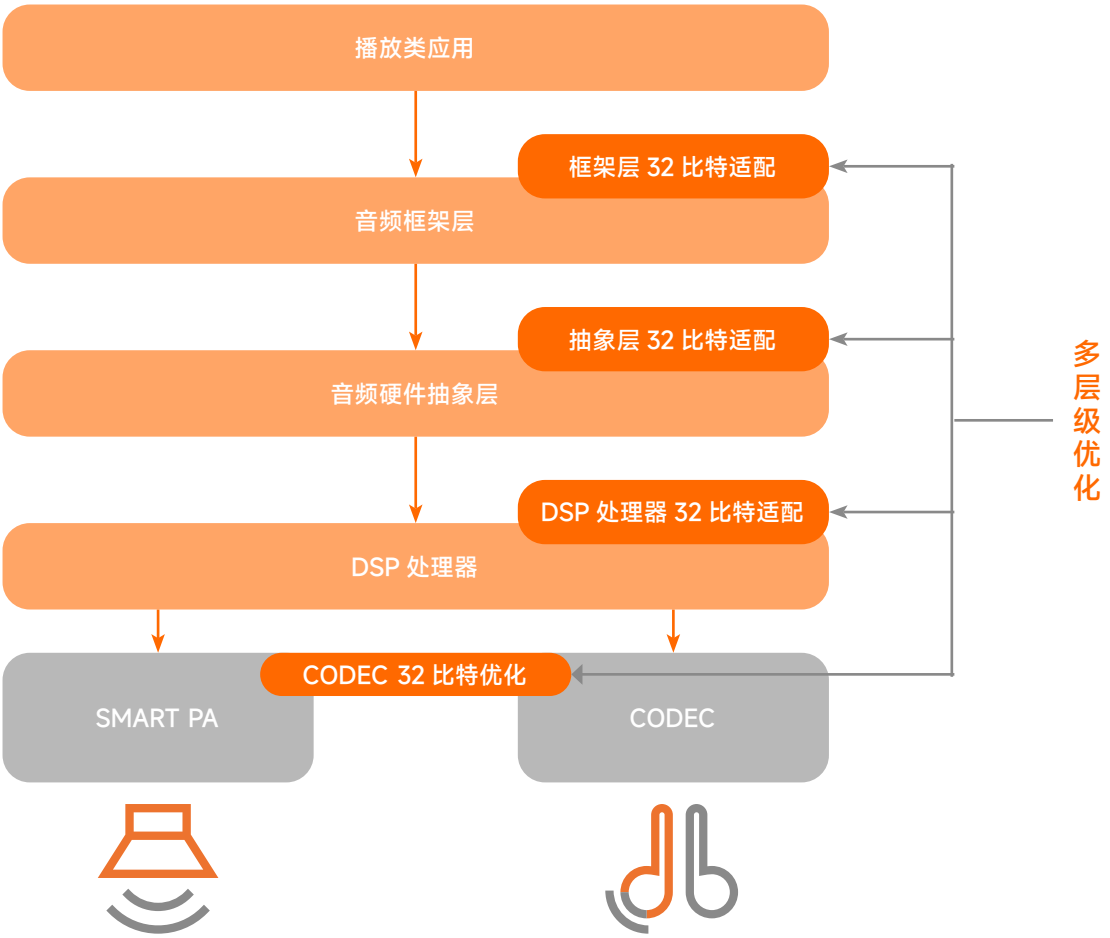


图 4.5-3 高清音频播放框架

2. **实时耳返：**在开启 K 歌应用，开启 K 歌耳返，能让您实时听到自己的声音。相比于 K 歌类应用自带耳返，澎湃 OS 实时耳返通路直接从 ADSP 环回，这样的设计有效地减少了音频延迟，给您极佳的返听体验。不仅如此，澎湃 OS 实时耳返还提供如下功能：

- 实时调整声音大小。
- 选择不同的混响效果，让您获得 KTV、剧场、音乐厅等不同的场景音效体验。

图 4.5-4 实时耳返框架

4.5.3.2 视频系统

技术介绍

视频系统是澎湃 OS 中负责视频播放、视频录制等功能的模块，具体功能如下：

- 视频播放，视频录制，硬件编解码，媒体库，缩略图等通用视频场景和功能实现。
- 视频文件封装，视频文件解析，视频转码等通用能力及定制。
- 视频后处理，视频画质增强，杜比视界录制，杜比视界播放等定制服务和能力。
- HDR10，HEIF，P3 等特定功能的通用实现和定制。

技术架构

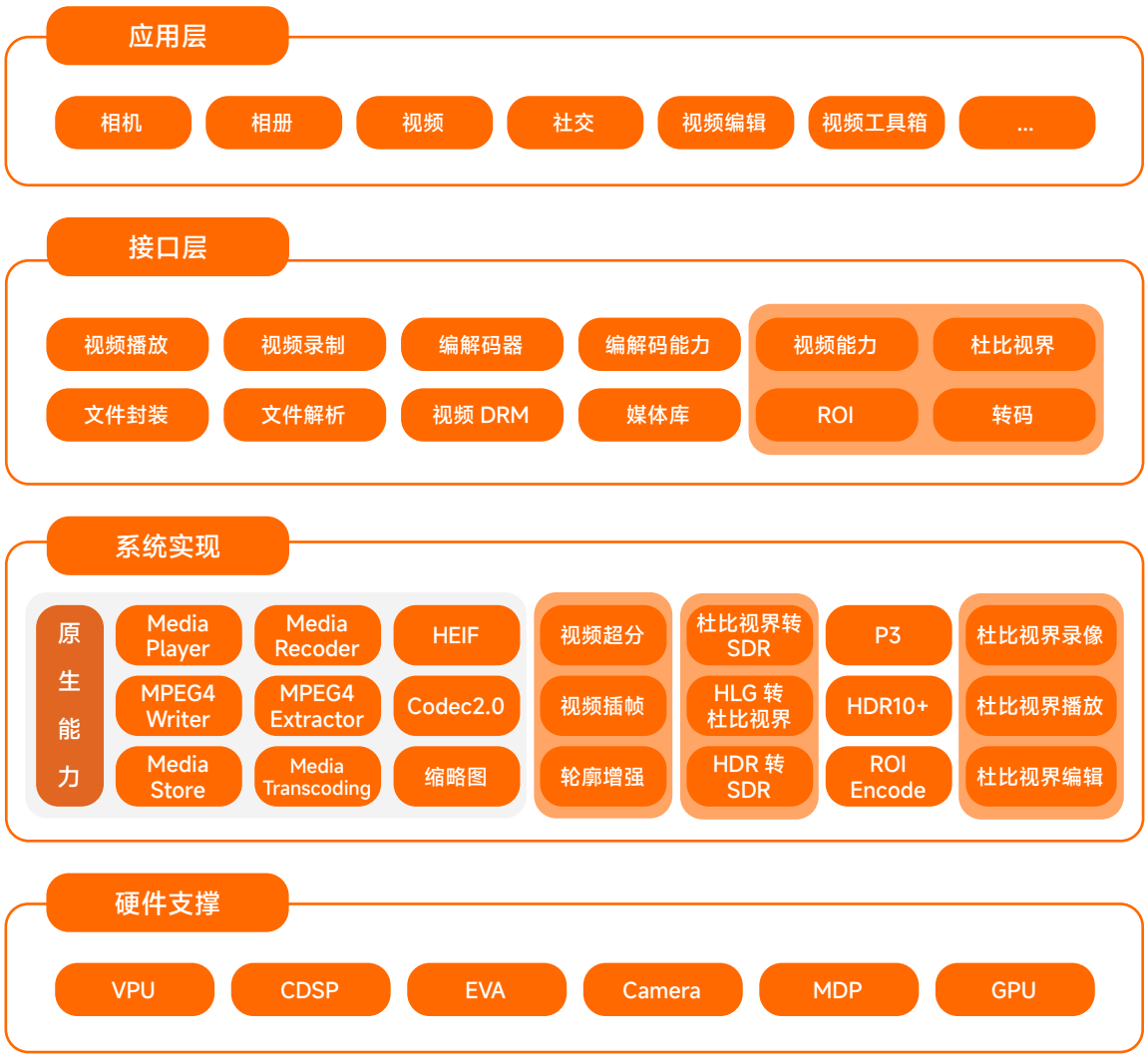


图 4.5-5 视频系统架构

关键技术

1. 多屏统一色域 -P3 编解码

- 提供全链路 P3 色域支持和 P3 编解码能力。
- 相对于 bt 709，P3 录像视频支持的色域范围更广，色彩更丰富。
- 投屏 / 短视频录制 / 社交分享场景 / 拍摄动态照片 / 相机录制等场景。
- 解决手机画面流转至电视、pad 场景下色彩失真问题，最终实现任意屏幕（电视、车机、平板、电脑）投屏，色域一致。

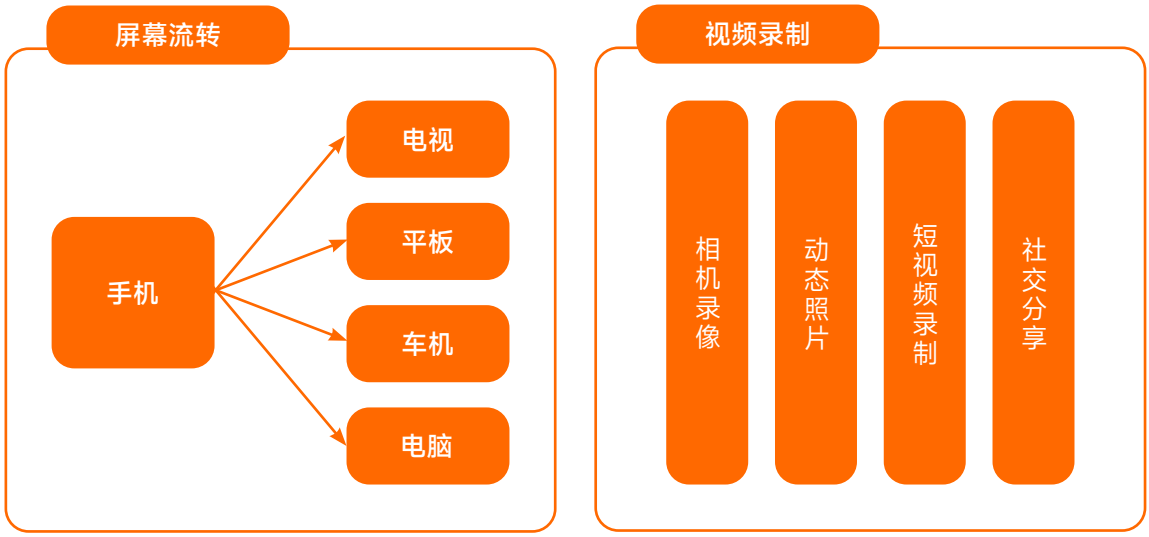


图 4.5-6 多屏同色和 P3 编解码场景示意

2. 全链路杜比视界

- 杜比视界录像，杜比视界播放，杜比视界编辑。
- 杜比视界 metadata 生成、保存和解析。
- 杜比视界编码、解码和显示画质。

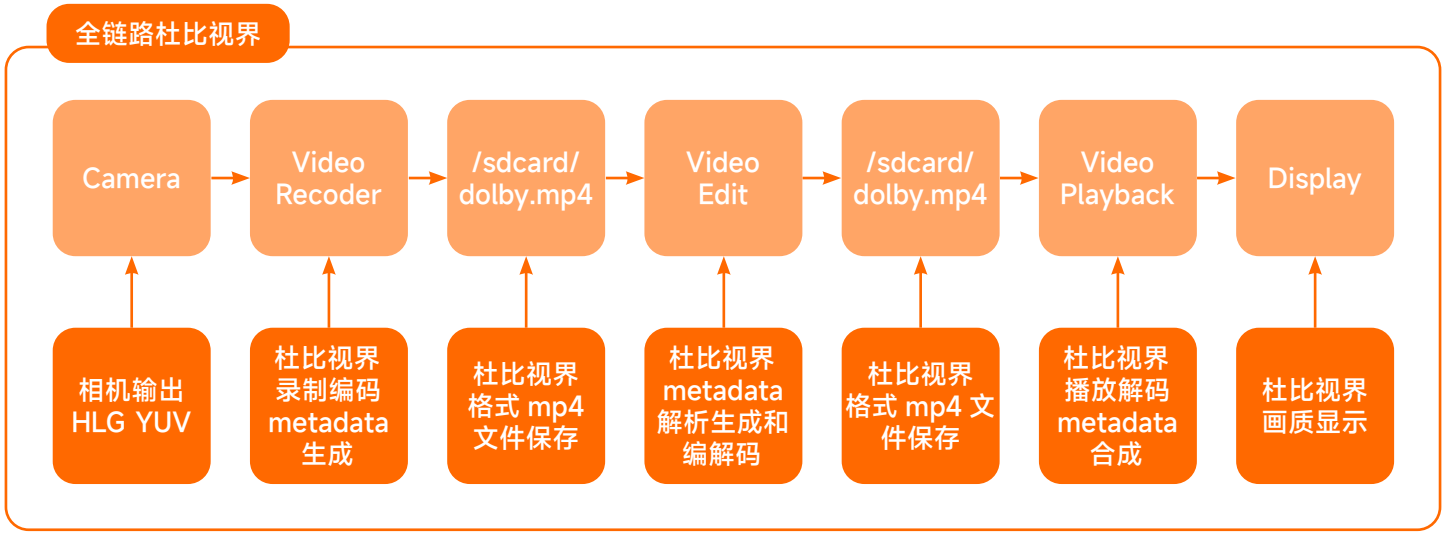


图 4.5-7 全链路杜比视界架构

3. **视频后处理**

- **视频超分**
  - 视频分辨率长和宽分别放大两倍或三倍后再输出显示
  - 更好适配屏幕分辨率，视频播放更清晰
- **视频插帧**
  - 基于硬件电路实现视频帧率提升两倍
  - 更好地匹配屏幕刷新率，视频播放更流畅
- **轮廓增强**
  - 基于硬件电路提升视频对比度、亮度和色度
  - 视频播放画质更鲜艳明亮

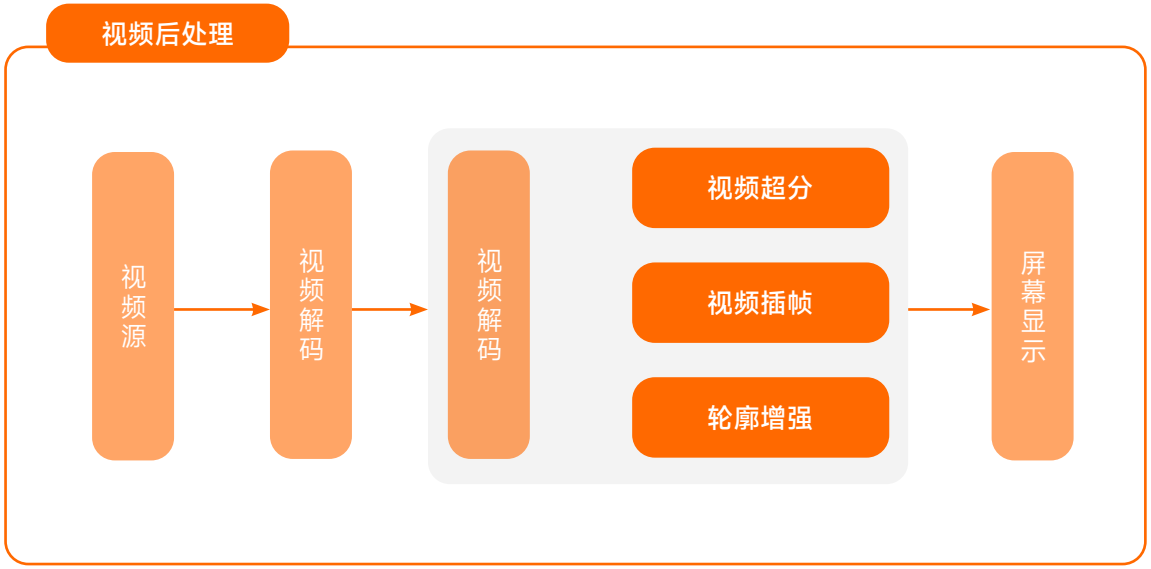


图 4.5-8 视频后处理流程

4.5.3.3 影像大脑

技术介绍

影像大脑是小米在软件方面的一大突破功能，通过重写相机架构，将软硬件能力整合重建，可以为小米手机相机带来全方面的快。

目前影像大脑升级到 2.0 阶段，包括：生态引擎、融合光学、仿生感知、色彩引擎、加速引擎以及 ISP 等。

技术架构

图 4.5-9 影像大脑架构

关键技术

打造三大核心特性：

1. **快拍相机：**自研加速引擎，启动、对焦、抓拍全流程提速，定格能力比肩专业相机。多项核心指标得到显著提升：
   启动速度：+16%
   对焦速度：+26%
   抓拍速度：+14%
2. **万物追焦 2.0：**惊鸟鸣虫，这次也能锁定；新增猫狗眼、昆虫、鸟类等智能追焦对象，出镜自动找回目标，真正万物可追。
3. **超色彩影像：**让色彩，获得真实表达。通过 AI 学习照片色彩与光影关系，搭配全链路 P3 广色域，高保真记录颜色层次与画面细节，还原你眼中的真实质感。

4.6 硬件服务子系统

4.6.1 子系统简介

硬件服务子系统是操作系统重要组成部分，作为硬件的抽象，为操作系统提供底层硬件服务支撑。多年来小米持续在该领域进行重点投入，从底层根技术到用户体验进行深层拆解，在多个核心方向进行自研攻关，成功打造视觉服务引擎、输入外设系统服务、智能感知服务、智慧充电引擎及振感框架等领先技术，整体体验处于行业领先水平。

4.6.2 子系统架构

硬件服务子系统是操作系统重要组成部分，作为硬件的抽象，为操作系统提供底层硬件服务支撑。多年来小米持续在该领域进行重点投入，从底层根技术到用户体验进行深层拆解，在多个核心方向进行自研攻关，成功打造视觉服务引擎、输入外设系统服务、智能感知服务、智慧充电引擎及振感框架等领先技术，整体体验处于行业领先水平。

图 4.6-1 硬件服务子系统架构

4.6.3 重点特性

4.6.3.1 视觉服务引擎

技术介绍

显示模块，是手机系统中非常重要的部分，它决定了手机与用户交互的效率与质感。应用于显示部分的视觉服务引擎，是基于图形渲染技术，通过对亮度，色彩、帧率等方面的精细控制，融合了机器学习技术和硬件定制，致力于打造出流畅、专业的视觉体验的解决方案。

小米的视觉服务引擎，得益于在显示架构中，从上到下的自研算法的使用，以及横向软硬件数据的拉通，具备了高度定制，灵活应用的能力。它采用了机器学习技术，深度融合各类软硬件数据，结合专业定制与校准的显示模组，能够智能的地适应各种复杂的环境和需求，自动优化图像处理过程，实现更为出色的画面效果。形成了以采用高效算法的图形渲染引擎为基础，以专业增强游戏视频场景体验为特长，能搭配专业定制显示硬件，兼顾多种视觉保护方案为特点的整体系统显示解决方案。小米视觉服务引擎的应用，可以为用户带来更为安全，优质、高效、个性化的视觉享受和操作体验，满足人们对美的追求，为手机与用户的交互带来新的无限可能。

技术架构

小米显示总体框架中，自研的算法和驱动融合在了从 APP 端到硬件驱动的整个过程中。自研算的融入，一方面专门针对手机硬件的特点进行优化，实现了更精确的色彩还原、更流畅的动画效果和更快的响应速度。同时，配合上层设计，提高了系统的稳定性和兼容性，形成特有的视觉服务引擎体系，还能提供更好的安全性和隐私保护。让用户在观看视频、玩游戏或浏览网页时享受到更清晰、更细腻的图像，更流畅的操作手感，更小的能量消耗，以及更安全的使用体验。

图 4.6-2 视觉服务引擎架构

关键技术

1. **图形渲染引擎：**手机图形渲染功能作为手机系统的重要组成部分，通过 GPU、图形 API 和驱动程序的协同工作，为手机系统的渲染场景加速提供了底层加速支撑。它不仅能够实现高质量的图像渲染和动画效果，呈现逼真的光影效果和细腻的纹理细节，还能提升用户界面的交互体验、多媒体内容的展示以及虚拟现实等领域的表现，为用户带来更加流畅、生动和真实的手机使用体验。

2. **游戏视频增强：**游戏和视频是手机重要的应用场景，在游戏视频引擎中，融合了超动态显示，memc 功能，超帧，画质增强，环境适应等技术，让用户在不同的环境下，不同的内容中，都能有更真实更丰富更流畅的视觉体验。除了上述技术的融合外，游戏视频引擎还加强了整合各组件的资源调度，避免卡顿和掉帧现象的发生，实现全局的性能加速，带来更流畅的操作体验。

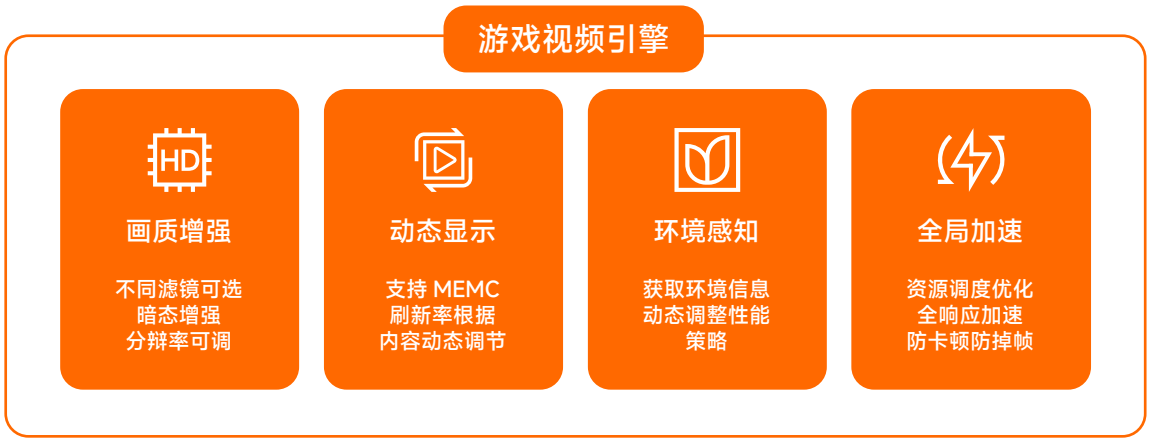


图 4.6-3 游戏视频引擎

3. **专业显示：**在系统的默认显示模式中，开启色彩管理功能，在桌面，相册等原生应用 app 端全面支持广色域显示，系统级实现色彩的准确还原。硬件端采用自研颜色校准算法，参考基于人眼仿生标准观测者模型 CIE2015 标准调校屏幕，不但让每个手机屏幕都获得了极致精确的色准，更实现了让不同设备，不同光谱的屏幕，显示出相同颜色，成功解决了色彩准确性和跨屏色彩一致性难题。

图 4.6-4 专业显示

4. **视觉保护：**视觉保护也是视觉服务引擎的一个重要方向。目前已经包含有经典护眼、节律护眼、纸质护眼、防闪烁调光、真彩显示和深色模式等多种功能的应用，可以为不同使用习惯的用户提供多种选择，在不同方面降低可能的物理损害，缓解用眼疲劳，提供更加舒适的视觉享受。

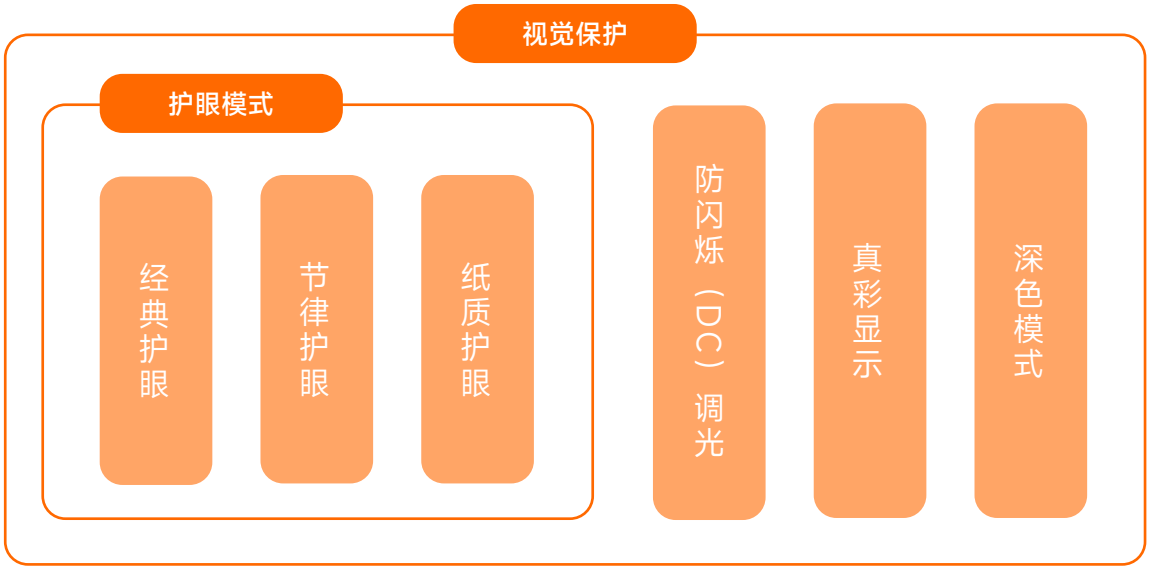


图 4.6-5 视觉保护

4.6.3.2 输入外设系统服务

技术介绍

输入外设系统服务基于 输入外设设备（触控，压感，手写键盘等），融合机器学习技术，打造准确流畅的输入操控体验。

技术架构

图 4.6-6 输入外设系统架构

关键技术

1. **自研触控框架（Touch Host Process）：**自研触控框架将底层硬件和算法实现分层，将触控原始信号的处理上移至平台端，实现更强大的报点速率，更平滑的轨迹预测，功耗与性能也获得更加优异的平衡点。同时在框架端实现各种触控算法的接入和扩展，其中触控算法包含对 Touch IC 电容信号滤波处理算法， Peak 合并算法，多点追踪匹配算法，平滑滤波算法以及坐标计算等，同时也专注于用户体验功能开发，如超高灵敏度触摸，智能防误触， 湿手触控等。

图 4.6-7 自研触控框架

2. **手写笔引擎（Pencil Engine）：**手写笔相关应用算法技术， 包括手写笔笔迹预测算法、文字识别算法和一笔成型算法等， 并负责 Pen Engine 开发，将算法提供接口给三方应用，来提高三方应用性能。

图 4.6-8 手写笔引擎

3. **算法融合：**触控模块充分利用其他器件提升触控体验， 算法融合触控，显示，指纹，传感器数据，实现握姿检测、姿态检测、口袋模式等，实现触控防误触，灵敏度增强，进一步提升指纹解锁体验和触控跟手性能。

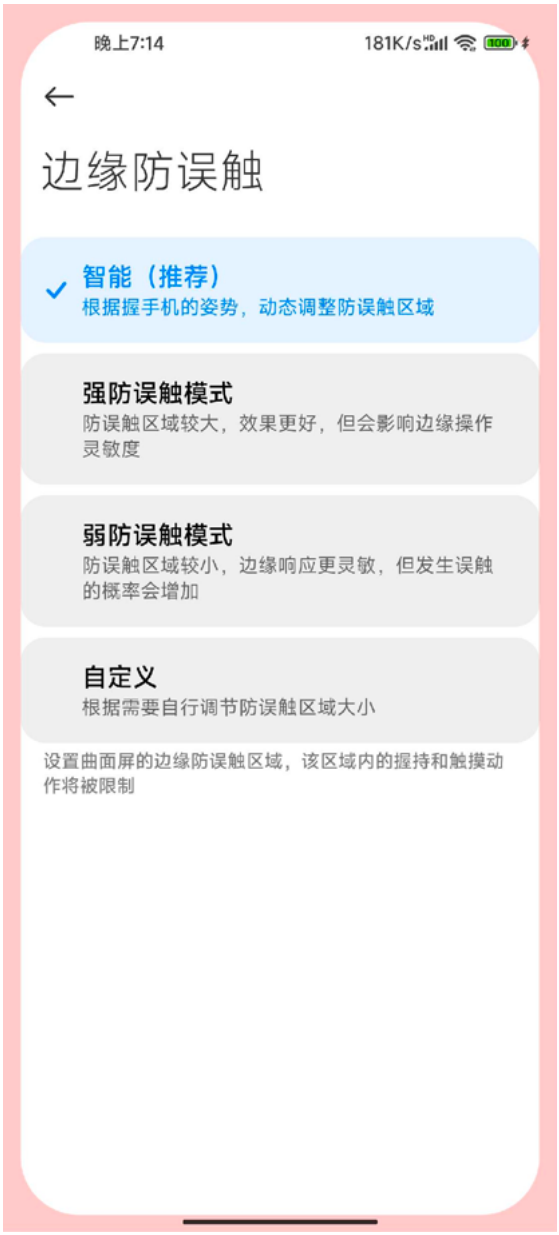


图 4.6-9 边缘防误触功能

4.6.3.3 智能感知服务

技术介绍

智能感知服务基于传感器外设设备（环境光传感器，加速度计传感器，陀螺仪，地磁传感器，距离传感器，TOF，电容传感器，气压计，温湿度传感器、低功耗相机等），在传感器融合算法中引入机器学习技术，为 OS 使用者提供丰富的定制化功能以及原始传感器数据做三方客制化的开发。

同时，为 OS 提供了多个相对独立的功能模块，服务于操作系统的基础体验。

技术架构

图 4.6-10 智能感知服务架构

关键技术

1. **自动亮度相关的特有技术：**
- "360° " 光感 / 色温检测技术
- 基于环境光传感器和运动类传感器的光源无损还原技术
- 基于运动状态检测和 AON 场景识别的智能调光技术
- 自动亮度调光策略及自动亮度客制化聚类曲线技术
2. **基于传感器运动和环境检测的防误触技术：**
- 基于使用者特定动作检测的防误触算法技术
- 基于多传感器数据融合加机器学习模型的防误触算法技术
3. **基于硬件传感器衍生的算法技术：**
- 计步器服务
- 运动状态检测（步行、骑行、乘车、地铁、高铁、乘机等）服务
- 手持、左右手手持姿态检测
- 快捷手势类算法（背部敲击、空中签名等）技术
- 常驻地磁校准算法技术

4. **基于低功耗相机的特有场景检测技术：**
- 人脸识别技术
- 基于人脸方向的智能旋屏技术
- 基于低功耗相机的眩光场景检测技术
- 可支持功能扩展和定制的低功耗相机软件架构

5. **基于互联互通的传感器数据共享技术：**
- 支持小米澎湃 OS 设备的传感器数据互传技术

6. **屏下环境光传感器的特有技术：**
- 不同芯片平台共用的屏下光感算法接口及软件架构
- 基于机器学习方式实现的显示内容实施转化为环境光亮度影响技术

4.6.3.4 智慧充电引擎

技术介绍

通过一系列智慧算法策略以及和上层用户交互、用户习惯学习，优化用户低电、导航、户外、重载等场景下的充电速度温升平衡或电池循环次数，提高充电体验和延长电池寿命，以提升小米产品的竞争力。

技术架构

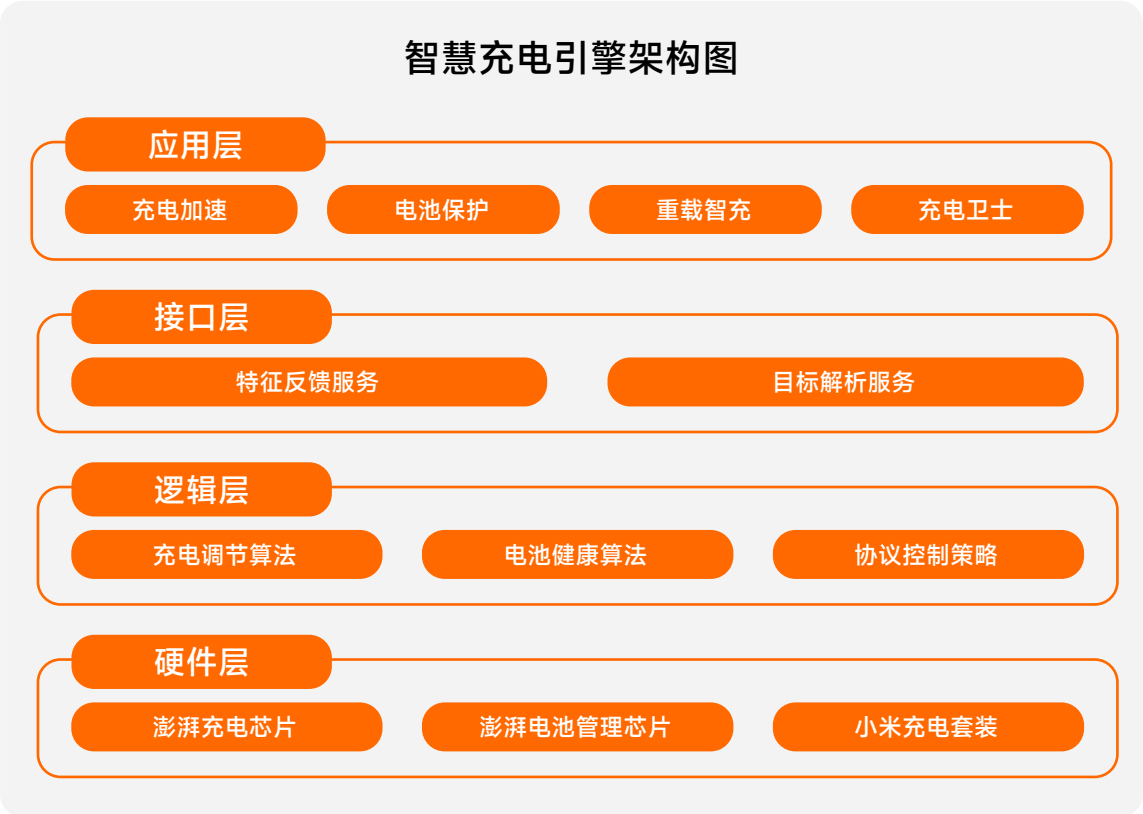


图 4.6-11 智慧充电引擎架构

关键技术

1. **导航断充算法：**打开“智能断充保护”功能后，可学习常用的地图软件使用导航功能习惯，静默通知用户断充保护生效，解决长时间导航和充电并发场景下电池过饱和时衰减加快的痛点问题。（注：不同产品、软件版本，在功能、界面上可能存在差异，请以产品实际情况为准）

图 4.6-12 电池保护功能

2. **低电疾充策略：**打开“低电疾充”功能后，手机在较低电量状态下连接原装充电器进行息屏充电时，会启动加速充电，满足用户低电量快速补电的较强需求，降低低电焦虑。此功能生效时，手机温度可能略有升高。（注：不同产品、软件版本，在功能、界面上可能存在差异，请以产品实际情况为准）

图 4.6-13 更多电池功能

3. **户外充电模式**：当充电场景智能侦测技术识别到用户在陌生环境或户外进行充电时，会针对专用充电端口类型进行定向电流提升，解决户外用户使用共享充电宝充电缓慢并且收费较贵的痛点问题。
4. **重载智充**：凭靠小米澎湃快充芯片的高效率优势，在用户边充电边重载使用手机场景下，通过小米自研多模提效算法进行快充芯片工作模式的自适应动态切换，实现同等充电电流下的手机温升降低约 1°C或者同等温升下充电速度的小幅提升。

4.6.3.5 振感框架

技术介绍

振感框架是一套振动波形的生成框架，可以通过参数或者 HE 格式为你的线性马达机器定制出丰富的振动效果。

技术架构

图 4.6-14 振感框架架构

关键技术

1. **波形采样率提升：**波形算法优化，提升波形采样点，增加到 16 个甚至更多。可以提升波形的材质感和纹理感。
2. **混播算法：**振动事件时间可以重合，重合的两个振动事件以混播的方式振动。在同一时间点同时发生两个振动事件，混播可以同时实现这两个层次的感受。
3. **频率与强度范围：**强度与频率范围为 -50~100。
4. **更多 Pattern 支持：**HE 文件支持不限数量的 Pattern，一个 HE 文件可以配合一个完整长视频振动。

4.6.3.6 指纹服务

技术介绍

指纹服务基于多种类型的指纹传感器（电容指纹、屏下光学指纹、屏下超声波指纹等），在保证用户设备安全的基础上为用户提供更便捷、快速的解锁服务。

技术架构

图 4.6-15 指纹架构

关键技术

1. **安全**：基于生物认证技术，在 TEE 安全环境中进行指纹信息的采集、加密存储和匹配处理，具备支付级安全等级。
2. **快速**：融合机器学习技术，快速进行指纹识别匹配，并与小米澎湃 OS 输入外设系统服务、视觉服务引擎紧密协作，为用户提供快速便捷的指纹解锁体验。

4.7 编译器与运行时子系统

4.7.1 子系统简介

编译器与运行时子系统的职责是可依据目标产品的不同，实现可大可小的小米澎湃 OS，不仅支持子系统裁剪、模块裁剪，还实现了模块可按需选择不同类型的编译器与运行时，将不同模块的源代码（如 C、C++、Java、TS 等）转换成机器代码，并针对不同编程语言提供特定的、可靠的、高效的运行时环境。

4.7.2 子系统架构

编译系统是基于 Ninja 和 Bazel 的现代化编译系统，设计了统一的产品配置模版，支持各种不同品类的产品配置其子系统及模块，编译时按配置文件自动选择正确的代码并编译目标系统，编译过程中采用了小米自研的并行编译技术、远程缓存算法，比传统的安卓构建编译快 5 倍。

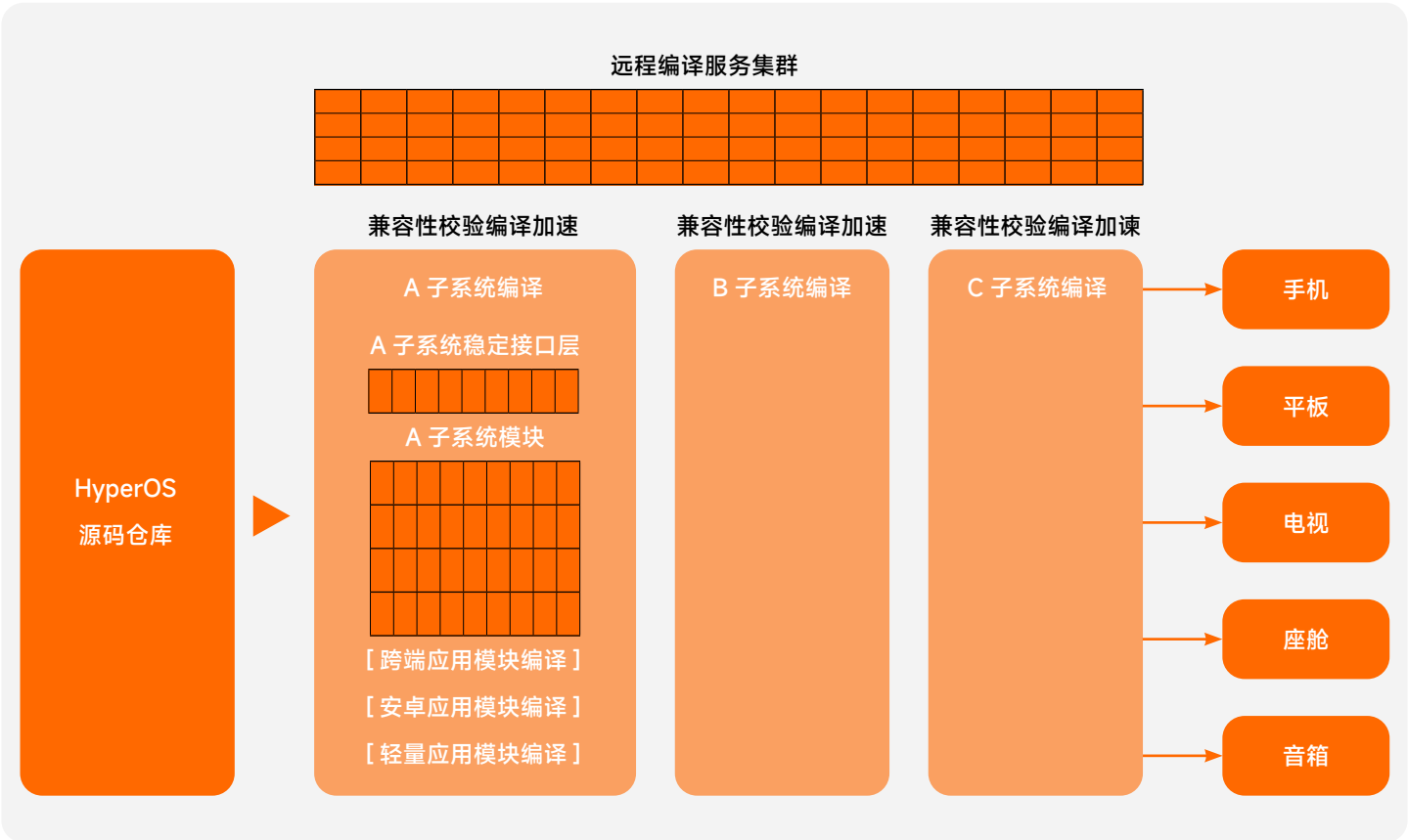


图 4.7-1 编译器与运行时子系统技术架构

流程主要包括以下步骤：

1. 配置解析：解析待编译的配置文件，加载相关配置，计算依赖关系。
2. 子系统动态规划：根据解析的配置（如子系统、编译类型），动态选择目标编译工具链和编译选项，裁剪不必要的子系统。
3. 子系统兼容性检查：启动预编译检查工具，对依赖的子系统及其稳定接口层的兼容性信息进行检查，包括接口名称、参数、版本等。
4. 模块动态规划：解析模块配置，裁剪不必要的模块，从远程服务器下载预编译缓存、编译优化配置。
5. 子系统模块编译：根据目标模块调用对应工具链进行源码编译，在所有模块完成编后进行子系统构建。
6. 系统构建：并行完成各子系统编译构建任务，合并目标文件，构建系统。

4.7.3 重点特性

4.7.3.1 安卓应用编译与运行时

技术介绍

采用 AOT 编译策略进行安卓应用模块编译，对其部分或全部代码从字节码形式编译成本机机器码，以便在 ART 运行时获得更好的执行性能和更快的启动时间。使用预热技术优化 AOT 编译过程，对超大内存设备进行了指令集优化、指令调度、寄存器分配和内存回收策略优化，对 ART 运行时电源消耗进行优化。

技术架构

图 4.7-2 安卓应用编译与运行时架构

关键技术

1. **预编译云端热点：**分析安卓应用执行数据，如热点代码，预先在云平台为每一款应用生成一个高效、稳定的代码编译配置文件，在应用安装期间，会对配置文件中的方法执行预先编译 (AOT)，应用更新后，结合应用新版本与热点代码在云端持续分析并更新编译配置文件，设备闲时进行持续优化。
2. **分层分级垃圾回收器：**针对不同硬件资源的设备优化了最合适的垃圾回收器，优化并发标记和并发清理的策略，在标记阶段，使用并发线程来标记存活对象，而应用程序可以继续执行，减少了垃圾回收对应用程序的停顿时间。在清理阶段，同样使用并发线程来清理未被标记的对象，并回收内存空间。使用读屏障（Read Barrier）技术在并发执行时确保引用的一致性。

4.7.3.2 跨端应用编译与运行时

技术介绍

跨端应用采用 UX 编程语言，拥有现代化编程语言特性，特点是更好的代码结构化与绑定式语法，不仅对开发者友好，也易于进行运行时优化，获得更好的运行性能。

技术架构

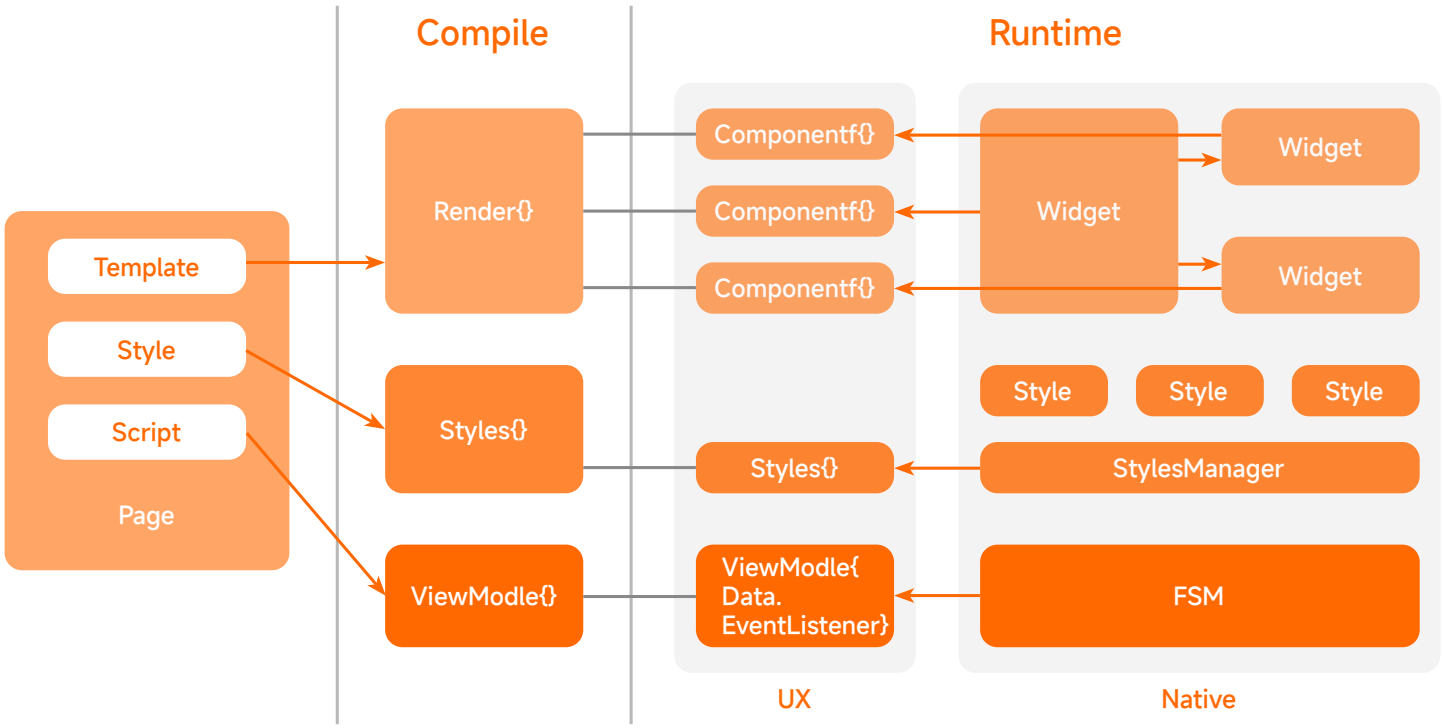


图 4.7-3 跨端应用编译与运行时架构

关键技术

1. 预编译机器码：

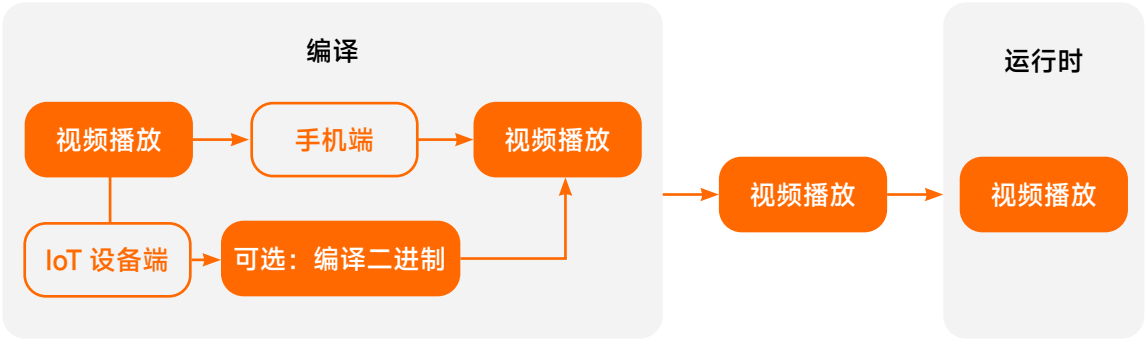


图 4.7-4 编译过程

跨端编译工具链支持在编译时对目标设备进行代码优化，将代码提前转换为机器码，消除了运行时的解析和编译过程，从而提高了执行速度和性能。

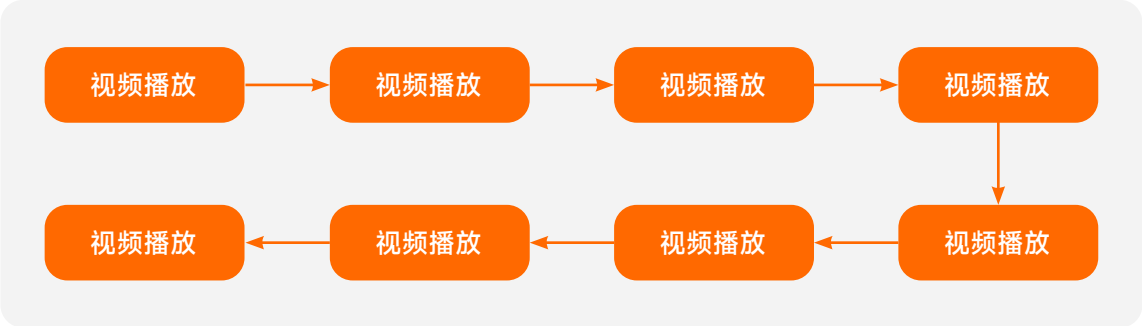


图 4.7-5 编译工具链

- 配置解析：解析项目中的配置文件，解析程序名称、版本、包名、页面配置、服务配置等信息。
- 依赖分析：根据代码中引用关系，进行各个代码模块的依赖关系分析。
- 语法转换：对 UX 进行语法转换，生成中间代码。
- 代码编译：对中间代码进行转换，生成二进制文件。
- 应用打包：获取应用的资源及二进制文件等产物，根据应用配置，使用用户签名进行应用打包。

4.7.3.3 轻量应用编译与运行时

技术介绍

跨端应用采用 UX 编程语言，拥有现代化编程语言特性，特点是更好的代码结构化与绑定式语法，不仅对开发者友好，也易于进行运行时优化，获得更好的运行性能。

技术架构

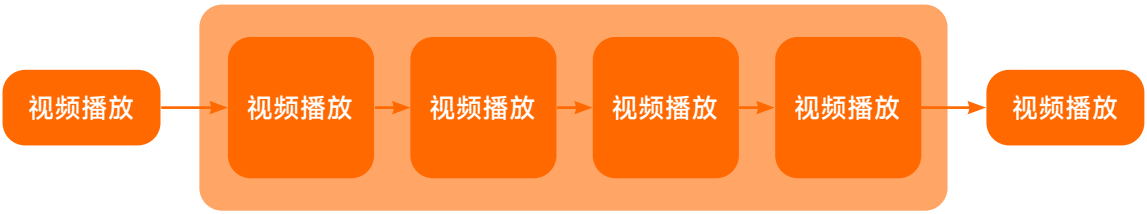


图 4.7-6 轻量应用编译与运行时架构

关键技术

1. **运行高效率：**编译器在编译 TS 源码时，会保留语言的类型信息，在编译期间尽量确定对象内存布局，进行静态化编译，以提高运行效率。另外，支持 AOT 技术，可以利用字节码中的类型信息，将 wasm 文件编译为机器码，使 TS 程序有着接近 Native 程序的运行性能。
2. **开发高效性：**保留了 TS 语言的动态特性，极大提升了前端开发人员的开发效率。对编译期无法确定的动态性，在运行时动态构建对象内存布局，并采用指令 cache 技术提高运行效率。

4.8 安全子系统

4.8.1 子系统简介

小米澎湃 OS 安全子系统结合软硬件安全技术，在硬件层、内核层、框架层和应用层使用不同的安全技术，对设备提供端到端的完整链路保护，在底层保障用户的数据安全和隐私，并在应用层为用户提供许多隐私安全特性。

- **硬件安全和可信执行环境：**小米澎湃 OS 的安全是基于一体化软硬件的平台，使用隔离的硬件资源结合可信执行环境构建整个系统的安全基石。
- **系统安全：**基于信任根建立完整的安全启动链，确保系统可信，保护内核及网络通信，对设备进行管理控制，安全高效地更新系统软件。
- **身份认证与数据安全：** 确保正确的用户才可以访问设备。基于 小米澎湃 OS 设计的数据保护架构，在保障用户数据安全的同时，也提升了 小米澎湃 OS 的易用性和便捷性。
- **应用安全：**小米澎湃 OS 确保应用的安全运行及用户数据的安全性，同时也对各种恶意软件进行治理，保证用户的良好体验。
- **安全开放框架：**小米澎湃 OS 开放各种隐私安全能力，与和各方共建良好的生态，共同保护用户的隐私安全。
- **安全认证与隐私政策：**小米在信息安全与隐私保护方面的总体原则、组织架构、安全与隐私认证、隐私政策以及持续改进机制。

4.8.2 子系统架构

图 4.8-1 安全子系统整体框架

4.8.3 重点特性

4.8.3.1 硬件安全和可信执行环境

4.8.3.1.1 硬件唯一密钥（HUK）

HUK（ Hardware Unique Key，设备唯一密钥）出厂前被固化在设备芯片上，每台设备的 HUK 都唯一且不可篡改，仅可以通过硬件加密引擎访问 HUK。HUK 为系统的锁屏密码、文件系统加密等功能所使用的密钥提供了唯一性保障。

4.8.3.1.2 硬件加解密引擎

加密和解密通常需要极大的计算能力。对于移动设备来说，计算速度、节能和安全性至关重要。通过高性能硬件加解密引擎，小米澎湃 OS 确保设备在保证安全强度的同时不对性能、续航等造成影响。

4.8.3.1.3 安全芯片（eSE）

eSE (Embedded Secure Element) 是一颗具备安全计算和安全存储能力的独立安全芯片，安全等级高，即使设备系统被 ROOT、硬件被飞线，eSE 的存储数据依然可信。eSE 的安全特性不依赖网络，可实现脱机认证和交易，被广泛运用于金融、支付和身份认证等行业。

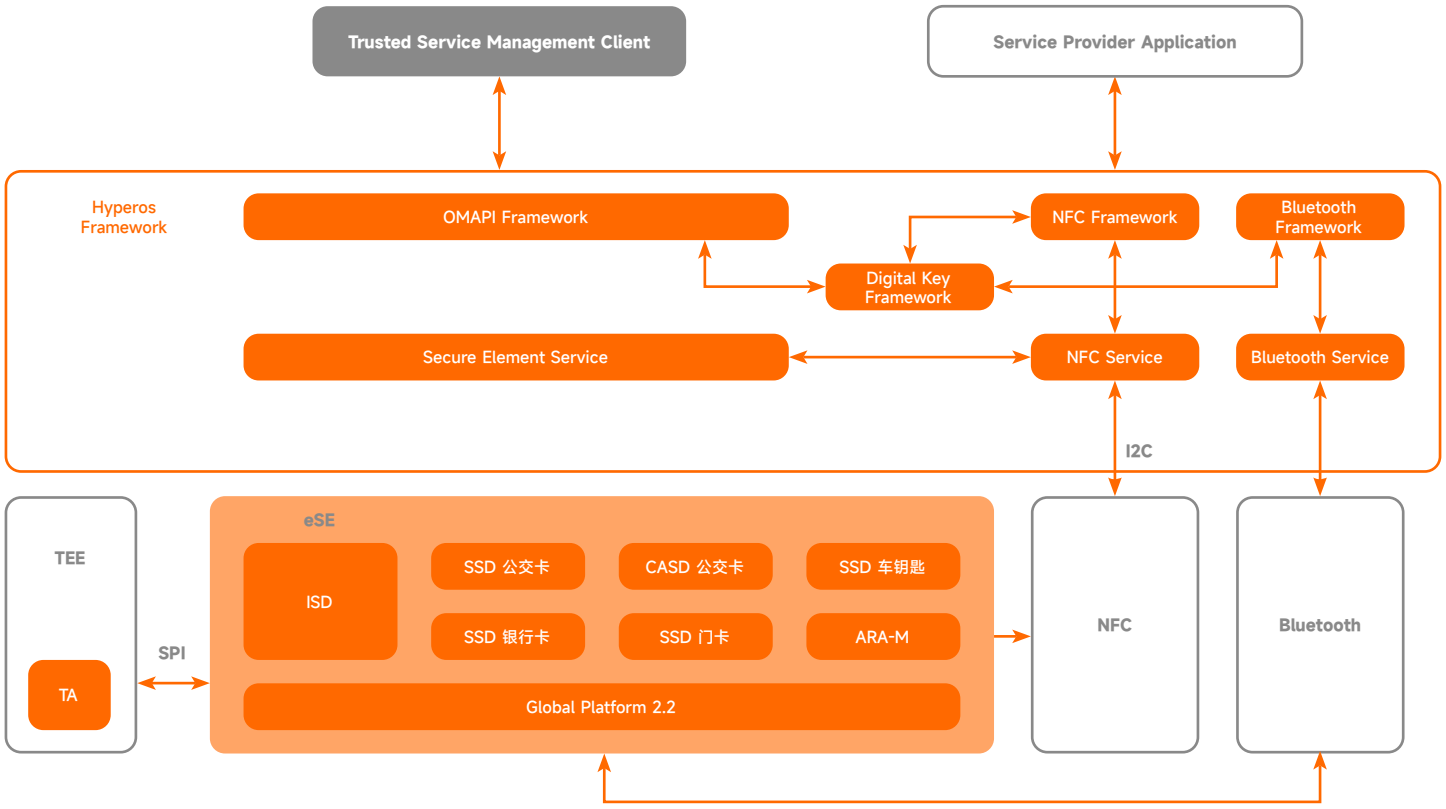


图 4.8-2 eSE 整体架构

基于 eSE 安全芯片，可以在很多生活场景中使用小米澎湃 OS 的设备：

业务名称	使用场景
数字车钥匙	使用 NFC、蓝牙、UWB 等技术连接手机和车，实现车辆的开门、启动等功能
交通卡	公交上下车、地铁出入站刷卡、线上充值、换机移卡
Mi Pay	商户 NFC 刷卡交易、线下商户主被扫二维码交易、线上 app 应用内交易
门卡	小区门禁 NFC 刷卡、办公楼宇 NFC 刷卡、智能门锁卡、换机移卡
校园卡	校园食堂、门禁、图书借阅等 NFC 刷卡
eID	深圳部分酒店、机场身份识别
数字人民币	商户 NFC 刷卡交易

4.8.3.1.4 设备证明

为了确保设备的可信度，小米给设备预装了证书，可以唯一识别每台设备。证书的公钥集中存储在小米的服务器中。在需要更高安全级别的场景中，应用程序可以向小米服务器发送认证请求，以验证设备的真实性。

4.8.3.1.5 TEE（Trusted Execution Environment）

TEE 访问的软硬件资源是与用户操作系统隔离的。TEE 提供了可信应用的安全执行环境，同时也保护可信应用的资源和数据的保密性、完整性及访问权限。TEE 内部包含了密钥管理、密码算法、安全存储、安全时钟、可信 UI 等安全服务。在 TEE 中，每个可信应用是相互隔离的，在未授权的情况下也不能相互访问。

小米澎湃 OS 基于不同的硬件环境使用不同的 TEE 方案，自研了 MiTEE 系统，同时兼容了标准高性能设备和轻量设备，保证了安全性和灵活性。

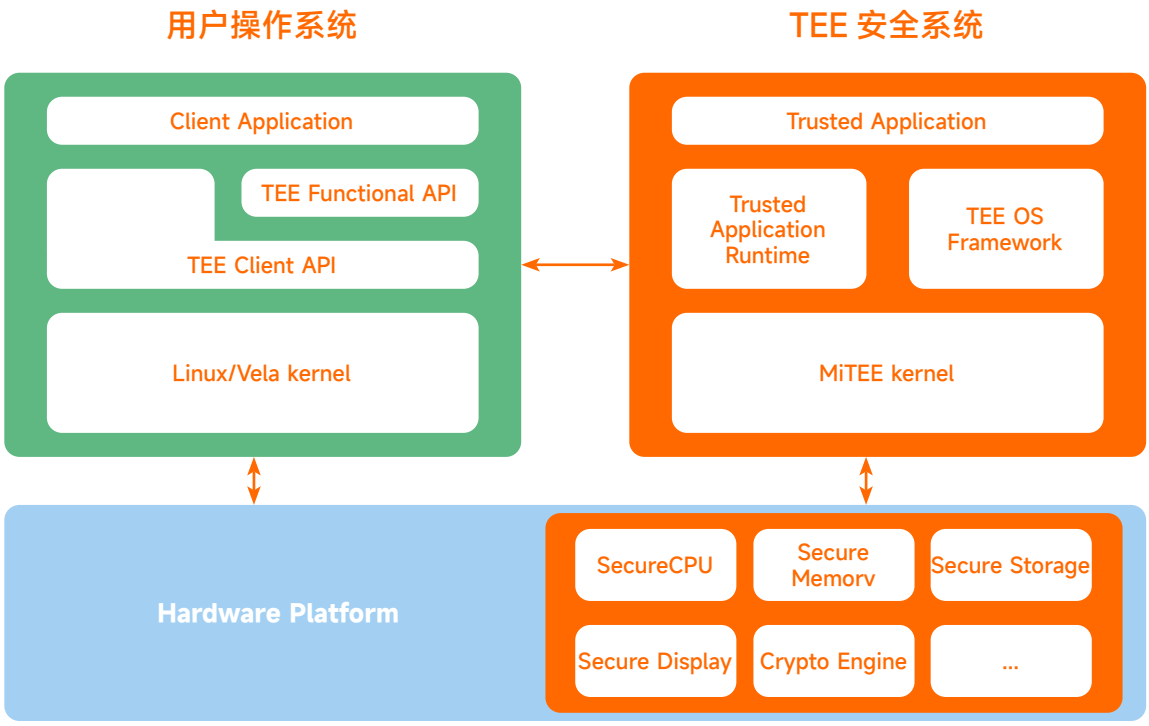


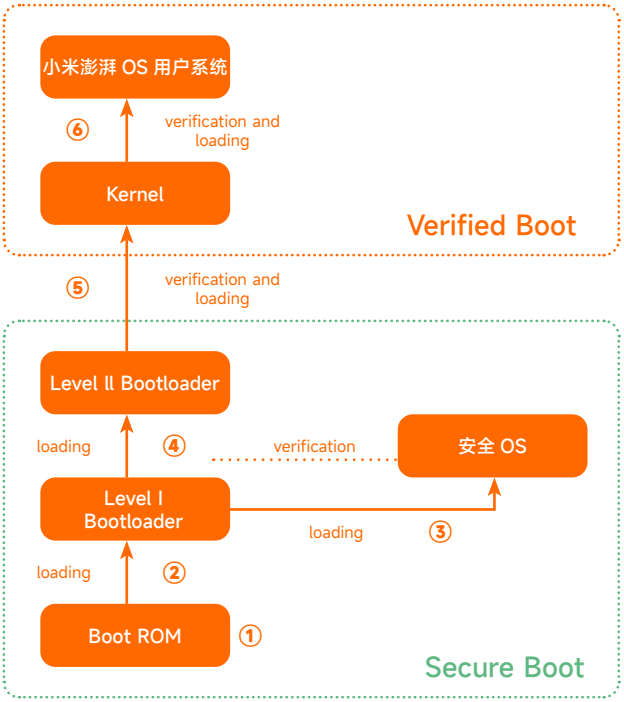
图 4.8-3 TEE 整体框架

4.8.3.2 系统安全

4.8.3.2.1 安全启动

安全启动是在系统启动过程中，通过公钥验证程序或数据的签名，确保启动文件或程序的完整和可信，以防止在启动过程中加载并运行了未经授权的程序。

在安全启动机制下，所有启动文件（如：启动引导程序、内核镜像、基带固件）均需先通过签名校验才被允许加载运行，在启动过程的任何阶段，如果签名验证失败，则启动过程会被终止。



1. 当设备通电时，PC 指针指向片内引导程序的地址，开始执行片内引导程序。
2. 片内引导程序从外部存储设备加载并验证一级 bootloader，然后跳转执行。
3. 一级 bootloader 加载安全 OS 映像文件。
4. 一级 bootloader 加载二级 bootloader，由安全 os 验证其完整性。
5. 二级 bootloader 验证并加载内核。
6. 内核进一步验证并加载 HyperOS 主用户系统。

图 4.8-4 HyperOS 安全引导过程

片内引导程序（ROM SoC Bootloader）是在芯片制造时被写入芯片内部只读 ROM 中的一段引导程序，出厂后无法修改，设备上电后最先执行此代码。

设备上电后，片内引导程序执行基本的系统初始化，从存储芯片中加载一级引导程序，并利用保存在主芯片内部 Fuse 只读空间的公钥对一级引导程序镜像的数字签名进行校验，验证成功后运行一级引导程序。随后，一级引导程序加载、校验和执行安全 OS 镜像，安全 OS 运行起来后，由安全 OS 和一级引导程序共同校验、加载和执行二级引导程序。以此类推，直到整个系统启动完成，从而保证启动过程的信任链传递，防止未授权程序被恶意加载运行。

小米澎湃 OS 系统的启动过程支持 VB (Verified Boot) 功能。在设备启动过程中，从受硬件保护的信任根到引导加载程序，再到启动分区和其他已验证分区（包括 system、vendor 和可选的 OEM 分区），无论是在哪个阶段，都会在进入下一个阶段之前验证代码可靠且没有任何已知的安全缺陷之后才会执行。VB 有助于防止永久驻留的 Rootkit 恶意软件持有 ROOT 权限危害设备，可确保设备在启动过程中的安全性。

4.8.3.2.2 内核安全

小米澎湃 OS 支持原生的 SELinux 特性，对系统中的进程、文件、目录等所有资源的操作均实施强制访问控制，任何进程在 SELinux 系统中执行操作之前都必须先在安全策略配置文件中赋予权限，而控制访问权限的策略配置文件在设备启动过程中会被保护起来，无法被第三方更改。通过 SELinux，小米澎湃 OS 可以阻止系统进程读写受保护的数据，还可以阻止系统进程绕过内核的安全机制或攻击其他进程。

小米澎湃 OS 支持 KASLR（Kernel Address Space Layout Randomization，内核地址空间布局随机化），在每次系统启动时，小米澎湃 OS 都对内核的地址空间进行随机布局，避免使用固定的内核的地址空间布局，提升代码重用攻击的难度，降低被许多复杂攻击的可能性。

小米澎湃 OS 还支持内核运行时保护机制，可以对内核及驱动模块的代码段、数据段以及关键动态数据进行保护，进一步提升了系统内核的安全性。

4.8.3.2.3 系统软件更新

小米澎湃 OS 支持通过 OTA(Over The Air) 方式更新系统软件，实现了安全、高效的系统升级管理。

系统软件更新前，系统更新程序验证 OTA 下载到设备或离线复制到设备的 ROM 镜像的完整性。在完成如文件大小、哈希值等校验后，根据设备选择不同的升级模式 :

1. **非 Virtual A/B 机型：**设备重启进入 Recovery 模式，再次验证签名密钥的正确性。校验通过后，Recovery 模式才会将更新的 ROM 包写入系统存储区域。
2. **VAB 机型：**双分区升级模式下，有系统升级时用户正常使用当前系统，升级模块会验证 ROM 的完整性，验证通过后将 ROM 镜像写入另一个分区中。更新完毕后设备重启，会自动同步更新内容并完成升级。

4.8.3.2.4 网络安全

- **安全网络协议**

  使用安全网络协议可以降低用户设备连接网络时遭遇数据泄露和篡改的风险。小米澎湃 OS 用户可以通过公共网络链路建立自己的虚拟专用网络（VPN）。小米澎湃 OS 支持多种 VPN 模式，包括： PPTP、L2TP/IPSecPSK、L2TP/IPSec RSK、IPSec Xauth PSK、IPSec XauthRSA 和 IPSec 混合 RSA。用户可以根据需要选择 VPN 模式来访问和传输敏感数据。

  小米澎湃 OS 的无线局域网连接支持 WEP、WPA/WPA2PSK、802.1×EAP、WAPI 等多种身份验证方法，满足用户不同级别的安全需求。

  小米澎湃 OS 的 WLAN 热点功能默认禁用，当用户启用该功能时，默认使用 WPA2PSK 认证方式保证连接的安全性，同时 WLAN 热点功能还支持配置终端 MAC 地址黑名单。

- **伪基站短信识别**
  伪基站是一种利用了通信系统缺陷的非法无线电通信设备，常被犯罪分子用于冒用他人手机号码向伪基站周边的设备发送诈骗短信或垃圾短信。当伪基站运行时，会干扰和屏蔽一定范围内的运营商信号，用户手机信号被强制连接到该设备，影响用户正常使用。
  即使用户尚未启用防伪基站功能，小米澎湃 OS 仍可向用户提供识别伪基站短信的功能，此功能不依赖芯片。使用人工智能机器学习，通过对伪基站接入电话的特征和伪基站发送的短信的文本特征进行分析，可以判定是否为伪基站消息。
  小米澎湃 OS 对伪基站消息的识别是在用户移动终端离线执行的。每当识别出伪基站短信，小米澎湃 OS 都会提示用户。
- **Wi-Fi 探头保护**
  通过监测其他电子设备发送的 Wi-Fi 信号携带的 MAC 地址，Wi-Fi 探头盒可以识别每个用户。小米澎湃 OS 能够发送带有随机 MAC 地址的数据包，以防止 Wi-Fi 探头获取设备的真实 MAC 地址。

4.8.3.2.5 设备控制

- **查找设备**
  小米澎湃 OS 为用户提供了查找设备功能，为用户找回丢失设备提供帮助，同时保护设备的数据安全。此功能需用户手动开启后才能使用。启用后，在设备丢失的情况下，用户可登录小米云服务网页（https://i.mi.com）远程对丢失的设备进行以下操作：查找定位、设备发声、丢失锁定和清除数据。

查找定位	允许用户通过网络或短信命令获取设备的当前位置并通过地图直观显示。
设备发声	通过网络或短信指令使设备响铃，用于查找可能就在附近的设备。
丢失锁定	通过网络或短信指令锁定设备，锁定后周期性自动上报定位位置，同时自动解绑 Mi Pay 绑定的银行卡。
清除数据	通过网络或短信指令重置设备，关闭数据同步并解绑 Mi Pay 银行卡。

- **设备锁定 / 解锁策略**
  启用查找设备功能后，设备将关联到当前登录的小米账号。在用户丢失设备或忘记小米帐号密码的情况下，设备均有可能被锁定，针对设备被锁定的情况，小米澎湃 OS 设计了多种安全策略来保障用户权利。

激活锁定	如果启用查找设备功能后，当小米设备恢复到出厂默认设置并对小米设备重新刷机时，设备将被锁定。设备只能使用关联到设备的小米账号密码进行解锁。
密码重置保护	为避免用户设备丢失后，帐号密码被使用设备验证进行重置，小米帐号重置密码后 3 天内无法关闭“查找设备”功能，为丢失设备的用户提供时间补办 SIM 卡，重新夺取帐号和设备的控制权。
客服解锁	如果用户忘记了自己的小米账号密码，无法找回密码，用户可以通过客服解锁设备。做出解锁决定所需的信息包括锁定界面或 IMEI 上的解锁代码、订单号或发票等购买证明以及小米账号的电话号码。如果购买证明被确认为无效或不完整，解锁申请将被拒绝。

此外，如果设备丢失，锁屏密码的存在使得设备极有可能被强制刷机。小米澎湃 OS 将账号与设备之间的关联存储到其云服务器（一些设备将关联状态写入不被刷机覆盖的特殊分区），保证关联关系不可篡改。启动时，要求设备连接到网络并从服务器获取真实关联。如果当前登录账号与服务器上的关联账号不同，小米澎湃 OS 将要求用户切换回关联账号后才能继续使用。

被解锁的设备可以通过强制刷机非小米澎湃 OS 系统或篡改过的小米澎湃 OS 系统来绕过设备锁定。然而，这种 ROM 不能执行 OTA 升级，也不能正常登录小米账号。当设备刷回官方小米澎湃 OS 包时，它将再次受到查找设备功能的保护。

- **手机分身**

  小米澎湃 OS 用户可通过手机分身创建一个完全独立于原系统的独立空间，用户的帐户、应用和数据等与主空间隔离，并分别加密保护。用户可以通过设置不同的解锁密码进入主空间和分身空间，达到如同拥有第二部手机一般的虚拟手机体验。用户可以通过对这个分身空间设置访问密码，保存各类私密文件、图片等信息，安装私密应用等。这个分身空间又类似于一个“沙箱”，在这个“沙箱”内进行任何的操作，都不会对主空间造成影响。

- **移动设备管理（MDM）**

  MDM（Mobile Device Management，移动设备管理）是小米澎湃 OS 提供给设备管理类应用的设备保护功能，是对设备进行管理和操作的接口。通过 MDM 应用和小米澎湃 OS 提供的 API 接口，企业 IT 系统可以轻松实现对小米澎湃 OS 设备的控制和管理。API 接口调用需要授权，保证接口调用的权限管控和安全性。

  对于引导或提供非正常使用设备管理器权限的应用，按标准执行系统管控策略，包括但不限于：在系统内强提醒用户进行关闭处理、禁止应用获取服务或权限接口。

  对于引导或提供通过设备管理器权限，对用户的数据、设备使用安全可能产生危害的应用，将严格执行如下操作：将该应用在小米应用商店进行下架处理、禁止应用获取相关服务接口、禁止相关应用在设备管理器应用列表中显示。

4.8.3.3 身份认证和数据安全

4.8.3.3.1 锁屏密码

小米澎湃 OS 锁屏密码支持绘制模式、数字密码和混合密码，每种密码都有最小密码长度要求，以确保更安全的密码。

- **绘制图案密码：**至少需要连接 4 个点。
- **数字密码：**这些密码支持 4-16 位之间的长度。
- **混合密码：**这些密码支持大小写字母、数字和长度在 4-16 个字符之间的符号的任何组合。

小米澎湃 OS 锁屏密码受硬件唯一密钥（HUK）保护，并在 TEE 中加密。当用户创建或修改锁屏密码，或使用锁屏密码解锁屏幕以进行验证时，锁屏密码将在 TEE 中处理。

小米澎湃 OS 限制输入错误密码的次数。连续多次尝试使用错误密码后，手机将被锁定，以防止锁屏密码被暴力破解。

在部分产品上，小米澎湃 OS 基于安全芯片，支持基于安全芯片的锁屏密码增强。对于锁屏密码的校验的错误计数等逻辑均运行在安全芯片中，提供抗硬件物理攻击的安全强度。

4.8.3.3.2 生物识别

小米澎湃 OS 提供指纹和人脸两种生物识别方式。

指纹识别

指纹识别功能是小米手机优化用户体验的特色功能设计之一。指纹识别基于生物认证技术兼具强密码安全性的特点，只需用户将手指放到设备的指纹传感器位置，根据用户提前录入的一个或多个指纹特征信息，就能快速解锁访问自己的设备。

考虑到密码是用户通过加密方式保护其个人数据的根基，指纹识别的目标是在保证用户设备安全的基础上为用户提供更便捷、快速的解锁服务，因此指纹识别功能不会取代设备的密码解锁模式。用户在每一次开启 / 关闭指纹识别功能前都需要输入密码，并在以下场景中，只能通过输入密码才能继续操作设备，以保障用户设备及器个人数据的安全：

- 当用户的设备刚刚打开或重新启动时；
- 当设备超过 72 小时未使用密码解锁时；
- 当用户的设备连续指纹解锁失败达到 5 次以上，20 次之内时，每次指纹解锁失败会临时锁定指纹功能 30 秒；
- 当用户的设备连续指纹解锁失败达到 20 次时。

人脸识别

人脸识别功能是小米手机优化用户体验的特色功能设计之一。人脸识别基于生物认证技术兼具强密码安全性的特点，通过人脸识别 AI 算法智能检测人脸面部特征并进行高精度匹配，让用户只需注视设备屏幕，就能快速解锁访问自己的设备。

考虑到密码是用户通过加密方式保护其个人数据的根基，人脸识别的目标是在保证用户

设备安全的基础上为用户提供更便捷、快速的解锁服务，因此人脸识别功能不会取代设备的密码解锁模式。用户在每一次开启 / 关闭人脸识别功能前都需要输入密码，并在以下场景中，只能通过输入密码才能继续操作设备，以保障用户设备及个人数据的安全：

- 当用户的设备刚刚打开或重新启动时；
- 当设备超过 72 小时未使用密码解锁时；
- 当用户的设备连续人脸识别失败达到 5 次时。

充分考虑安全与用户隐私的生物识别机制。小米澎湃 OS 一直以来都将用户的隐私保护置于各项产品和服务的重要地位，对于 AI 技术及产品而言也没有例外。我们将隐私设计的理念贯穿于生物识别产品生命周期的各个阶段，不仅符合法律的要求，更在于追求实现最佳实践。我们深知用户生物特征信息对于用户的重要性，在生物识别功能设计初期，考虑到对数据采集、传输、存储、销毁等各个处理环节可能存在的风险，通过识别风险、制定策略和技术措施防护，监督落实等方法，在为用户提供更优使用体验的同时，力求将安全与隐私风险降到最低。我们承诺只有在用户授权和明确同意的情况下，才会保存和调用生物特征值信息。

仅采集用户不可逆的生物特征值信息。生物识别功能需要用户提前录入生物特征模板，在录入的过程中，AI 算法提取人脸或指纹的特征值信息后，仅保存提取后不可逆的生物特征值信息，不会保留用户原始的生物信息。生物特征值信息的不可逆性也降低了通过人脸特征值信息还原原始人脸生物信息的可能，进一步保障了用户原始人脸生物信息的隐私性。

生物特征信息仅在用户设备本地存储和使用。生物识别 AI 算法集成在设备端本地运行，人脸特征信息的录入、存储、验证过程都在用户的设备本地完成。我们不会将用户的人脸特征数据备份或上传到任何包括小米服务器在内的其他平台，从根本上避免了由于数据传输过程中可能存在的安全与隐私风险。

构建算法安全架构，提供安全的运行和存储环境。小米澎湃 OS 通过构建基于硬件隔离技术的生物识别安全架构，在 TEE 环境中进行完成对生物特征信息的采集、存储和验证，保障生物特征信息数据不被任何第三方应用获取。

- **在采集的过程中。**设备基于硬件安全机制（比如 2D RGB 镜头方案的安全摄像头），使小米澎湃 OS 的用户侧只能控制生物信息采集设备的打开 / 关闭等操作，不能直接读取生物信息采集设备采集到的原始信息。
- **在存储的过程中。**用户的生物特征信息存储在 TEE 提供的安全文件存储系统中，安全文件存储系统的加密密钥存储在芯片内部，只有 TEE 可以访问，即使是生物识别 TA 也不能直接访问密钥，保证攻击者在入侵内核和平台后也无法读取和篡改生物特征值信息。
- **在验证的过程中。**TEE 环境外的小米澎湃 OS 只能发起生物特征认证的请求和接收认证结果，生物特征识别算法的生物特征信息比对过程在 TEE 环境下进行，任何 TEE 环境外的应用都无法访问生物特征数据本身。

用户可在本地对生物特征数据进行管理。生物特征信息仅保留在设备本地，不会上传或备份到任何包括小米服务器在内的其他平台。用户可以通过设备的“设置 - 指纹、面部与密码”页面下对已录入的生物数据进行管理如添加、命名、删除。为保证用户信息的隐私与安全，用户在进入生物数据管理界面前，每一次都需要用户输入设定的密码进行身份验证。

用户可随时选择关闭 / 开启生物识别功能。用户可以通过设备的“设置 - 指纹、面部与密码”页面随时选择开启或关闭人脸解锁功能。在每一次开启或关闭生物识别功能时，都需要用户输入密码进行身份验证。停用生物识别功能后，生物识别服务将不会被调起和使用，启用时已保存在生物特征数据不会自动删除，需要用户在相应的生物信息管理页面人工进行删除操作。

4.8.3.3.3 通用密钥库

小米澎湃 OS 的通用密钥库主要用于管理应用程序开发人员使用的密钥和证书的生命周期，同时还为硬件隔离环境中（部分设备为 TEE，部分设备为 TEE 和安全芯片双环境）的设备证书提供远程认证。通用密钥库具有以下功能：

生成和存储

小米澎湃 OS 的通用密钥库提供了硬件保护的密钥存储机制，应用程序中生成的密钥是加密的，只能由相应的设备使用。

加密和解密

当应用程序需要使用该密钥时，将先前生成的加密密钥和待加密数据发回相应设备的硬件隔离环境中，该密钥只能用于在相应设备的硬件隔离环境中执行加密和解密操作。

密钥认证

每一部小米手机在制造时，都在硬件隔离环境内部安全生成一对公私钥，其中公钥证书被导出，存储在云服务器中，用于安全认证每一台小米手机的合法性。基于该公私钥对，进一步支持了远程密钥认证功能，提供三方网络服务在无法窥探用户隐私的前提下，对小米澎湃 OS 设备进行合法性认证。

小米澎湃 OS 通用密钥库的技术基础是硬件隔离环境，通过防止密钥提取和密钥使用授权等措施，防止在设备外部和设备上未经授权使用密钥材料。

预防提取

为避免在小米澎湃 OS 设备之外以未经授权的方式使用密钥材料，通过 Keystore 密钥执行加密操作时，应用会将待签署或验证的明文、密文和消息发送到执行加密操作的系统进程，而不是应用进程。因此，即使应用进程遭受攻击，攻击者也无法提取密钥材料。同时，小米澎湃 OS 还将密钥材料绑定到小米设备可信执行环境的安全硬件，使其不会暴

露于安全硬件之外。即使 小米澎湃 OS 操作系统遭受攻击或者攻击者读取到设备的存储空间，也无法从设备上提取这些绑定安全硬件的密钥材料。

密钥使用授权

为了减少小米澎湃 OS 设备上密钥的未经授权使用，在生成或导入密钥时，Keystore 会让应用指定密钥的授权使用方式。一旦生成或导入密钥，其授权就不能更改。以后每次使用密钥时，都会由 Keystore 强制执行授权。

小米澎湃 OS 中支持的密钥使用授权分为以下类别：

- **加密**：授权的密钥算法、操作或目的（加密、解密、签名、验证）、填充方案、块模式和可以使用密钥的摘要。
- **时间有效性间隔：**密钥被授权使用的时间间隔。
- **用户身份验证：**只有当用户最近进行了身份验证时，才可以使用密钥。

4.8.3.3.4 用户数据加密

基于锁屏密码和通用密钥库机制，小米澎湃 OS 实现了一套完整的用户数据加密方案。小米澎湃 OS 文件系统分为系统分区和用户分区。系统分区是只读的，与用户分区隔离。常见的应用程序只能访问系统分区的部分目录。在用户分区的情况下，系统提供了基于文件的数据加密和目录权限管理机制，以限制不同应用程序之间的数据访问。此外，小米澎湃 OS 提供了更多基于硬件隔离加密技术的安全功能和应用程序，提高了小米澎湃 OS 的便利性和可用性，同时也保护了用户数据安全。

文件级加密

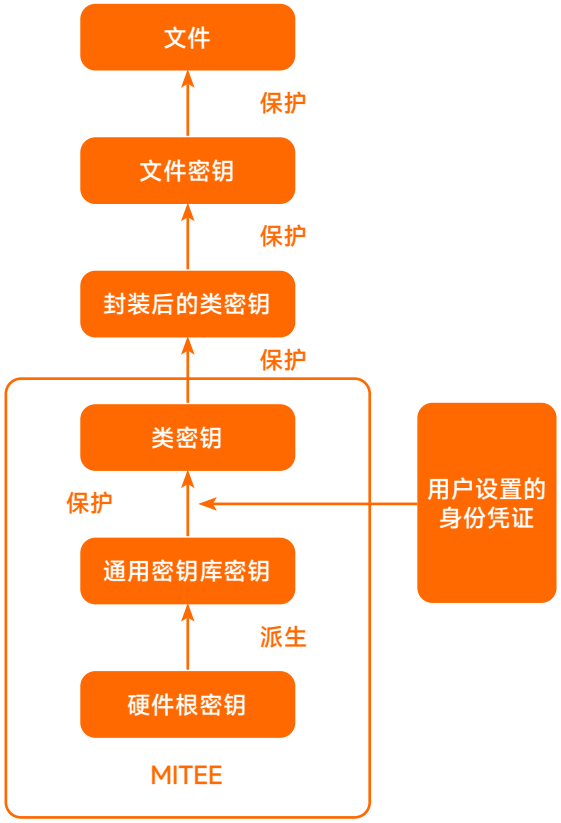
小米澎湃 OS 基于 TEE 的安全能力，结合芯片提供的硬件加解密引擎及内核加密文件系统模块，基于对称密码算法提供了文件级的用户数据加密保护。

基于小米澎湃 OS 的小米设备都为每个用户提供了两个应用程序存储位置：

凭据加密（CE）存储区域：CE 区域是默认存储区域，只有设备重启后，用户首次使用锁屏密码解锁设备才能访问。

设备加密（DE）存储区域：无论是否解锁，设备通电后都可以访问 DE 区域。

CE 存储区域是小米澎湃 OS 中应用程序存储数据的默认存储区域，以确保应用程序和应用程序数据的安全性。只有无线身份验证、闹钟、铃声、蓝牙等应用程序才会在 DE 存储区域中存储某些数据。这确保了某些基本服务将在用户提供凭据之前运行，同时系统继续保护用户隐私信息。



1. 用硬件根密钥派生通用密钥库密钥。
2. 使用通用密钥库密钥和用户身份凭证加密类密钥。
3. 在启动系统时，为每个类密钥进行密码学封装，用于防止类密钥的明文在用户环境中暴露。
4. 使用封装后的类密钥加密和保护文件密钥。
5. 使用文件密钥加密文件。

图 4.8-5 基于文件的数据加密过程

安全存储

小米澎湃 OS 的安全存储功能由基于 TEE 的安全文件系统（SFS）实现，用于安全存储密钥、证书、指纹模板等敏感信息。运行在 TEE 中的可信应用程序（TA）使用安全存储应用编程接口加密和存储数据。加密数据只有 TA 才能访问，因此外部应用程序无法访问。

小米澎湃 OS 中的安全存储采用高强度对称密码进行加密、解密和真实性完整性封装。安全存储密钥来自硬件唯一密钥（HUK），并始终存储在设备的 TEE 中。通过密钥加密的数据不能在 TEE 之外解密。

小米澎湃 OS 进一步提供 RPMB（重放保护内存块）分区功能，以保护某些系统数据免受未经授权的删除和访问。RPMB 分区由 TEE 直接控制，用于安全，并绑定到硬件唯一密钥（HUK）派生的密钥。只有 TEE 可以访问 RPMB 保护的数据，外部用户环境不提供访问 RPMB 的接口。RPMB 通过内置计数器、密钥和 HMAC 验证机制防止重放攻击，以确保数据不能被恶意覆盖或篡改。

安全擦除

一般的“出厂重置”并不保证存储在物理存储中的数据会被彻底擦除。为了提高效率，这通常是通过删除逻辑地址来实现的。然而，物理地址空间实际上并没有被擦除，数据可以被恢复。小米澎湃 OS 为用户提供了当他们希望恢复出厂设置时格式化模拟 SD 卡的选项。一旦选择了这个选项，系统将格式化存储空间并完全擦除数据，以保护设备出售或报废后的用户数据安全。

4.8.3.4 应用安全

4.8.3.4.1 照明弹

又称为**应用行为记录**。在应用发生特定敏感行为时，小米澎湃 OS 会进行记录并展示给用户，主要包含了自启动与关联启动记录、敏感权限使用记录、跨端行为记录。同时敏感权限中的相机、麦克风和位置在使用中会在状态栏有明显的状态提醒，用户可以实时查看哪些应用正在使用这三个权限，并支持用户对应用进行关闭。实时保护用户个人隐私知情权。

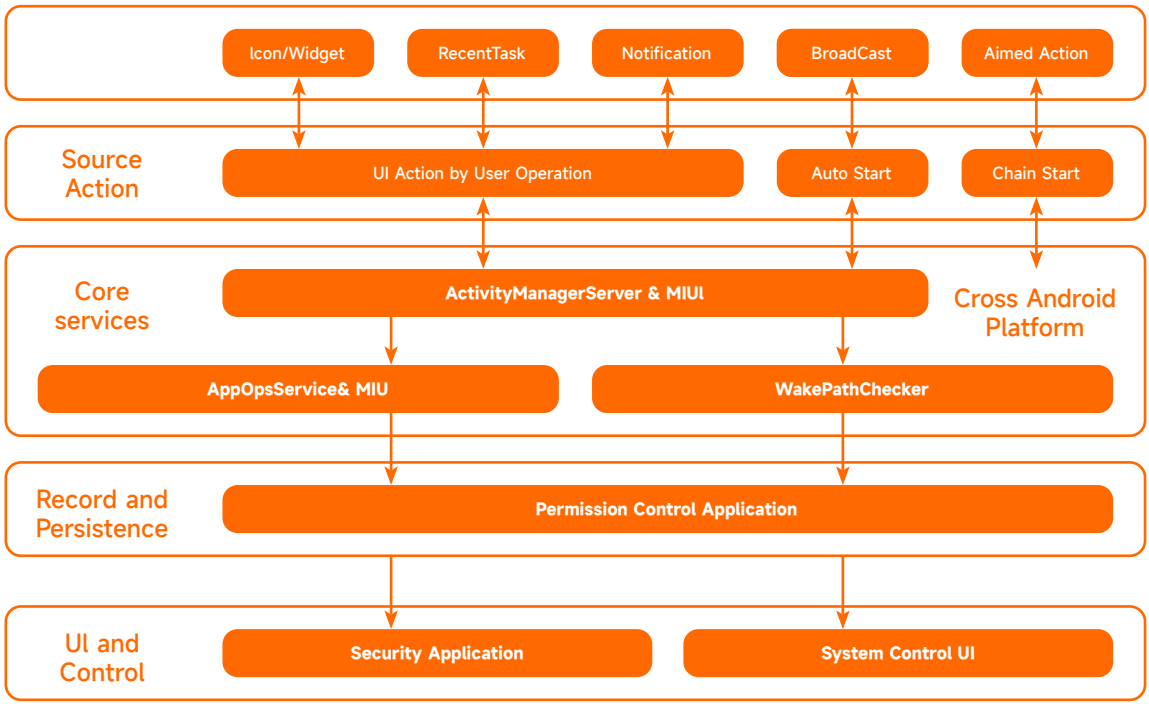


图 4.8-6 应用行为记录整体框架

4.8.3.4.2 拦截网

小米澎湃 OS 支持多个权限的多种授权状态，除了允许与拒绝外，还有仅在使用中允许与本次使用中允许。

仅在使用中允许：仅允许进程状态处于前台的应用使用该权限，后台应用访问会被拒绝。

本次使用中允许：用户在一次连续使用过程中允许使用该权限，一旦进程被杀或者用户停止使用一段时间后，权限将被自动收回。

4.8.3.4.3 防追踪

小米澎湃 OS 禁止三方应用获取不可重置标识符，如 IMEI、MEID、IMSI、Mac 地址、Serial 号等。同时对外开放了一套用户可重置可关闭的虚拟身份 ID，同时智能识别并禁用应用内嵌入的各类型追踪器，例如通过手机存储共享个人标识，共用 SDK 追踪等。

4.8.3.4.4 权限管理

小米澎湃 OS 将权限分为不同的类型，包括安装时权限、运行时权限和特殊权限。每种权限类型都指明了当系统授予应用该权限后，应用可以访问的受限数据范围以及应用可以执行的受限操作范围。

安装时权限

安装时权限授予应用对受限数据的受限访问权限，或允许应用执行对系统或其他应用只有最低影响的受限操作。如果在应用中声明了安装时权限，应用商店会在用户查看应用详情页面时向其显示安装时权限通知。系统会在用户安装您的应用时自动向应用授予权限。

小米澎湃 OS 提供多个安装时权限子类型，包括一般权限和签名权限。

运行时权限

运行时权限也称为危险权限，此类权限授予应用对受限数据的额外访问权限，或允许应用执行对系统和其他应用具有更严重影响的受限操作。因此，需要先在应用中请求运行时权限，然后才能访问受限数据或执行受限操作。

当应用请求运行时权限时，系统会显示运行时权限提示，用户可以选择允许、拒绝以及仅在使用中允许等选项。

许多运行时权限会访问用户私人数据，这是一种特殊的受限数据，其中包括可能比较敏感的信息。例如，位置信息和联系信息就属于用户私人数据。

特殊权限

声明特殊权限的应用会显示在系统设置中的**特殊权限设置**页面内。如需向应用授予特殊权限，用户必须转到此页面：**设置 > 隐私与安全 > 其他权限 > 特殊权限设置**。

如显示在其他应用上层权限，不会弹出授权弹窗，需要跳转单独的管理界面。

4.8.3.4.5 安全扫描

病毒扫描

集成了多个知名安全厂商（腾讯，安天）的病毒查杀引擎及小米自研病毒引擎。

提供本地及云端查杀能力， 确保设备在有无联网的情况下都能发现应用的安全风险。

病毒查杀引擎支持应用安装时检测，用户也可以在手机管家中手动触发设备病毒扫描功能，对恶意应用和恶意文件进行扫描。

异常检测

此外，“手机管家”-“异常检测”功能还提供了 ROOT 安全检测以及手机性能、操作、耗电等异常检测， 主要包括以下功能：

手机性能异常检测	**检测 APP 是否开启了辅助功能、设备管理器，手机剩余内存是否不足等**
操作异常检测	**检测是否开启了飞行模式、拦截联系人来电、拦截陌生人来电、勿扰模式、护眼模式等**
耗电异常检测	**检测自启动应用是否过多，是否开启了热点等**
其他异常检测	**检测系统是否被 ROOT，存储空间不足时提示用户无法安装应用等**

4.8.3.4.6 运行时保护

应用沙箱

小米澎湃 OS 使用应用沙箱机制，确保每个应用运行在沙箱中且相互隔离，保证应用运行时安全。

运行时内存保护

小米澎湃 OS 支持地址空间布局随机化（Address Space Layout Randomization，ASLR）及数据执行保护技术（Data Execution Prevention，DEP）。ASLR 提供对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，增加攻击者预测目的地址的难度， 防止攻击者定位攻击代码位置，达到阻止溢出攻击的目的。ASLR 技术提高了攻击者在利用内存漏洞 上的难度。DEP 机制会把内存中的特定区域标注为不可执行区，以防止内存漏洞攻击。

安全输入

用户在输入密码时，小米澎湃 OS 自动启用安全键盘。安全键盘不具备联想和记忆功能，没有联网权限，禁止后台录屏或第三方应用截屏， 禁止第三方应用悬浮窗覆盖在安全键盘之上，确保用户的密码输入安全。

应用锁

应用锁既可保护应用数据安全，同时又防止应用中的隐私信息被他人窥见。 小米澎湃 OS 用户可通过“应用管理”进入“应用锁”模块，为应用设置多种样式的解锁密码（图案、数字、混合），用户可设置在退出应用后或是退出应用 1 分钟后锁定，以及在锁屏后再次打开应用时验证应用锁。为了增加解锁的便捷性和安全性，小米澎湃 OS 增加了指纹和人脸的生物识别解锁机制。

屏幕共享防护

用户在使用共享屏幕等功能时，小米澎湃 OS 会在状态栏进行明显的状态提醒，防止用户忘记处于共享状态。同时，通过虚拟屏幕技术，将短信、金融软件类等敏感数据、悬浮通知隐藏或模糊起来，保护用户不会泄露验证码等个人隐私数据，导致财产损失等。

4.8.3.4.7 安全支付

使用支付类应用时，将会自动检测系统环境是否处于安全状态。如若发现风险项，将会通过弹框等形式进行提醒，帮助用户判断是否继续完成交易，以降低可能的风险带给您的不必要损失。

检测的因素主要包括：

- 当前交易环境中连接的 WIFI 是否安全
- 使用的输入法是否安全（检测用户的输入法是否为白名单中的正版安全输入法）
- 系统中是否存在病毒应用（检测后台进程是否有木马、病毒运行）

4.8.3.5 安全开放框架

应用开发者存在被恶意应用攻击的风险。同时，部分三方应用的人脸识别、指纹识别等功能并未采用硬件级方案，存在安全隐患。为了帮助开发者抵御恶意应用威胁，提高安全保护级别，HyperOS 通过安全开放框架为应用提供系统级安全开放能力。安全开放框架以 SDK 的方式为开发者提供了设备可信服务、系统安全检测、硬件级安全保护三方面的功能。

4.8.3.5.1 设备可信服务

HyperOS 基于可信执行环境 TEE 提供系统安全检测结果，在 TEE 中动态获取系统完整性数据，可信度高，并通过设备可信服务将设备状态数据提供给应用，为应用进行业务风控提供高可信的系统环境检测数据。

4.8.3.5.2 系统安全检测服务

恶意应用检测服务

HyperOS 基于端侧扫描为应用提供了获取当前系统是否安装了恶意应用的能力，应用可以通过恶意应用检测服务获取到的恶意应用类型，来确定需要采取的风控策略。

虚假点击检测服务

HyperOS 为应用提供了系统级的虚假点击检测服务，基于当前系统的运行时状态以及点击事件触发方式和特征动态判断当前设备是否存在虚假点击的行为，提升应用的风控能力。

4.8.3.5.3 硬件级安全保护

安全人脸验证

通过硬件级安全防护，保障在线人脸验证的真实性。

设备安全能力

提供可信应用的开发和部署能力，允许三方可信应用部署在 HyperOS 可信执行环境中，通过高安全能力支撑应用安全业务场景。

4.8.3.6 安全认证与隐私政策

小米公司以尊重和保护用户隐私与安全，致力于打造让用户信任的产品，享受科技的乐趣为目标。

为确保信息安全与隐私保护策略的贯彻执行，小米在 2014 年就正式成立信息安全与隐私委员会，通过技术防护、流程制度、评估和审查机制等建立了完善的安全管理体系。同时，小米聘请了欧盟当地的资深律师作为欧盟业务的数据保护官，以确保小米符合各国法律法规的要求。

为向用户提供符合法律法规及业界标准要求的业务运行环境及服务，小米已开展全球化合规治理工作，并接受外部监管机构的定期审查。小米已经通过多种隐私安全的认证，更多具体认证信息请访问：

https://trust.mi.com/compliance

小米尊重并保护所有用户的个人隐私权，通过隐私政策详细介绍小米如何收集、使用、披露、处理和保护您在使用小米产品和服务时，提供给我们或我们收集的信息。不同语言版本的隐私政策链接：

https://privacy.mi.com/all

* 注：部分产品有单独的隐私政策链接，可在对应产品页面查看。

小米拥有一支专业的安全与隐私团队，负责为小米产品的安全与隐私提供技术支持，为开发中和已发布的产品提供安全与隐私审核和测试。同时，小米通过自建的小米安全中心 (SRC)、Hackerone 及邮箱等多种途径，向全球的安全研究者征集安全问题和安全情报，并依据问题或情报级别给予不同金额的奖金。同时小米通过“小米智能生活安全守护计划”主动邀请安全研究者对小米产品进行安全测试，并给予高额奖金。 小米会将已确认的安全问题列为高优先级事项并尽快解决。

小米安全中心联系方式：

https://sec.xiaomi.com/

https://hackerone.com/xiaomi

security@xiaomi.com

5 系统内核（Kernel）

5.1 Linux 内核子系统

5.1.1 子系统简介

结合用户体验和性能需求，针对传统 linux 内核的内存管理、虚拟文件系统进行重点重构和优化（进程调度、网络接口、进程间通信等部分分别在其余对应子系统中做重点优化和展示。

5.1.2 子系统架构

图 5.1-1 Linux 内核子系统架构

5.1.3 重点特性

5.1.3.1 Hyper 内存分配管理引擎

技术介绍

Hyper 内存分配管理引擎主要负责为上层业务分配所需要的物理内存资源，以此满足业务需求。内存分配速度的快慢直接关系到上层业务的软件服务质量。小米内存分配管理引擎打破了传统的 Linux 内核无法感知上层业务的限制，能够感知上层业务整个生命周期，即使在极端场景下，也能够及时满足前台业务的内存分配请求，保证前台业务的软件服务质量。同时，针对大内存设备，Hyper 内存分配管理引擎通过大页管理技术降低系统内存管理开销，提升系统整体性能和流畅度。

技术架构

图 5.1-2 Hyper 内存分配管理引擎技术架构

关键技术

1. **核心场景内存预留：**通过识别核心场景内存需求，提前为核心场景预留一定的内存，确保核心场景内存分配请求能够得到及时满足，提升核心场景系统流畅度和稳定性。
2. **内存加速：**针对资讯类与短视频类应用，将临时文件目录通过内存文件系统挂载，减少 UFS 写入，提升 Flash 寿命。
3. **内存分片**: 改变过去单区域内存管理模式，将内存进行多区域管理，内存可以并行分配，减少内存分配延迟，降低内存碎片化概率。
4. **大页管理：**针对大内存手机，改变过去 4k 页面管理方式，降低大内存手机系统内存管理开销，提升系统流畅度。
5. **预分配：**基于用户日常使用习惯预测应用启动时机，通过预加载机制加速应用启动速度。

5.1.3.2 Hyper 内存回收交换引擎

技术介绍

随着系统长时间使用，系统可用内存不断降低。为了保证上层业务内存分配请求得到及时满足，系统需要从当前已分配内存中选择一部分内存数据进行回收，以此释放出空闲内存。Hyper 内存回收交换引擎通过冷热内存分离技术以及前台感知技术，能够有效识别出对用户体验影响较小的内存数据，通过优先回收这部分内存数据来降低系统回收对用户体验的影响。对于系统中那些长期不活跃的匿名页，Hyper 内存回收交换引擎通过内存扩展技术，将这些数据交换到 Flash，进一步提升系统可用内存大小。

技术架构

图 5.1-3 Hyper 内存回收交换引擎技术架构

关键技术

1. **前台感知回收：**感知上层应用整个生命周期，优先回收用户感知不明显应用内存数据。
2. **内存冷热数据分离：**通过对应用前台使用过程中的内存数据访问进行采样，识别应用冷热内存数据，优先对冷数据进行回收。
3. **内存扩展：**通过多次标记技术，将内存中不活跃内存数据并将其交换到 Flash 来提升系统可用内存，改善用户体验。
4. **内存冻结：**在内存、ZRAM、Flash 之间建立三级缓存，根据 TOP 应用物理内存占用，在三级缓存之间交换内存数据，提升系统后台驻留能力。

5.1.3.3 存储引擎

技术介绍

存储引擎包含 3 个存储方面的专项内容：

1. 智慧 IO 引擎 2.0，负责提高存储的性能。
2. 焕新存储，负责预防存储的老化。
3. 系统精简，负责精简系统资源对存储大小的占用。

技术架构

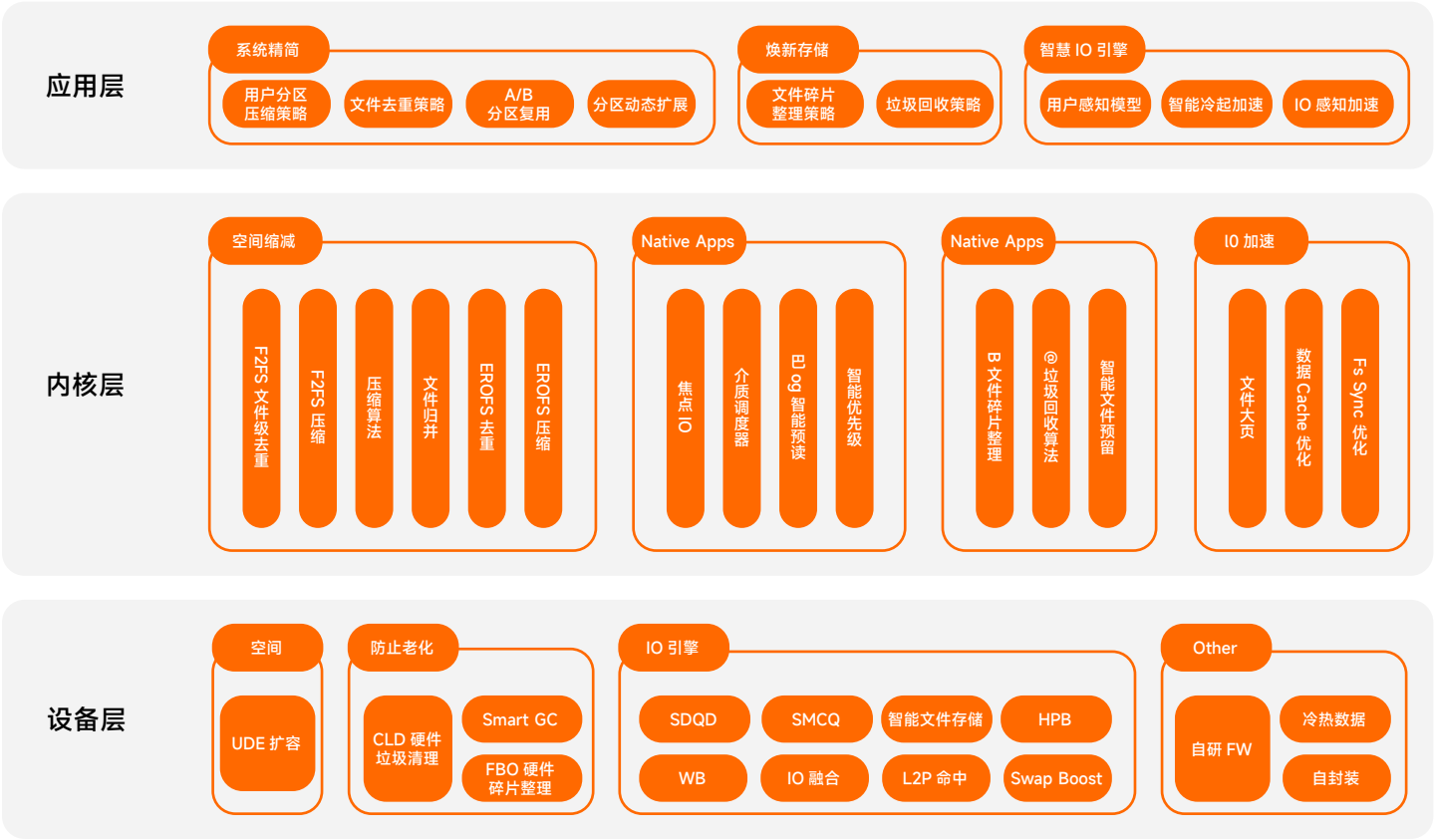


图 5.1-4 存储引擎技术架构

关键技术

1. **智慧 IO 引擎 2.0：**包含焦点 IO、智慧 IO 预加载、数据 Cache 优化三项核心能力。

- **焦点 IO**

  IO 性能是保证系统流畅的一个基础必要因素，受益于行业上游硬件的发展，硬件 IO 带宽在过去 10 年提升了 10 倍，但是我们仍然有可能会出现前台 app 操作不够流畅，或是在一些大文件加载或拷贝的场景加载过慢。这些场景的卡顿有很大概率是后台应用的 io 请求抢占了前台应用的 io 请求造成的。

  焦点 IO 技术将前台 / 后台的 io 带宽分配一个合理的比例，前台 io 优先下发，避免后台 io 对前台 io 的抢占，进而优化 app 冷启和使用过程中的流畅度。

- **智慧 IO 预加载：**通过机器学习预测需要哪些 I/O 并提前执行来减少应用程序启动时间，预取数据操作完成后，应用程序几乎可以立即从 pagecache 访问这些数据，从而显着减少应用程序启动延迟。

  工作流程：

  - 在学习阶段，即应用程序的第一次启动期间，用户模型感知模型通过监视磁盘读取或页面错误收集与启动相关的块以及它们的访问序列。
  - 后续应用程序启动期间，智能冷启加速模块使用收集的启动序列进行磁盘预取以加速加载。

- **数据 Cache 优化**

  在 f2fs 基础上加入 AI 算法的智能文件系统，通过 AI 算法，识别文件的生命周期，对使用过程中产生的应用缓存文件进行智能分类，识别出那些生命周期很短的文件，提前将其删除，避免 data 分区不必要的回写量，减少存储碎片化，延缓存储老化进程，延长 UFS 寿命（UFS 写次数是有限的）。

2. **焕新存储：**存储老化及性能改善方案，底层硬件的 FBO 换新存储整合了四个象限，成为目前世界上第一个完整的整理功能。其中，文件碎片整理提高文件读取性能，降低 APP 在手机老化情况下的启动时间；Smart GC 智能碎片整理，大负载后保证 IO 性能；GC 垃圾回收算法降低功耗，增加 UFS 寿命；智能文件预留控制运用的文件范围，降低碎片程度，提升性能；CacheSifter 降低 Flash 写操作，减缓存储碎片化，提升性能。
3. **系统精简：**系统精简主要集中在系统固件的精简以及用户可用空间的增大上，系统固件精简在新机上已做到智能手机业界（包括 Android 阵营和苹果手机）行业第一。

5.2 Vela 内核子系统

5.2.1 子系统简介

5.2.1.1 物联网操作系统技术演进史

目前，物联网操作系统没有严格的定义，按发展路径和阶段可以分为四大类：一是由传统的嵌入式 RTOS 发展而来；二是基于传统操作系统进行的“剪裁”和定制；三是专门面向物联网研发的操作系统；四是新一代统一型操作系统，通常称为跨设备分布式操作系统。这些操作系统，虽然出现的时间点各有先后，但并非代际替换的关系。一些物联网操作系统，比如 QNX，虽然出现很早，但现在还是在工业界，特别是车机领域得到大量的应用。

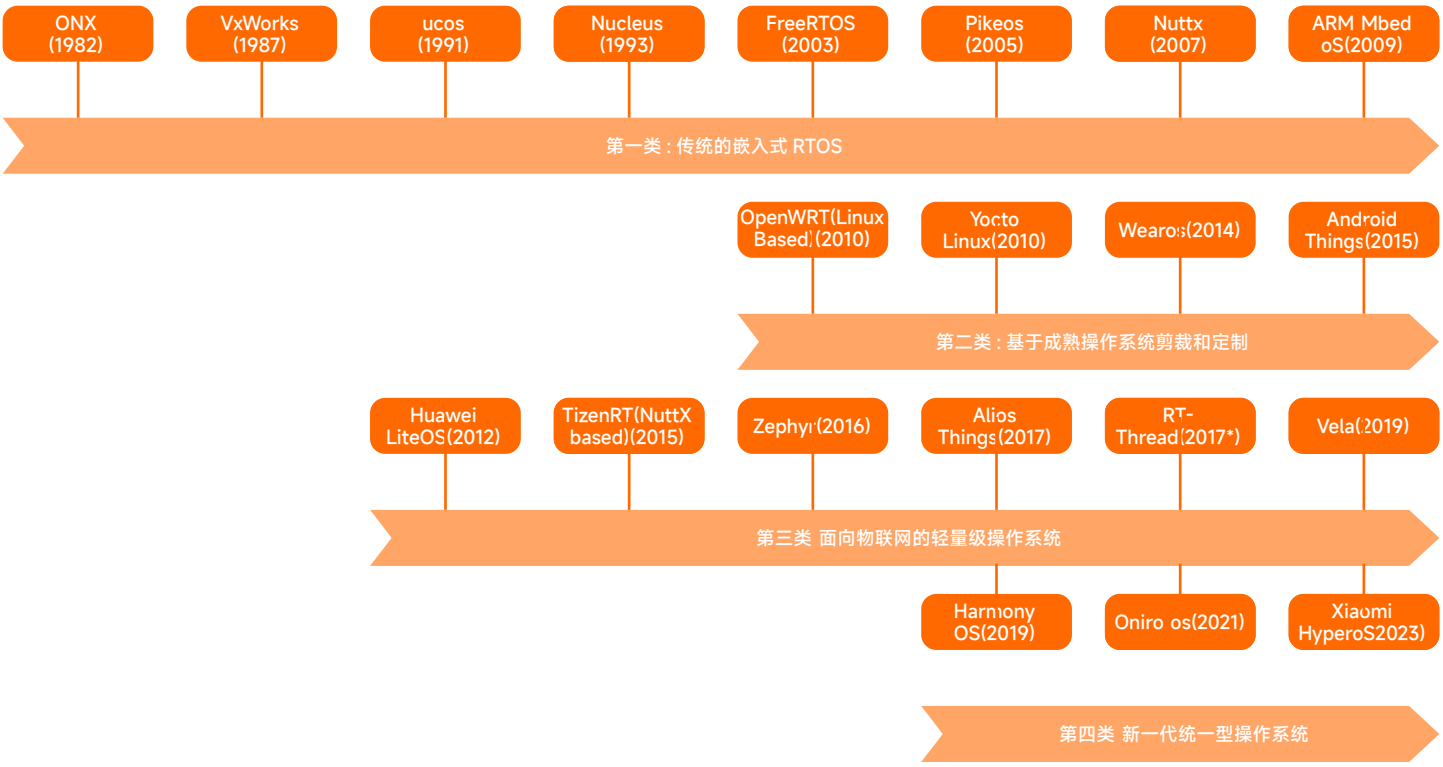


图 5.2-1 物联网技术演进史

5.2.1.2 物联网操作系统技术发展趋势

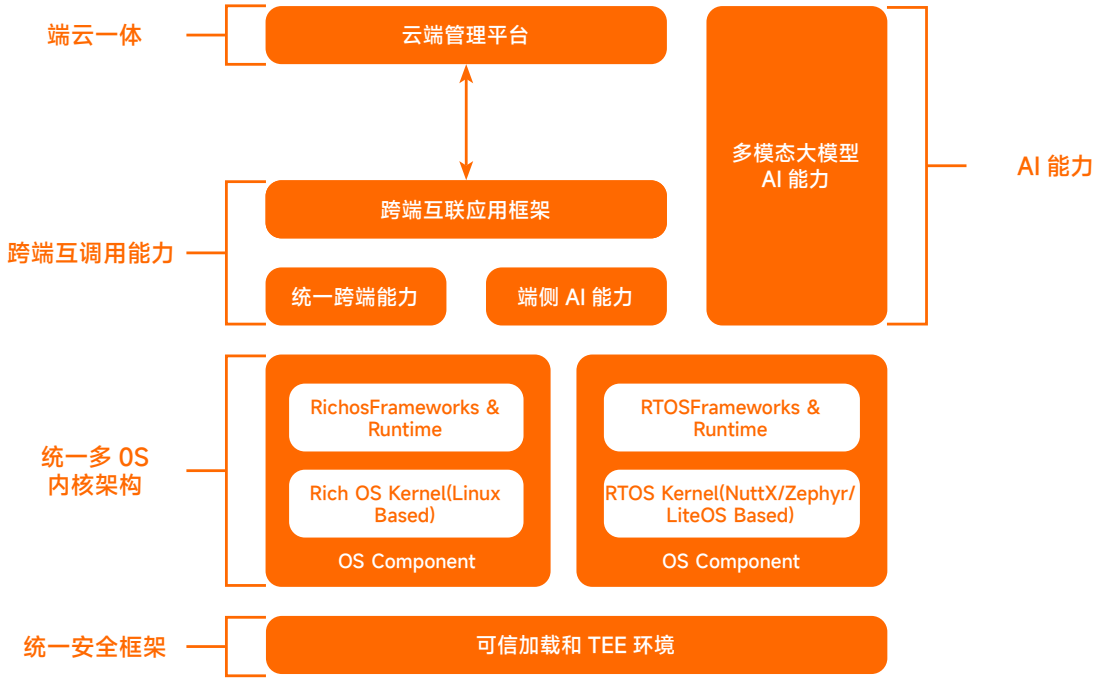


图 5.2-2 物联网技术发展趋势

- **端云一体：**随着云计算技术的不断发展，越来越多的物联网设备开始与云端进行交互。因此，物联网操作系统需要具备更好的端云一体支持，如云端管理、云端数据存储、云端分析等，而云端的数据可以结合目前的多模态大模型 AI 能力，建设 IoT 的智能化能力。
- **跨端互调用：**未来的物联网操作系统面临的是多端的环境，可能运行在不同的端侧，但具备相同的跨端能力和跨端互联框架，这样的趋势也是泛在操作系统演化的基本形态。
- **统一多 OS 内核架构：**未来的物联网 OS，会是「多内核共存」的形式，每种内核负责解决不同场景下的问题；典型的如 Open Harmony/Oniro，会整合 Linux，LiteOS，Zephyr 这几种不同的内核以及和内核密切相关的框架和运行时，内核组件化的趋势逐渐开始形成。内核组件化的本质还是为了解决物联网应用场景的扩展和不断增加的设备种类所带来的碎片化问题。
- **统一安全框架：**随着物联网设备的数量不断增加，设备的安全性问题也越来越受到关注，由于其联网的属性，所以未来安全性也是 IoT 设备的重大挑战之一。物联网操作系统作为技术基础底座，需要具备更加完善的安全机制，所以未来的主要挑战之一也就是物联网操作系统的安全性。
- **AI 能力：**下一代的 AI 能力体现在多模态大模型，这是指同时处理多种数据类型（如文本、图像、语音等）的大型深度学习模型。多模态大模型可以为操作系统的“智能化”赋能，从云侧和端侧两边同时发力，推动物联网技术的发展和应用。

5.2.1.3 Vela 的目标和定位

- **Vela 的目标**

  IoT 技术领域的核心痛点就是碎片化，其本质是市场和需求的碎片化所带来的技术组件碎片化。碎片化带来软件的低复用、低效率、和低质量。Vela 的目标是提供标准化的软件平台，高效、高质量支撑小米 AIoT 亿级设备的开发和接入，降本增效，解决 IoT 设备的碎片化问题

图 5.2-3 IoT 领域核心痛点

- **Vela 的定位**

  在 Xiaomi HyperOS 的技术体系下，Vela 定位为轻载硬件资源下，使用的轻量操作系统。所以 Vela 所覆盖的设备是所有 L0 ~ L2 类设备，同时覆盖 L3 ~ L5 设备的小核 / 小系统。

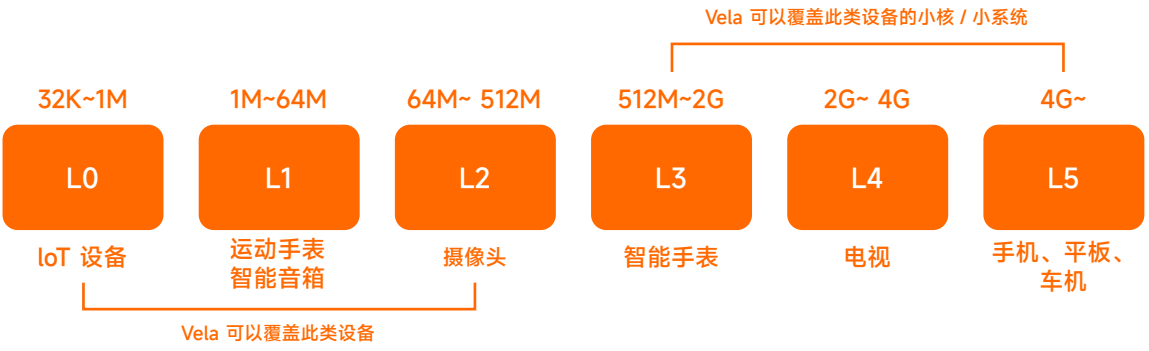


图 5.2-4 Vela 的定位

5.2.2 子系统架构

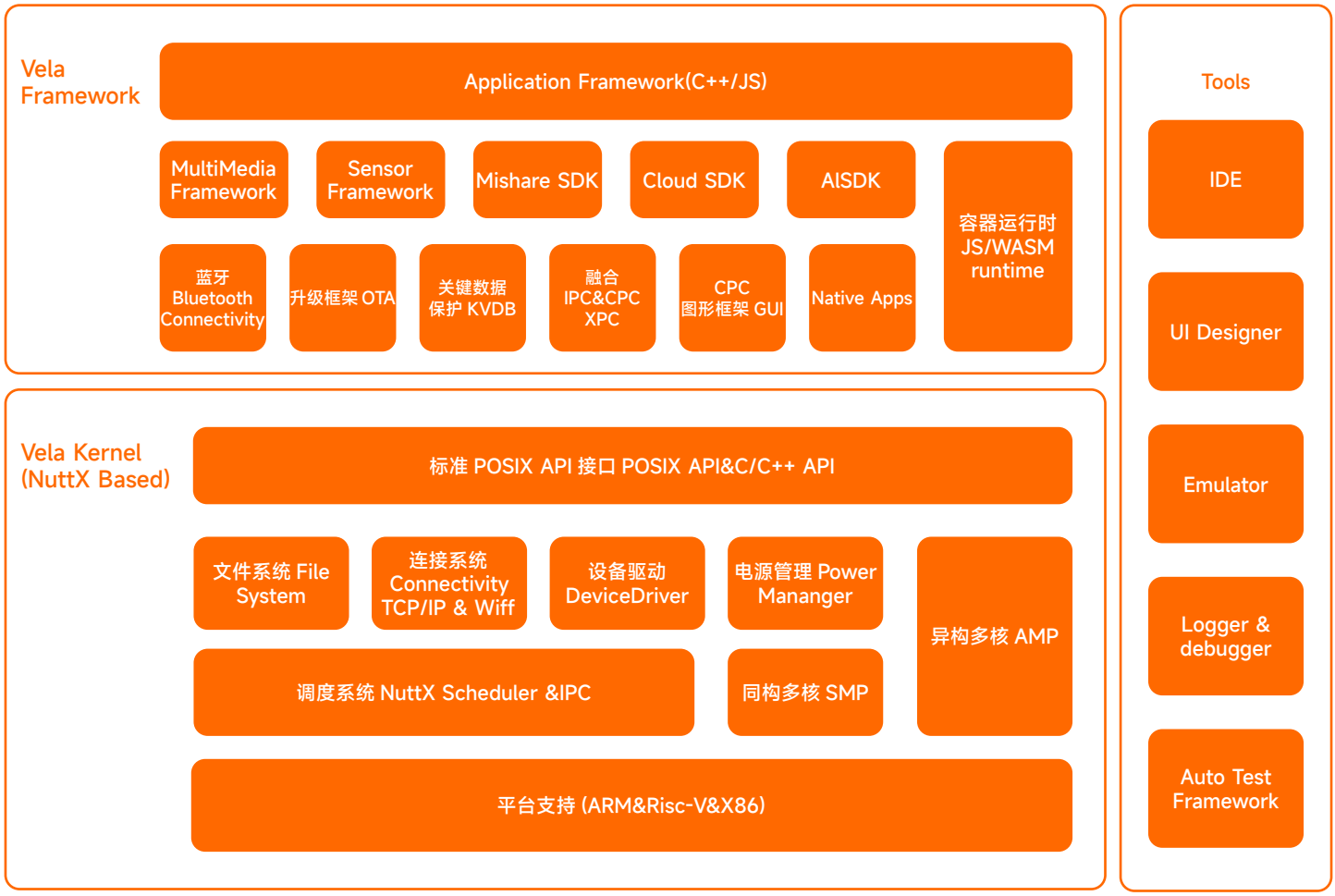


图 5.2-4 Vela 的定位

- **Vela Kernel**：提供最基本的任务调度、跨进程间通信、文件系统等基础 OS 功能，同时也提供简洁高效的设备驱动、轻量级的 TCP/IP 协议栈和电源管理等组件，也包括同构多核和异构多核的支持。
- **Vela Framwork:** 分为上下两层，下层是为扩展系统服务而提供的通用应用框架，上层是针对不同的物联网应用而开发的定制应用框架，例如多媒体应用框架和传感应用框架，提供 Cloud SDK 可以方便开发者更快速的接入小米云服务。
- **Vela Tools**：除了常见的 Logger 和 Debugger 工具，Xiaomi Vela 还提供 Emulator 工具来帮助开发者提升调试效率，使用 Emulator，开发者可以利用 PC 端丰富的调试工具和调试信息，降低嵌入式系统开发和调试的难度。

5.2.3 重点特性

5.2.3.1 异构多核框架

技术介绍

异构多核设计是低功耗融合计算平台的发展趋势，Vela 提供了标准化、高可靠、低延时、轻量级的跨核通讯机制。Vela 核间通信能力有如下一些优势：

- 对用户无感，基于标准系统调用，跨核服务均在系统底层实现。
- 对开发者友好，配置简单清晰，接口明确。
- Vela IPC 框架为分布式架构提供整体解决方案，多媒体、网络、传感器等系统核心服务可以基于该框架实现异构多核能力的融合。

技术架构

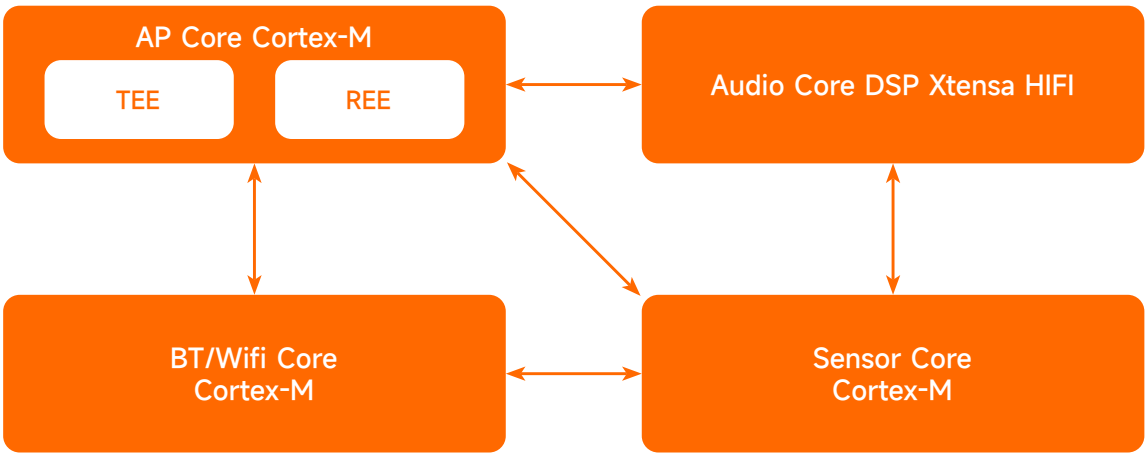


图 5.2-6 多核异构架构

- 轻量级，高可靠，低延迟。
- 功能全，覆盖 15+ 跨核服务框架（文件系统，套接字，诊断系统，虚拟串口、终端，分布式时钟树等）。
- 灵活度高，多媒体、网络、传感器等系统核心服务可以基于该框架实现异构多核能力的融合。
- 成熟性好，支持业内常见架构及 10+ 头部 SoC 平台差异化多核架构设计。

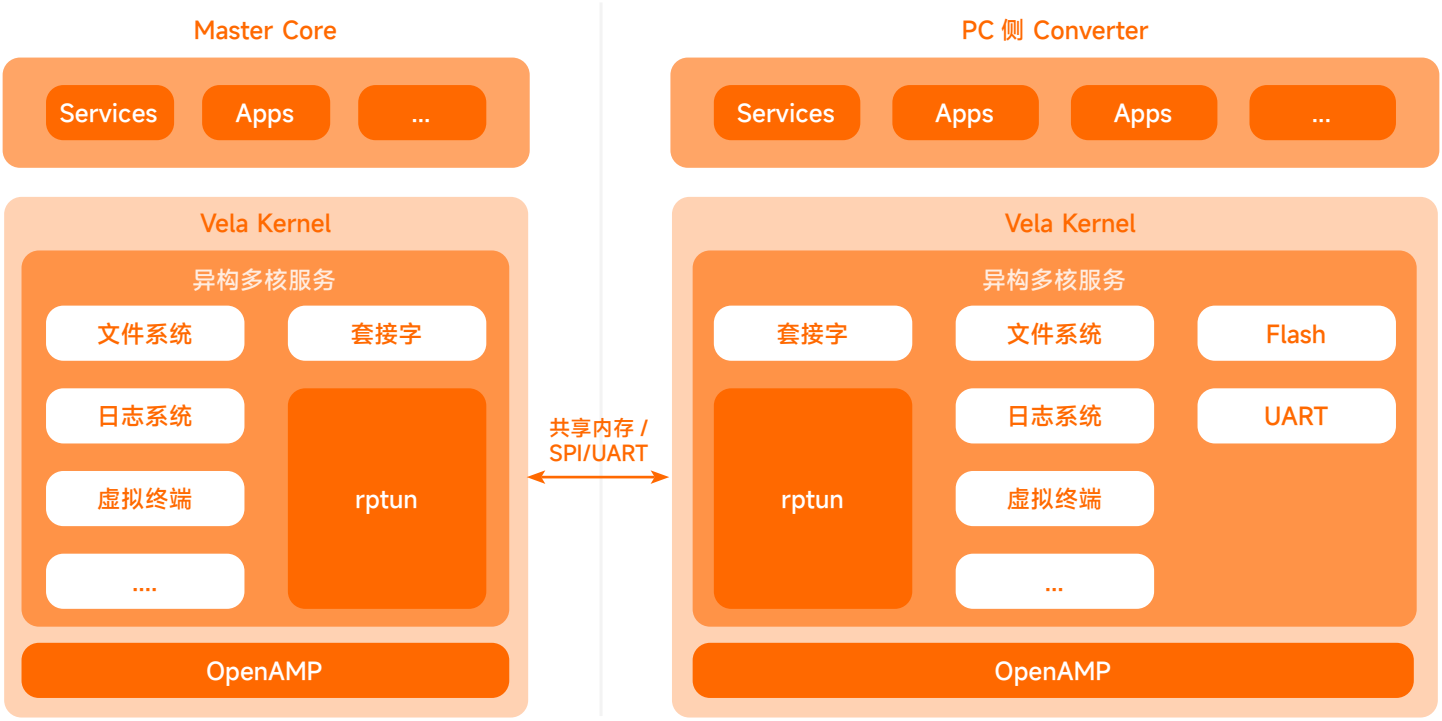


图 5.2-7 多核异构模块

关键技术

1. 异构多核文件系统服务

- 文件系统分布式多核共享。
- 对上层无感，开发者无需关心文件系统处于哪个核心，从虚拟文件系统层支持异构核文件系统访问。
- 文件、目录相关系统调用完全支持，支持动态挂载 / 卸载。

2. 异构多核虚拟串口、终端服务

- 支持异构核虚拟终端，提供多核终端动态切换功能。
- 节约前期硬件资源和布线成本，支持物理串口多路复用或单串口多路软件复用。
- 降低开发难度和诊断成本，支持传统协议工具 (adb/telnet/ssh) 对于多核的调试。

3. 异构多核网络套接字服务

- 异构多核协议族基于标准网络套接字 (socket)，完全兼容所有套接字系统调用。
- 降低开发难度和学习成本，套接字完全兼容主流多路复用和异步 IO 模型。
- 多协议族可无缝切换，一次开发即可支持本地 / 跨核 /IP 网络三种通信模型。

4. 其它异构多核服务

- **异构多核日志诊断服务：**将日志和系统诊断信息输出到其他核心。
- **异构多核硬件 IO 访问服务：**支持跨核访问硬件 IO 资源 (GPIO/I2C)。
- **异构多核网络服务：**支持跨核 IP 网络访问。
- **异构多核时钟服务：**支持多核共享 RTC 时钟源，支持订阅、发布原语。
- **异构多核电源管理服务：**多核动态电源管理框架。
- **异构多核微对象发布订阅服务：**高度融合的混杂数据订阅发布框架，特别适用于多传感器需要数据仲裁的场景（低延时、高性能、动态采样、数据融合）。

- **异构多核时钟树服务：**分布式多核时钟树可以在异构核间实现动态调节硬件时钟。
- **异构多核保活服务:** 异构多核保活服务用于监听和诊断远端核心（上下电，异常诊断），同时也可以评估跨核通讯时延。
- **异构多核远程应用调用服务：**基于多核网络套接字的跨核远程应用调用服务。
- **异构多核数据库：**分布式多核数据库实现，可在异构核间共享数据库资源。

5.2.3.2 安全技术框架

技术介绍

Xiaomi HyperOS 在 IoT 设备上使用 MiTEE，结合软硬件安全技术，在硬件层、内核层、框架层和应用层使用不同的安全技术，结合 Web Assembly 技术开发的 MCU 安全三重隔离技术，为应用提供安全保障。主要有如下的一些关键技术：

- **硬件安全和可信执行环境：**基于一体化软硬件平台，使用隔离的硬件资源结合可信执行环境构建整个系统的安全基石。
- **系统安全：**基于信任根建立的可信密钥链，提供安全启动、安全升级等功能，确保设备系统安全。
- **安全框架：**确保应用安全和应用敏感数据的安全，提供密钥库系统、加解密引擎、安全存储等服务，简化应用开发的难度。
- **应用沙箱安全隔离：**通过 Web Assembley 容器技术结合可信安全执行环境，进一步为应用提供了 MCU 安全三重隔离技术，保证 IoT 设备的安全性。

技术架构

图 5.2-8 Vela 安全技术架构

MiTEE 可信执行环境

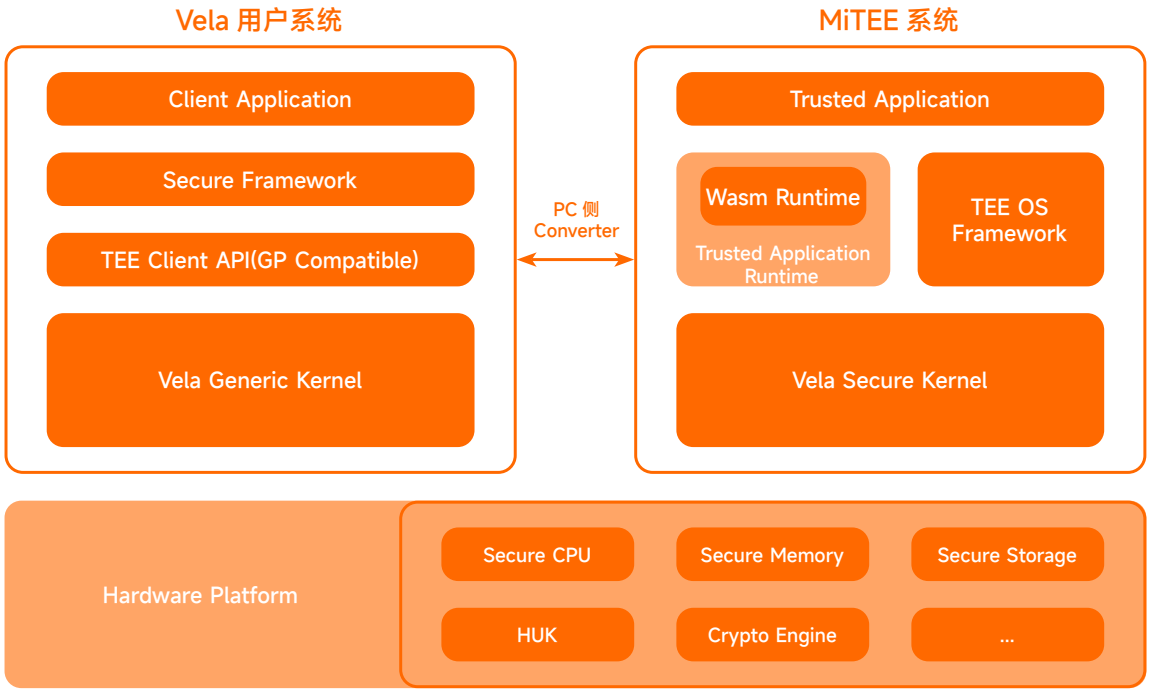


图 5.2-9 MiTEE 可信执行环境

关键技术

1. **安全启动：**安全启动是在系统启动过程中，通过公钥验证程序或数据的签名，确保启动文件或程序的完整和可信，以防止在启动过程中加载并运行了未经授权的程序或镜像。Vela 安全启动，各镜像都需要一级级校验，保证启动过程中的信任链传递，签名验签通过后才允许加载和运行。在启动过程的任何阶段，如果签名验证失败，则启动过程会被终止。
2. **安全升级：**Vela 支持通过 OTA(Over The Air) 方式更新系统软件，实现了安全、高效的系统升级管理。系统软件更新前，系统更新程序验证 OTA 下载到设备 ROM 镜像的完整性和合法性。在完成如文件大小、哈希值、签名等校验后，开始更新分区。此外，常规启动（包括升级后的首次启动）也会在 bootloader 对下一阶段要启动的分区进行验证，以杜绝启动篡改过的非法系统。
3. **密钥库系统：**Vela 轻量级密钥库系统，用于向应用程序提供密钥和证书的生命周期管理，同时为应用屏蔽不同的硬件安全环境（如 TEE 安全环境）的差异，简化应用开发的难度。包括密钥生成、安全加解密、安全存储、密钥认证等，确保整个存储和计算在安全硬件的隔离环境中。
4. **加解密引擎：**Vela 加解密引擎支持主流的加解密算法，如 AES, RSA, MD5/SHA1/SHA,X.509 证书管理和 TLS/DTLS 等协议支持，并利用硬件的能力，通过 Vela 内核的 Crypto 框架对接硬件加速，进一步提高安全性和性能。

5. **权限控制：**Vela 轻量级的权限控制，用于控制三方应用的权限，权限分为不同的类型，包括安装时权限、运行时权限。
6. **应用沙箱安全隔离：**IoT MCU 设备通常没有硬件内存管理单元，无虚拟内存，应用使用的内存无隔离环境，导致应用间很难做到内存隔离。基于 Web Assembley 容器技术，保证应用和应用间的私有内存是相互隔离的，并且接口必须要导入才能被使用来保证安全性，因此恶意应用无法冒充合法应用的身份去获取敏感数据。因此，Vela 基于 Web Assembley 容器技术结合 TEE 可信执行环境，提供了 MCU 设备三重安全隔离技术来保障安全。

- TEE 和 REE 之间隔离：提供安全隔离的环境。
- TEE 内核和用户空间的 TA 隔离：保证用户开发的 TA 不会影响 TEE 安全系统。
- 应用 TA 间，TA 和 Runtime 之间隔离：保证可信应用之间也无法相互获取对方的敏感信息。

5.2.3.3 轻量级图形渲染框架

技术介绍

Vela 低功耗图形渲染子系统主要包括 Widgets、动画、布局、输入事件、矢量绘制、字体和并行渲染等模块，满足硬件资源受限的 IoT 设备应用的开发。

各模块介绍如下：

- **UI 组件（Widgets）：**包括 Div, Image, Video, Chart, Swiper, RichText, List, Input, BarCode, Progress, TextArea 等。
- **动效引擎模块（Anim Engine）：**支持自定义属性插值动画和物理动画。
- **布局模块（Layout）：**Flex, Grid 等。
- **输入事件模块（Input Event）：**支持多种输入设备，如 Mouse, Touch, Keyboard 等。
- **2D 图形模块（2D graphics）：**包括点、线、圆、图片和文字等绘制。同时支持软件渲染和硬件加速渲染。
- **矢量绘制（Vector-Drawing）：**支持矢量图形渲染。
- **并行渲染模块（Parellel rendering）：**支持渲染任务并行处理，提高渲染效率。

技术架构

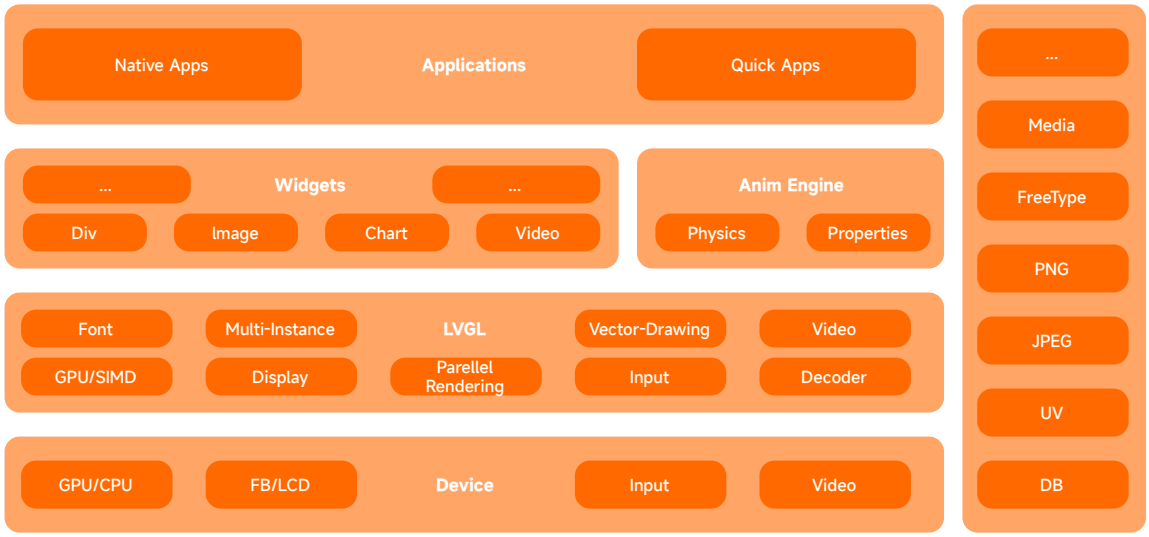


图 5.2-10 轻量图形渲染框架

关键技术

Vela 轻量级图形渲染架构，以轻量图形渲染库 LVGL 为基础，向下高度定制发挥 GPU 能力，向上提供特色组件和动效引擎，使应用的渲染更加便捷高效，同时支持一方应用和三方应用的开发。

1. Vela 图形框架兼顾多种应用需求（一方应用、三方应用），支持多种图形加速引擎，实现功耗与动画效果的最佳平衡。
2. Vela 三方应用框架，一方面利用跨端应用的已有生态，降低开发者进入门槛，二方面，支持高安全性，实现应用间的有效隔离，第三方面通过架构创新，利用前端开发的优势，有效的解决前端开发的性能瓶颈。

5.2.3.4 连接和通信子系统

技术介绍

连接和通信是 IoT 操作系统的基础必备能力，到目前为止 Vela 支持的连接和通信子系统类型如下：

- **网络子系统：**包含 80+ 的网络协议栈能力及 20+ 网络工具集，满足 IoT 领域对网络的通信和诊断需求。
- **蓝牙子系统：**实现了 GAP/A2DP/AVRCP/HID/HFP/PAN/GATT/Mesh 在内的所有 IoT 常用的蓝牙 profile。
- **Wi-Fi 子系统**：定义了统一的 Wi-Fi 驱动接口，并实现了 Wi-Fi 网络监控及自愈能力。
- **Telephony 子系统：**涵盖了网络、通话、短信、数据、modem 及卡配置管理等功能，是 IoT 操作系统中实现较为完善的无线通信软件管理框架。
- **互联互通子系统：**无缝和使用小米 IoT 模组、音响及穿戴类产品进行互联及通信。

除上述几个子系统外，还支持 NFC/UWB/CAN 等类型的通信。所有的通信子系统都提供了统一的 Framework API 给应用程序使用，方便应用程序的开发并确保了应用程序跨硬件平台的通用性。

技术架构

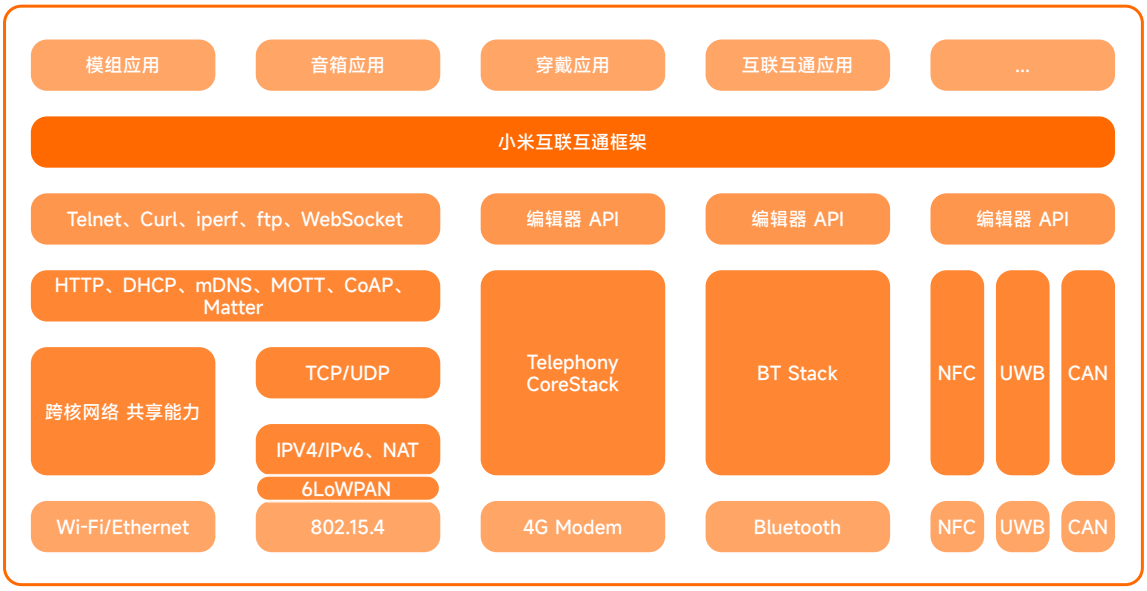


图 5.2-11 Vela 通信子系统

关键技术

除了常规的连接能力外，Vela 还提供了很多特色功能，比如：

1. **跨核应用部署能力：**在多核异构 CPU 系统架构下，通过跨核连接通信功能，让原生不具备连接能力 Core 上的应用也可以无缝的使用通用的 Framework API，方便连接类应用的跨核部署。
2. **蓝牙相关子系统跨核部署能力：**蓝牙子系统需要和其它 Audio/Telephony 模块进行通信，在多核异构的场景下，蓝牙子系统可以和其它相关子系统部署在不同的 Core 上，让产品可以在低功耗和高性能之间做到很好的平衡。

5.2.3.5 Vela 端侧 AI 推理框架

技术介绍

Vela 端侧 AI 推理框架可支撑在低算力设备上执行 AI 任务。利用统一框架优势，实现一次优化，可提升多个 Vela AI 业务（面向低算力设备，不包含大模型）。主要的框架技术包括：

- **模型转换器（Model Converter）：**
  - 模型转换器能够将训练好的 AI 模型转换为适合在 Vela 系统上运行的格式。涉及到将模型权重量化、剪枝或压缩，以满足嵌入式设备的资源限制。
- **推理引擎（Inference Engine）：**
  - 推理引擎能够在嵌入式系统上加载转换后的模型，并执行实时 AI 推理任务。
  - 采用优化技术比如 tiling、内存重用等技术以提高推理速度、内存效率和能效，适应嵌入式环境的资源限制。
  - 支持多种 AI 任务，如健康检测、语音识别或自然语言处理，提供统一的 AI 模型部署开发 API 和方案，简化 AI 业务开发流程。
  - 考虑硬件加速，如 NPU、DSP 或专用 AI 芯片等，设计多后端联合优化方案，充分发挥硬件性能，进一步提速 AI 推理任务。
- **性能检测工具（Benchmark Tool）：**
  - Benchmark Tool 提供模型分析、设备上内存占用、模型端到端推理时延检测、算子时延检测等，增强框架的鲁棒性。
- **部署和集成（Deployment and Integration）：**
  - **统一的部署开发流程**，框架对外提供统一的模型部署 API，开发者只需按照固定流程开发对应的 AI 业务，调用统一的 API 完成模型加载、算子注册、定义输入输出缓冲区、执行推理即可。
  - **封装底层异构硬件及优化技术，**开发人员只需关心推理及性能结果，使用统一的部署流程即可完成不同设备上 AI 业务的开发。
  - **文档和工具，**以便开发人员能够轻松部署和集成嵌入式 AI 推理引擎到他们的应用程序或设备中。

关键技术

框架主要分为四部分：Converter 模块，Runtime 推理引擎，Benchmark Tool，Operators Library 扩展接口。

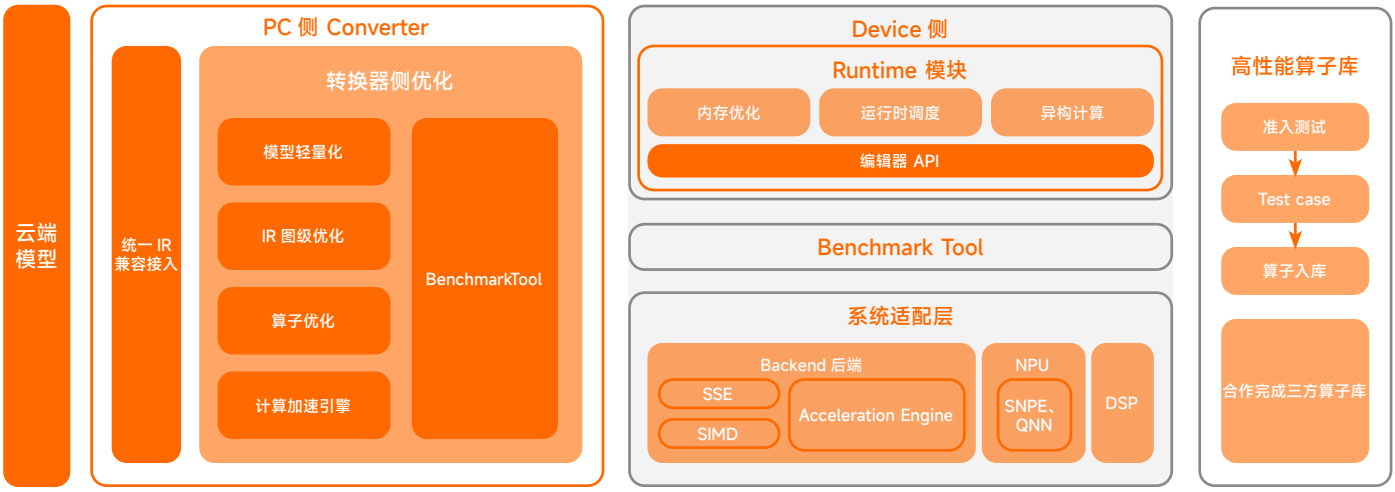


图 5.2-12 Vela 端侧 AI 推理框架

1. **Converter 模块：**转换其他框架模型为本框架可执行格式，并且对转换后的模型文件进行图优化，在保证模型精度的前提下，缩减模型尺寸，提升模型推理效率。
2. **Runtime 推理引擎：**主要包含引擎的适配及优化，开发编译优化功能，实现针对异构硬件的访存特性，生成针对性算子；使用 delegate 接口，扩展框架 NPU、DSP 后端，实现多后端推理。采用多个正交优化技术，联合优化框架的性能。
3. **Benchmark Tool：**

a. Benchmark Tool for Converter：Converter 中的 Benchmark Tool 用来检测模型转换前后的误差、推理时延等，以此评价 Converter 的转换效果。

b. Benchmark Tool for Runtime：Runtime 中的 Tool 通过检测 e2e Latency、op Latency、Accuracy 等信息，提供给框架使用和开发者来检测框架的推理性能、自定义算子或者计算库的误差等等。通过检测 e2e Latency、op Latency、Accuracy 等，提供给框架使用和其他开发者来检测框架推理性能、自定义算子或者计算库的误差等等。

4. **Operators Library 扩展接口：**对外提供算子扩展接口，其他厂商可通过定义的规范向框架中添加自定义算子，并提供 test case 做准入测试，用于测试自定义算子的规范性。

5.2.3.6 Binder XPC 框架

技术介绍

XPC 是 IPC(进程间通信)，CPC(核间通信) 和二者的统称。Vela Binder-XPC 融合架构主要包括如下的一些技术特色：

- **异构系统融合：**基于 Android 系统中普遍使用的 Binder AIDL 技术，实现跨进程，跨核的基础通信链路，构建与 Android 之间的跨核服务调用能力。
- **编程语言无关：**实现的服务程序，编程语言可能是 C/C++/JAVA/Rust/JS 等，服务的调用和具体的编程语言无关，不同语言之间可以进行互调用。
- **ABI 调用：**服务接口实现规范化，可以做到服务的 Binary 的替换和升级，并支持服务调用的权限管理。
- **系统互操作:** 兼容并符合 Android 的 Binder 标准，和支持 Binder 的操作系统（Android，Wear OS，包括 Vela ）之间可以进行服务的互调用。
- **DFD/DFT：**支持跨进程和跨核操作系统间的 DFD(Design for Debugging) 和 DFT(Design for Testing)，能进行跨操作系统的服务状态获取和记录，以及不同操作系统上的接口通过命令实现互调用。
- **可用性：**使用方法符合 Android 的标准，使熟悉 Android 服务调用的技术人员可以更快更容易地使用该接口来进行服务和应用开发。

技术架构

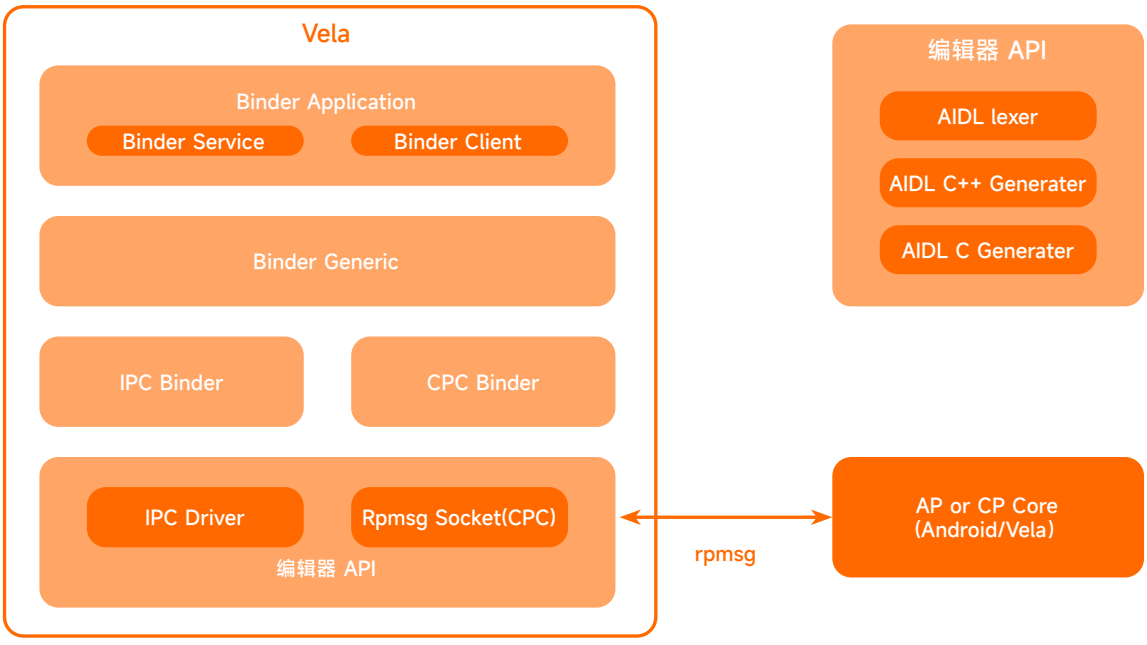


图 5.2-13 Vela Binder XPC 框架

关键技术

1. **Vela Binder 通用层：**面向 Binder 应用层的接口层，主要实现 Binder 通信中的 Bn/Bn 对象，完成客户端和服务端的通信模式，同时也包括 AIDL 语言的传输格式支持。这个层上，底层的传输层实现可以是 IPC，CPC。
2. **Vela Binder IPC:** 面向 Binder 通用层，和内核的 IPC 驱动配合实现 Vela 的进程间服务调用。应用层的实现可认为是传输层，基本传输模式兼容 Android 的 AIDL 方式。
3. **Vela Binder CPC：**面向 Binder 通用层并和 Binder IPC 对应，实现 Binder CPC 通信传输层，其底层链路采用异构多核框架中的 rpmsg socket 通信链路，支持不同 CPU 核之间的 Binder 调用和传输。
4. **AIDL Tool 工具：**基于标准的 Android AIDL 工具进行改造，用于将 AIDL 描述语言文件转换为对应的语言文件。在 Vela 中，支持的转换语言为 C++，为满足资源受限设备的需求，也提供 C 语言转换支持。

5.2.3.7 Emulator 模拟器开发平台

技术介绍

Vela 模拟器可以提供等同于实际硬件的模拟，满足开发，调试，测试和验证的需求，主要目标

- **开发者生态支持：**使 IoT 开发者可不用实际拿到 IoT 硬件就可以开始着手自己的应用开发工作，提升开发效率，开发体验更加友好。
- **工程效率提升**
  - 可启动多个模拟器同时运行不同的测试程序，更高效的进行压力测试和代码提交时的检查测试。
  - 不依赖于实际硬件，可提供更丰富便捷的调试和验证手段，显著提升复杂问题的分析和调试效率。

技术架构

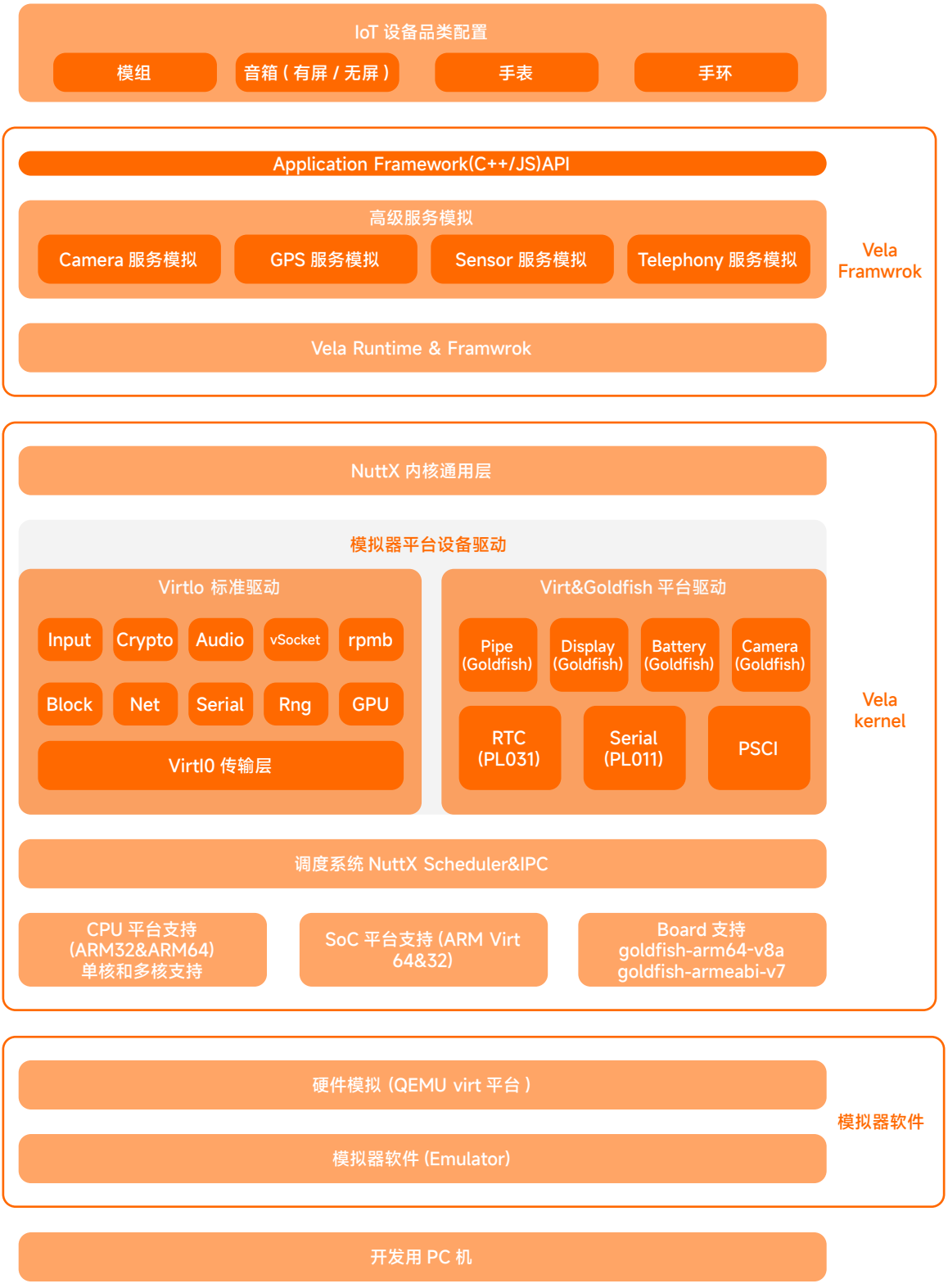


图 5.2-14 Vela 模拟器平台

关键技术

Vela 模拟器基于 Android 的 GOLDFISH 模拟器进行深度改造，其内核采用 QEMU 的 Virt 平台，支持 ARM32/64 的大部分设备的模拟，主要包括了如下的一些功能：

1. **基础平台支持：**主要包括 arm64/arm32 平台架构层的支持，支持对称多核 SMP，并在此基础上支持 QEMU 的 Virt 平台和 goldfish 的板级配置。
2. **标准 VirtIO 驱动支持：**支持大概 18 种 VirtIO 驱动，主要包括传输层，Block，Ethernet，Serial，Input，GPU，Audio，RNG 等，目前 Vela 是除 Linux 内核外对 VirtIO 驱动支持最全的 IoT 操作系统。
3. **Virt&GoldFish 平台驱动：**这些是对 QEMU Virt 和 GoldFish 平台模拟出来的非标准 VirtIO 设备的支持，主要包括 RTC（PL031），serial（PL011），PSCI(Power off)，Goldfish Pipe，Goldfish Display，Goldfish Battery 等。
4. **高级服务模拟：**基于驱动层和 Vela 的框架层，所实现的对高级功能的模拟，目前主要是 Camera，GPS，Sensor，Telephony，下一代 Vela 的模拟器将实现更多的高级功能。
5. **IoT 设备品类配置：**Vela 的主要面向 IoT 设备品类，基于模拟器已有的功能，支持了目前基本所有的 Vela 设备的模拟器配置，这些设备上的很多功能可以直接在模拟器上进行验证开发。

5.2.3.8 基于 WebAssembly 的统一应用容器

技术介绍

在 Vela 系统上提供一个统一的支持多语言支持的应用容器，总体目标：

- 支持 Vela 打造 Typescript 语言应用开发生态，开发 TS 编译器和运行时。
- 支持 UI 应用框架编译与运行，支持一方与三方开发者使用 ts 语言或者 C/C++ 语言开发 APP。
- 支持应用服务 TS 框架编译与运行，支持一方与三方开发者使用 TS 语言开发应用服务。
- 支持 Feature 新框架，可以与 JS 框架共用相同的 Feature 模块。
- 提供 TS 语言调试功能，结合自研 IDE，为开发者提供优良的开发体验。

技术架构

图 5.2-15 Vela WebAssembly 的统一应用容器

关键技术

1. 支持多种语言开发 Vela 应用或者模块

a. C/C++/Rust 语言编写的模块编译为 wasm 模块，加载到 wasm 环境中，作为独立的模块运行。

b. 与开源社区合作开发 wasmnizer-ts 项目，支持 TS 语言编译为 wasm 模块，在 Vela 系统中运行。

2. 统一的接口 API

a. 开发了 Feature 框架和自动接口语言生成工具 (JIDL)，为 Javascript, Typescript, C/C++ 语言提供统一的 Feature 接口，让不同语言开发者使用同一接口。

b. 提供 Vela 的公共模块，如 libuv, sensor, lvgl 给 webassmebly 模块，并开源到相应的项目中去。

3. 支持 Vela 多种形式的应用需求

a. 模组应用开发。

b. TEE 安全环境下应用开发。

c. Vela UI 应用 / 系统服务开发。

5.2.3.9 多媒体子系统

技术介绍

多媒体子系统在众多 IoT 设备上实现了丰富多样的多媒体功能，并且为开发者提供了简单、易用、高效的 API 接口，让开发者和用户都可以轻松使用多媒体功能。

技术架构

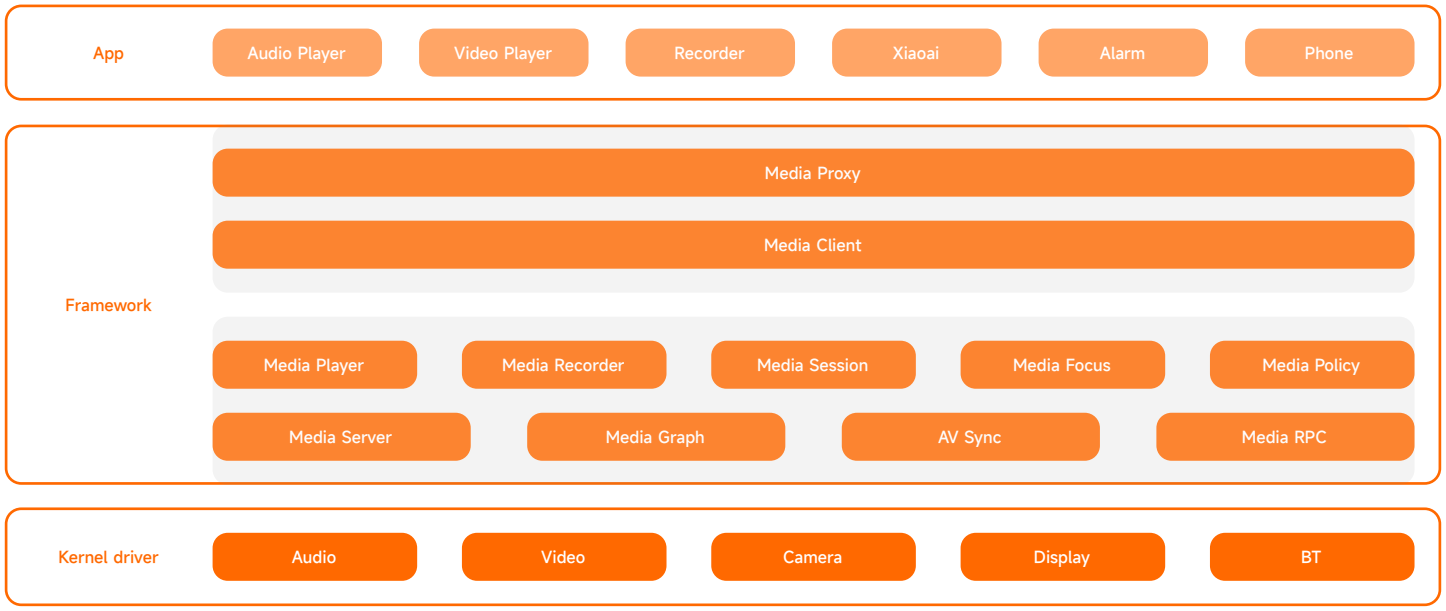


图 5.2-16 Vela 多媒体子系统

关键技术

1. **主要功能如下：**
   - 音频的播放和录制。
   - 视频的播放和录制。
   - 相机拍照和录像。
2. **架构和代码实现的主要特点如下：**
   - 组件化、低耦合。
   - 易扩展 。
   - 硬件资源占用少。

6 系统生态（Ecosystem）

6.1 开发者平台

6.1.1 子系统简介

提供标准系统与轻量系统的统一的集成开发环境，是小米澎湃 OS 一站式的应用开发平台。开发者不仅可以使用相同的工具、框架来构建跨端应用、安卓应用、轻量应用，同时还可以基于 AI 辅助编程能力极大提升开发效率，AI 辅助编程工具可以通过自动完成、代码生成、错误检测等功能，加快代码编写的速度。

6.1.2 子系统架构

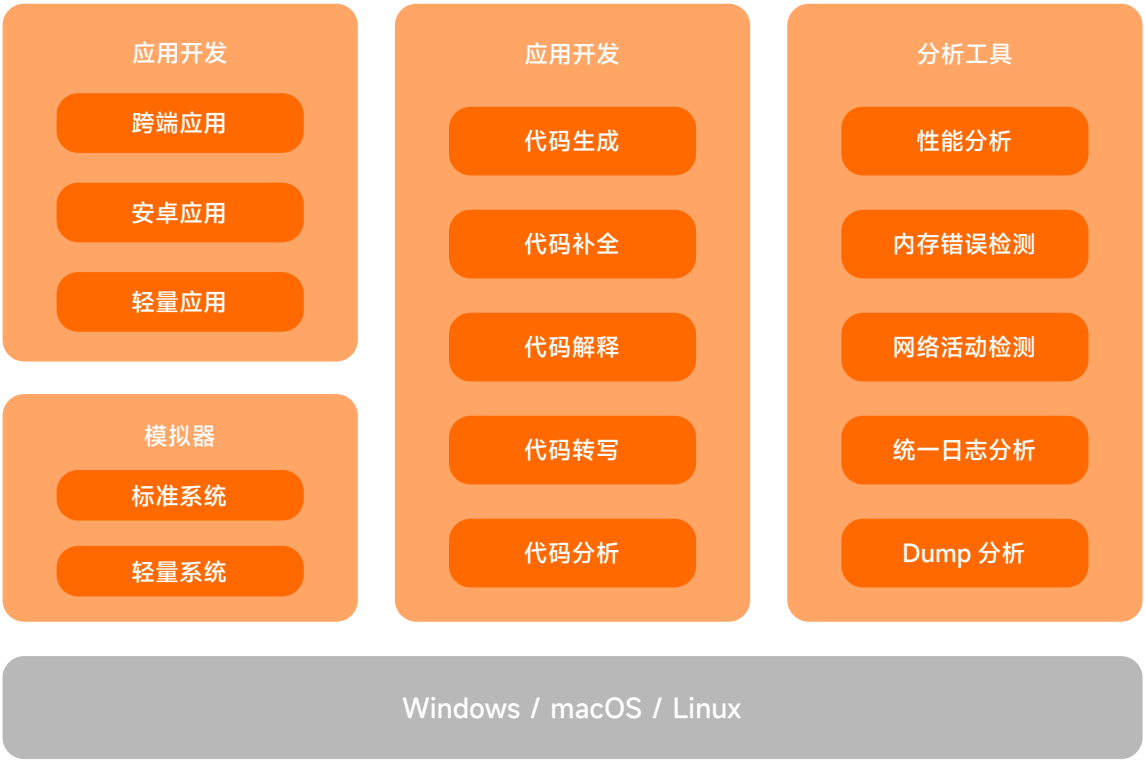


图 6.1-1 开发者平台技术架构

6.1.3 重点特性

6.1.3.1 统一开发环境

技术介绍

通过统一的应用开发环境，开发者可以使用相同的工具、框架来构建跨端应用、安卓应用、轻量应用。减少了跨端开发的学习成本，开发者可以编写通用的代码模块和功能组件，然后在不同的设备中重用，提高代码的可维护性和重用性，以及简化应用的管理和部署过程。

技术架构

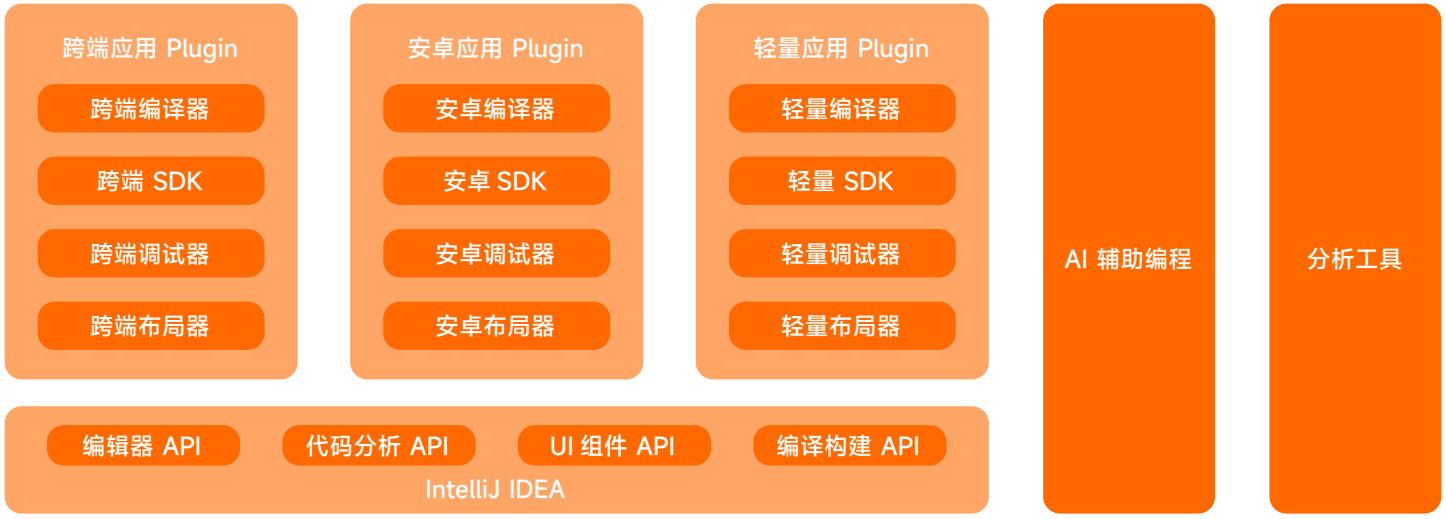


图 6.1-2 统一开发环境

关键技术

在 IntelliJ IDEA 的基础上定制和扩展，复用了 IntelliJ IDEA 的核心功能和特性，包括代码编辑器、调试器、版本控制集成等。通过集成插件的方式，为跨端应用、安卓应用、轻量应用开发提供了独特的功能和工具，例如虚拟设备管理器、布局编辑器、分析器等。

6.1.3.2 AI 辅助编程

技术介绍

基于大模型技术加入代码预训练，实现 AI 辅助编程能力，支持 Python、Java、C++/C、JavaScript、Go 等多种语言，支持代码生成、代码补全、代码解释、代码分析、代码转写等特色功能。

技术架构

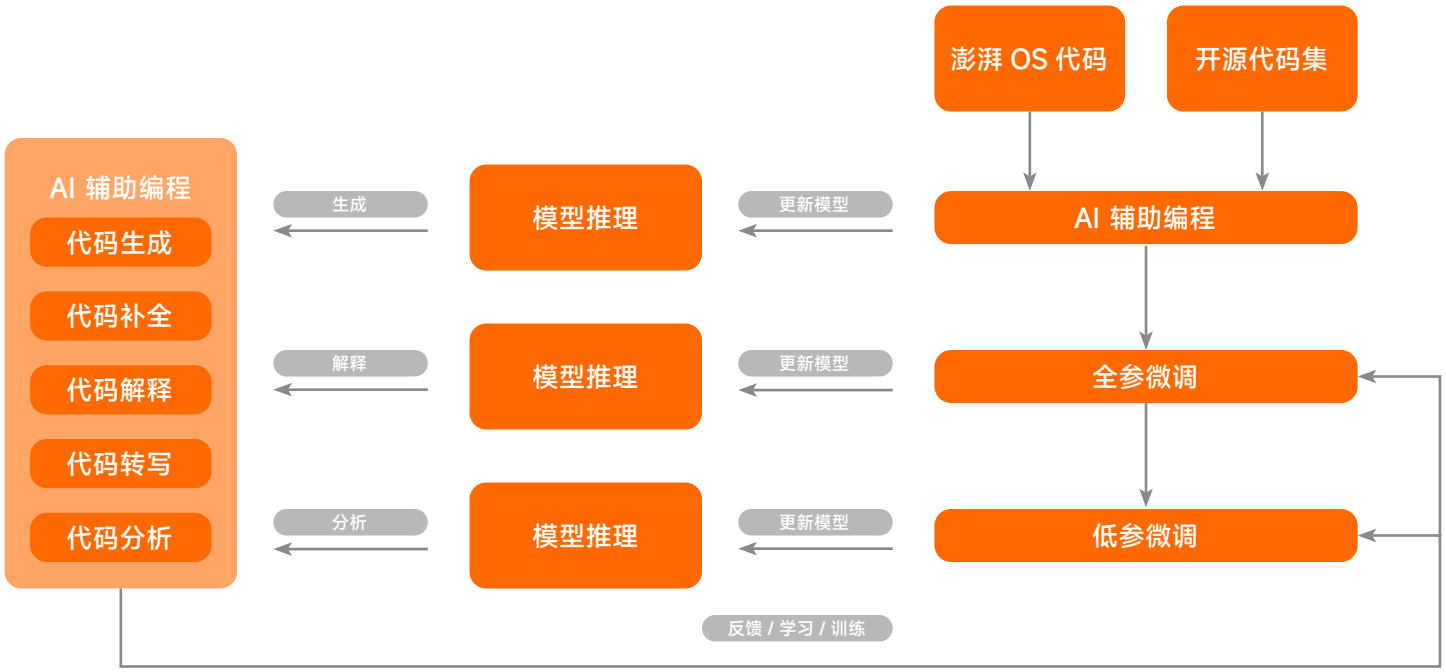


图 6.1-3 AI 编程助手

关键技术

1. **强大的代码能力：**经过了 600B 代码数据预训练，在操作系统开发辅助能力上全面领先，相比开源的编程语言在 HumanEval-X 评测集上领先优势明显，超越规模更大的 StarCoder-15B。
2. **全链路自学习体系：**打通开发者工具、推理服务、模型训练全链路体系，及时获得开发者使用数据，对 AI 辅助能力精准评估，对反馈数据进行二次低参微调、全参微调。能够不断学习和理解大量的实际代码样本，不断进化，提供更加强大的辅助能力。

6.2 分发体系

6.2.1 子系统简介

在万物互联时代人均持有设备量不断攀升，设备和场景的多样性使服务开发变得更加复杂、服务入口更加丰富。在此趋势下，应用提供方和用户迫切需要一种新的服务提供方式：使应用开发更简单，服务的获取和使用更便捷。为此，手机厂商除了支持传统方式的需要安装的应用外，还需支持提供特定功能的免安装的应用。

同时各大手机厂商均在构建自己的互联互通系统能力，并围绕以手机为中心的跨端智能分发系统。应用程序已从传统的单一手机应用逐步发展为多端异构设备下的垂直服务。服务的形态越来越趋向于免安装、独立入口和更加便捷。巨大的市场需求是各大厂商投入跨端系统和生态建设的重要因素之一。

小米的投入和生态建设已见成效，自身分发平台已经初步具备对小部件服务卡片的基本管理能力，继续在小米澎湃 OS 多端、跨端等基础能力上正在进一步建设中。流量入口场景拓宽应用正在寻求更优的解法发展。AI 智能分发能力将在 2023 年结合最新 AI 引擎和意图识别体系初步完成多场景智能推荐的建设。小米当前的分发体系主要可以分为**应用分发、服务分发**两个大的方面。

6.2.2 子系统架构

图 6.2-1 分发体系技术架构

6.2.3 重点特性

6.2.3.1 服务分发

技术介绍

物联网生态对垒已经开始，智能手机已成为万物互联的重要中心，各厂商都在构建自己的物联网生态体系和标准。应用程序已从传统的单一手机应用逐步发展为多端异构设备下的垂直服务。服务的形态越来越趋向于免安装、独立入口和更加便捷。服务的原子化和多端多场景的智能分发，以及三方生态扩展，是应用服务分发的必然趋势。

小米以澎湃 OS 操作系统为基础，服务卡片依托于 Andriod Widget 和 maml 实现，包括应用类服务、内容接口类服务、静态卡片类服务，一次开发多端支持（手机、平板、折叠屏）；赋能开发者拓展场景未来，标准化接入新型应用生态。跨端应用服务框架和开发工具正在规划和研发中；在 2023 年完成多场景智能分发的开发。

技术架构

各大厂商均在构建自己的互联互通系统能力，并围绕以手机为中心的跨端智能分发系统，小米以小米澎湃 OS 为技术基座，感知用户行为，实现小米一站式服务分发。

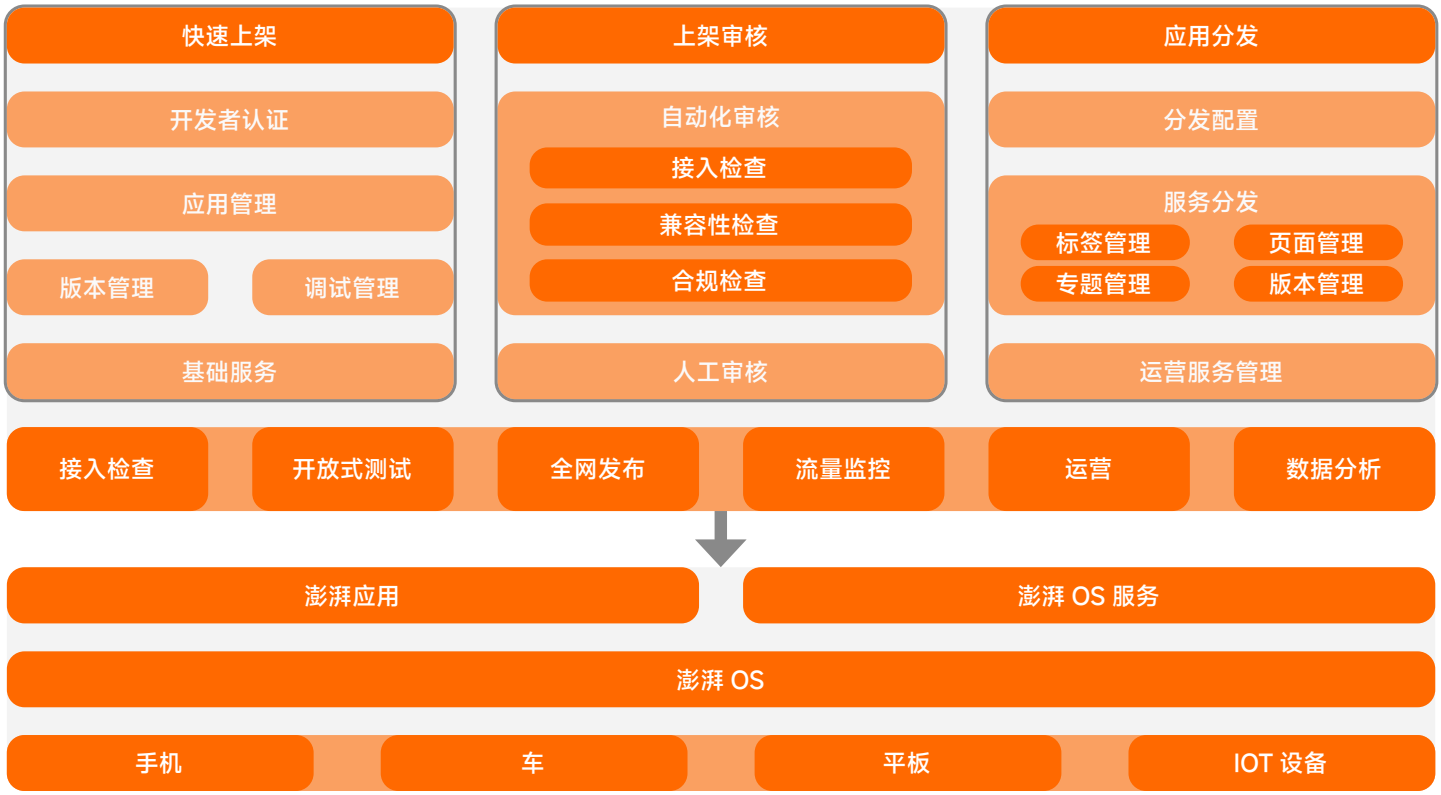


图 6.2-2 服务分发技术架构

关键技术

1. **服务直达：**服务直达项目是智能服务分发的重要尝试。以手机为载体，将核心、高频场景服务内容推荐给用户，将重要信息与场景化服务从应用中前置到桌面小部件显示，可以大大缩短用户的操作路径，提升用户获取信息的效率和体验，同时创造相应的商业化价值。目前规划中的小米服务直达能力能完成交通出行、健康管理、生活服务、情感关怀四大场景的 12 个垂类场景的智能分发能力，由传统的“人找服务”转变为智能化的“服务找人”，在提升用户体、减少冗余操作的同时还能附加人文情感关怀。
2. **服务堆叠：**为优化服务使用率，同尺寸小部件服务支持成组堆叠，节省桌面 / 负一屏空间，通过智能外露，让用户更加方便的获取所需服务。

6.2.3.2 应用分发

技术介绍

小米应用分发生态致力于打造最安全、最健康的应用生态，让小米用户享受获取应用的乐趣，助力开发者全面成长。

用户侧，依托于应用商店、游戏中心等应用分发媒体，基于系统优势，为用户提供更丰富、更便捷、更安全的应用下载及使用体验。

开发者侧，依托于小米开放平台，从应用开发、分发、增长、变现的全生命周期流程出发，基于系统优势，为开发者带来效率、流量、口碑及商业价值。

技术架构

应用分发体系整体架构包含用户层、服务入口层、能力层、基础平台层、数据层 5 大层级。

- **用户层：**包含用户、开发者两类平台用户
- **服务入口层：**支持用户侧、开发者侧多服务入口，同时提供多终端服务能力
- **能力层：**围绕应用全生命周期（应用开发、分发、增长、变现）提供了全方位的服务能力
- **基础平台层：**包含了对内、对外的基础平台，支撑上层能力的实现
- **数据层：**包含了应用分发体系的整体底层数据

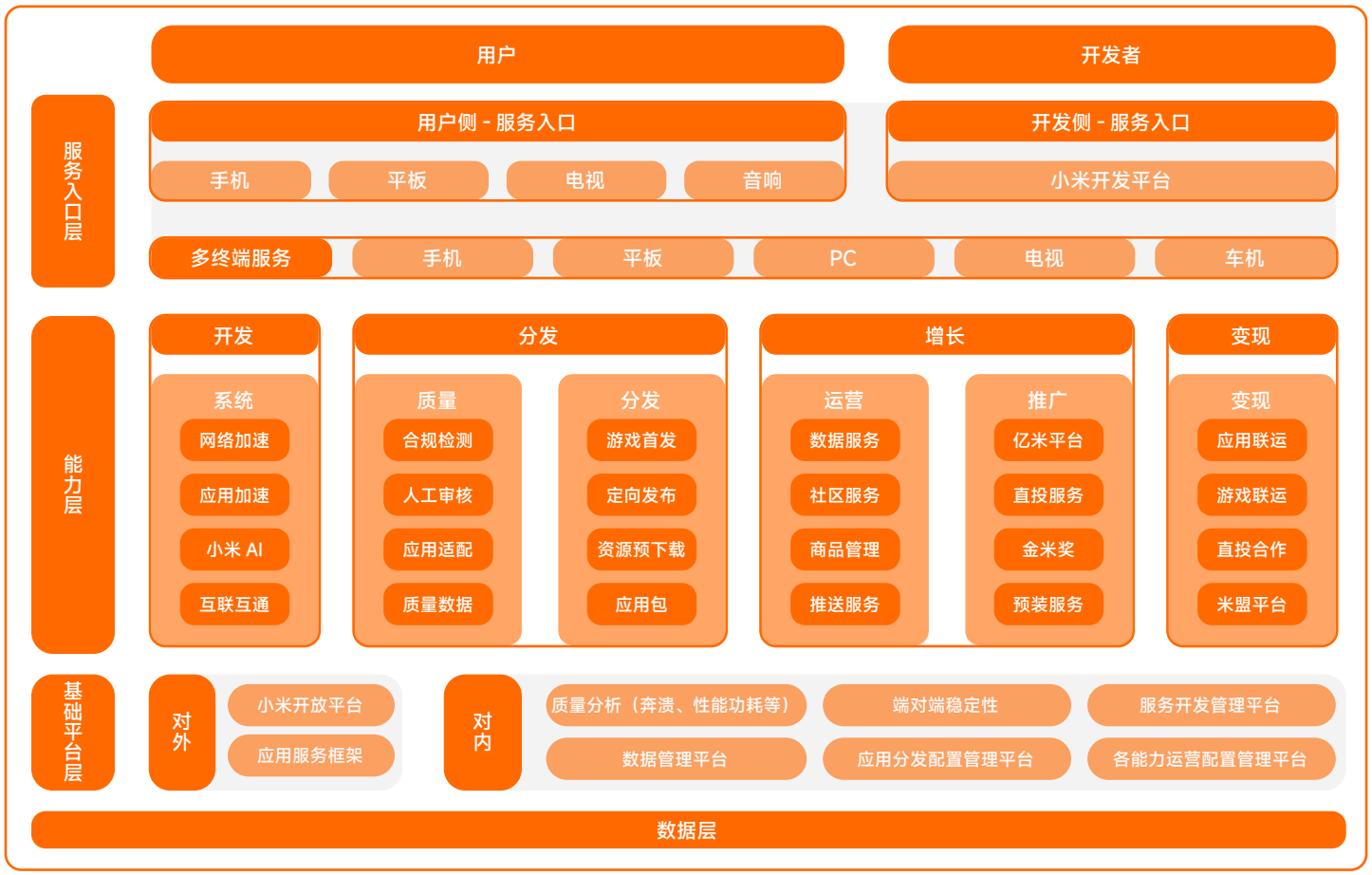


图 6.2-3 应用分发技术架构

关键技术

1. **小米服务框架：**小米服务框架（XiaoMi Service Framework 简称 XMSF）对标 GMS“应用服务 App services”建设。整合已有应用服务能力，构建小米澎湃 OS 的终端移动应用服务，面向三方开发者开放，帮助三方应用更便捷、高效地接入系统服务，为用户提供卓越的应用使用体验。

   目前，XMSF 已完成插件化服务能力的建设，同时基于插件化服务实践推送服务、应用更新服务、应用直投服务等应用服务能力的建设。
2. **极速分发服务：**应用分发链路是应用分发生态最关键的环节之一，基于小米澎湃 OS 系统优势，小米极致提升了应用在下载、安装、更新等环节的效率及用户体验，同时也面向开发者提供了应用分发服务的开放能力。

- 在应用首次分发上，支持多网并行高速下载、应用包体压缩、安装提前云端校验、Bundle 包体动态分发、游戏资源包预下载等链路提效能力。
- 在应用更新上，支持定向灰度升级、差分包升级、WLAN 下自动更新、应用内更新引导等能力。
- 分发服务开放上，面向开发者开放了直投 2.0 应用分发服务，帮助开发者提升应用分发转化率，助力开发者应用增长。

7 开源开放

近期我们会逐步开放小米澎湃 OS，这将是一个突破性的时刻，标志着我们在开放创新和生态系统建设方面迈出了重要的一步。同时我们将宣布 小米澎湃 OS 全球开放合作伙伴计划的启动，展望未来，我们将与行业上下游企业紧密合作，为用户提供更卓越的科技产品和更美好的生活。

开放、共赢的软件平台

在科技领域，开放和合作是不可或缺的核心价值观。我们坚信，只有通过共同努力和创新，我们才能不断推动科技进步，为用户创造更多的价值。小米澎湃 OS 的开源将是小米这一承诺的具体举措。

我们鼓励开发者参与到小米澎湃 OS 的开发和改进中来，共同打造一个更强大、更创新的软件平台。开源 小米澎湃 OS 的代码将提供给全球的开发者，让他们能够自由地探索、扩展和定制 小米澎湃 OS，创造出更多精彩的应用和功能。

除了开源代码，我们还将发布 小米澎湃 OS 全球开放合作伙伴计划。这个计划将为合作伙伴提供一系列支持和资源，包括技术培训、市场推广、生态合作等，以便他们更好地融入 小米澎湃 OS 的生态系统。我们欢迎各个领域的企业、创业者和开发者参与到这一计划中来，与我们一起构建一个开放、共赢的未来。

为用户带来更优秀的科技产品

在开放创新的指导下，我们的目标是为用户提供更卓越的科技产品，让他们能够体验到更多的便捷、智能和娱乐。小米澎湃 OS 是我们为实现这一目标而打造的软件平台，具有许多引人注目的特点。

1. **多样化的应用和服务**

   小米澎湃 OS 的开放性使得用户可以享受到各种多样化的应用和服务。从实用工具到娱乐应用，从智能家居到健康管理，我们致力于为用户提供一个涵盖各个领域的全方位体验。

2. **智能互联**

   小米澎湃 OS 的智能互联能力使得设备之间能够更好地协作，提供更顺畅的用户体验。无论是家庭设备的智能控制，还是汽车互联网的应用，小米澎湃 OS 都能够实现设备之间的智能互通。

3. **安全与隐私**

   我们深知用户对于安全和隐私的关切，因此，小米澎湃 OS 的设计考虑了安全性和隐私保护的重要性。我们将不断提升安全技术和隐私控制，以确保用户的数据和信息得到妥善保护。

展望未来

小米澎湃 OS 将与全球的开发者、合作伙伴和技术爱好者们共同讨论未来的发展方向。我们期待与各位分享我们的愿景，并探讨如何共同推动科技创新，为用户带来更多惊喜和便利，让我们一起构建一个开放、共赢的软件平台，为消费者带来更优秀的科技产品和更美好的生活。

8 附录

参编人员：

张国全、李凯、李昂、黄帅、肖翔、杨晓星、王楠楠、顾凌华、牛坤、徐国市、顾瑶瑶、李楠、李晓杰、刘陶、滕孝军、王乐、郑晴、胡鹏龙、张楠、高山、亢彦军、张墅潇、钟桂林、潘双全、王宝林、史润宇、孔令涛、刘才、陈雍、周鹏、秦蔚、邱峰、张元星、王福业、张雅琼、王林、张达、顾瞿、武新楠、张柳军、潘兴隆、谷超、董万强、郭红华、肖成伟、乔光军、熊平、马振华、闫昊、张卫东、汤思远、郭玉成，吴汉钊、许家铭、卢明、余俊飞、王凯、安超、张嫄、熊裕聪、陈宗强、赵中玺、王世宏、郭斌 、黎国梁、李松