第一篇　开发准备篇

第 1 章

HarmonyOS 系统简介

鸿蒙操作系统(HarmonyOS)是一款面向物联网全场景的分布式操作系统,如图 1.1 所示。鸿蒙操作系统不同于现有的 Android、iOS、Windows、Linux 等操作系统,它设计的初衷是解决在 5G 万物互联时代,各个系统间的连接问题。鸿蒙操作系统面向的是 1+8+N 的全场景设备,能够根据不同内存级别的设备进行弹性组装和适配,可实现跨硬件设备间的信息交互。

图 1.1　HarmonyOS 操作系统

“鸿蒙”名字源于华为公司内部一个研究操作系统内核的项目代号。

“鸿蒙操作系统”的英文名字 HarmonyOS,Harmony 之意为和谐,引申为世界大同、和合共生,是中华文明一直秉持的理念。“鸿蒙”有盘古开天辟地之意,“鸿蒙初辟原无性,打破顽空须悟空”,鸿蒙生态刚刚起步,需要华为、国内外企业的共同努力,需要众多“悟空”共同推动构建更加绚丽多彩的世界。华为的鸿蒙,中国的鸿蒙,必将成为世界的鸿蒙。

1.1　HarmonyOS 的设计目标

尽管 HarmonyOS 是在美国对华为公司实施制裁时从“备胎”提前转正,但是实际上华为公司在 2012 年就开始规划自有操作系统“鸿蒙”,这是华为公司面对以 5G 技术推动的产

业革命和国外技术风险做出的提前布局，HarmonyOS是华为公司专门为5G万物互联时代打造的战略性产品，创造性地通过分布式技术打造一个万物互联互通的物联网操作系统。

HarmonyOS的设计目标是为解决5G智能物联网时代操作系统严重碎片化问题，同时也是应对国外技术封锁和制裁下的自力更生，确保了华为在未来国际竞争中的商业安全和信息安全，同时为国家操作系统自主可控和信息安全提供了有力保障。

1.1.1 5G万物互联时代

以“超高网速、低延时高可靠、低功率海量连接”为特征的5G(第五代移动通信系统的简称)万物互联时代的到来，传统的面向单一设备的开源操作系统Android和闭源操作系统iOS都很难满足人们在不同场景下的需求，如图1.2所示，在产业层面亟须一款专门为5G时代定制的操作系统来推动产业的长远发展。

图1.2 5G三大应用场景(eMBB、mMTC、uRLLC)

在5G技术的大背景下，物联网、移动计算、智能家居、智能手机、可穿戴设备、智慧城市、无人驾驶汽车、智慧医疗、VR(虚拟现实技术，英文名称：Virtual Reality，缩写为VR)等被认为是受益最大的领域。目前，公认的5G技术适用的三大应用场景为mMTC(超高带宽引领下的智能物联网产业)、eMBB(超高清流媒体引领下的视频流产业)、uRLLC(需要5G高效低时延特点的产业，如车联网、自动化产业等)，如图1.2所示。

针对未来的5G技术发展，华为制定了“1+8+N的5G全场景战略”，1代表智能手机，8代表大屏、音箱、眼镜、手表、车机、耳机、平板等，围绕着关键的八类设备，周边还有合作伙伴开发的N类领域，围绕着智能家居、穿戴、办公、影音、娱乐等，华为将致力搭建一套更加完善的5G服务生态体系。

1.1.2 物联网操作系统碎片化

随着近几年智能物联网产业的高速发展，物联网领域的深层次问题亟待解决，物联网目前落地的痛点是下游应用场景与需求的高度碎片化，物联网终端异构、网络通信方式与操作系统平台多样化，对设备之间互联互通的实现造成较大挑战，操作系统的碎片化阻碍了万物

互联时代的业务创新。

HarmonyOS的定位就是万物互联时代的操作系统，创造性地通过分布式技术，以及高性能的软总线技术，将多个物理上相互分离的设备融合成一个“超级终端”。按需调用、组合不同设备的软硬件能力，为用户带来最适合其所在场景的智慧体验。即使用户切换场景，智慧体验也能跨终端迁移，无缝流转。

HarmonyOS通过软总线和分布式技术打通了不同设备之间的壁垒，让内容无缝流转。例如在出行领域，HarmonyOS可以通过手机、手表、车机的协同，优化出行体验。在等待网约车时，用户不需要频繁掏出手机查看车辆动态，车牌号、车辆位置等信息会在手表上实时同步，抬手可见。

家庭智能化产品中带IoT(物联网)功能的设备越来越多，如电冰箱、豆浆机、摄像头等，访问不同的IoT设备需要安装不同的App，基本上每个IoT设备都有一个App，导致手机上App众多，操作和查找起来都非常不方便，由于这个原因，实际中，App的安装率不到10%，而安装的App的使用率不到5%，HarmonyOS可以大大简化IoT设备的访问。

HarmonyOS碰一碰能力(OneHop Engine)通过NFC解决App跨设备接续难、设备配网难、传输难的问题，并能够和后台智能系统结合起来，进行相关操作推荐，如结合个人运动健康数据推荐合适的豆浆配方，智能冰箱推荐菜品的保存温度等。

1.1.3　下一代操作系统的发展方向

操作系统经历了60多年的发展，历经多代，如表1.1所示，从最早期的多任务操作系统，如MULTICS和UNIX，到适用于个人计算机的多处理器操作系统，如Linux和Windows，再到最近十多年广泛流行的移动操作系统，如iOS和Android，其核心技术已经非常成熟，软件复杂度也达到了上亿行代码的规模。操作系统的每一次大发展必定跟计算机硬件的发展密切相关。随着物联网时代的到来，操作系统必将迎来新的发展。

表1.1　每一代操作系统的特征

所属年代	第几代	产业环境	OS驱动力、需求	典型操作系统
20世纪60年代	第一代	大型机	多用户、多任务	MULTICS、UNIX
20世纪80年代末—20世纪90年代初	第二代	SMP硬件架构、虚拟内存	硬件架构	Windows NT、Linux、386BSD
2007年	第三代	设备、PDA、智能手机	设备通信	MbedOS、RT-Thread、LiteOS、FreeRTOS
2017年至今	第四代	各种物联网设备使用场景	大量IoT设备需要管理，分散的设备，AI算法	HarmonyOS、Fuchsia OS

在第二代(PC时代)和第三代(手机时代)，人们依赖一个单一设备实现网络连接和智能计算，但是在5G时代，连接网络和具备计算能力的终端数量呈几何基数增长，尽管PC和

手机依然是工作和生活的主力装备,但是在更多场景会有越来越多的连接和计算在更多其他设备(包括边缘设备)上完成。操作系统所管理的设备的概念外延就扩展了。以往的操作系统通常对单一设备进行管理,但是未来的操作系统需要对处于连接状态的分布式多终端进行统一管理。

因此,HarmonyOS 并非移动智能操作系统,而是面向未来全场景的分布式操作系统。

从技术角度尤其是设计理念来看,HarmonyOS 和 Android 有本质区别,虽然都是基于 Linux 内核,但是 HarmonyOS 采用多内核设计,同时基于分布式架构和组件化设计,能够实现弹性部署(不同设备选取原生操作系统的不同组件进行拼装)、同时支持实时(无人驾驶车机)和分时(生活娱乐)、虚拟化快速连接(不同终端从底层 OS 已被联通)。

HarmonyOS 是一款"面向未来"、面向全场景(移动办公、运动健康、社交通信、媒体娱乐等)的分布式操作系统。在传统的单设备系统能力的基础上,HarmonyOS 提出了基于同一套系统能力、适配多种终端形态的分布式理念,能够支持手机、平板、智能穿戴、智慧屏、车机等多种终端设备。

对消费者而言,HarmonyOS 能够将生活场景中的各类终端进行能力整合,可以实现不同的终端设备之间的快速连接、能力互助、资源共享,匹配合适的设备、提供流畅的全场景体验等。

对应用开发者而言,HarmonyOS 采用了多种分布式技术,使应用程序的开发实现与不同终端设备的形态差异无关。这能够让开发者聚焦于上层业务逻辑,以便更加便捷、高效地开发应用。

对设备开发者而言,HarmonyOS 采用了组件化的设计方案,可以根据设备的资源能力和业务特征进行灵活裁剪,满足不同形态的终端设备对于操作系统的要求。

1.2 HarmonyOS 技术特性

HarmonyOS 具有三大技术特性:硬件互助,资源共享(分布式架构)、一次开发,多端部署(一套代码适配各种终端)、统一 OS,弹性部署(系统可裁剪)。

1.2.1 分布式架构

硬件互助,资源共享:基于分布式软总线技术,如图 1.3 所示,结合分布式设备虚拟化平台实现不同设备的资源融合、设备管理、数据处理,使多种设备共同形成一个超级虚拟终端。

任务自动匹配后执行于不同硬件,从而让任务能够连续地在不同设备间流转,充分发挥不同设备的资源优势。分布式数据管理基于分布式软总线的能力,实现应用程序数据和用户数据的分布式管理。用户数据不再与单一物理设备绑定,业务逻辑与数据存储分离,应用跨设备运行时数据无缝衔接,为打造一致、流畅的用户体验创造了基础条件。

图 1.3　HarmonyOS 分布式架构

1.2.2　操作系统可裁剪

统一 OS，弹性部署，如图 1.4 所示，HarmonyOS 通过组件化和小型化等设计方法，支持多种终端设备按需弹性部署，能够适配不同类别的硬件资源和功能需求。支撑通过编译链关系去自动生成组件化的依赖关系，形成组件树依赖图，支撑产品系统的便捷开发，从而降低硬件设备的开发门槛。

HarmonyOS 支持多种组件配置方案，实现了组件可选、组件内功能集可选、组件间依赖关系可关联。

图 1.4　HarmonyOS 系统可裁剪

1.2.3　一套代码多端运行

一次开发、多端部署，如图 1.5 所示，HarmonyOS 提供了用户程序框架、Ability 框架及 UI 框架，支持在应用开发过程中对多终端的业务逻辑和界面逻辑进行复用，能够实现应用

的一次开发、多端部署，提升了跨设备应用的开发效率。

图 1.5　HarmonyOS 一套代码多端运行

1.3　HarmonyOS 技术架构

HarmonyOS 整体的分层结构自下而上依次为内核层、系统服务层、应用框架层、应用层。HarmonyOS 基于多内核设计，系统功能按照“系统→子系统→功能/模块”逐级展开，在多设备部署场景下，各功能模块组织符合“抽屉式”设计，即功能模块采用 AOP(面向切面编程)的设计思想，可根据实际需求裁剪某些非必要的子系统或功能/模块，如图 1.6 所示。

HarmonyOS 实现了模块化耦合，对应不同设备可实现弹性部署，使其可以方便、智能地适配 GB、MB、KB 等由低到高的不同内存规模设备，可以便捷地在诸如手机、智慧屏、车机、穿戴设备等 IoT 设备间实现数据的流转与迁移，同时兼具了小程序的按需使用，过期自动清理的突出优点。

1.3.1　内核层

内核层基于 Linux 系统设计，主要包括内核子系统和驱动子系统。

内核子系统：HarmonyOS 采用多内核设计，支持针对不同资源受限设备选用适合的 OS 内核。KAL(Kernel Abstract Layer，内核抽象层)通过屏蔽多内核差异，对上层提供基础的内核能力，包括进程/线程管理、内存管理、文件系统、网络管理和外设管理等。

驱动子系统：包括 HarmonyOS 驱动框架(HDF)，HarmonyOS 驱动框架是 HarmonyOS 硬件生态开放的基础，提供了统一的外设访问能力和驱动开发、管理框架。

1.3.2　系统服务层

系统服务层是 HarmonyOS 的核心能力集合，通过框架层对应用程序提供服务。该层包含以下几个部分。

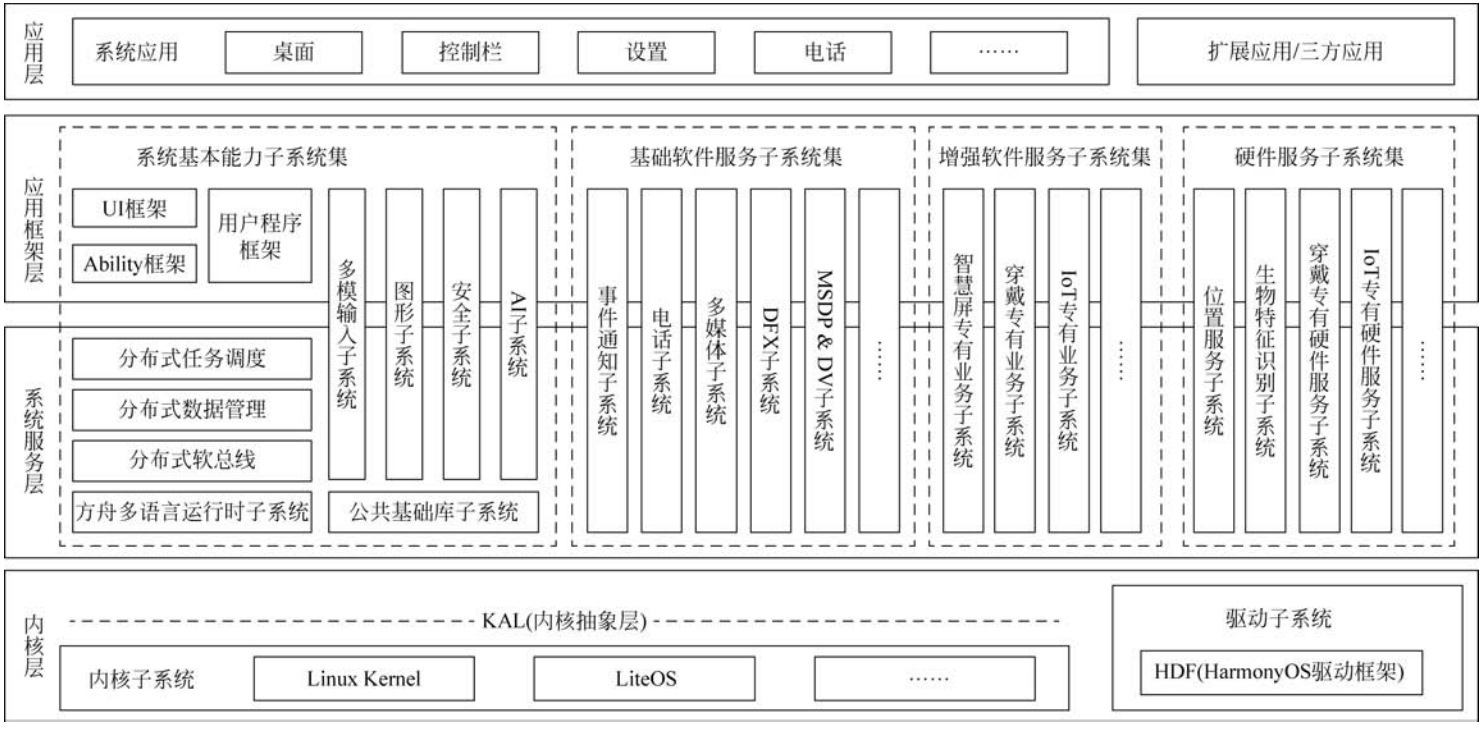

图 1.6　华为 HarmonyOS 发展历程

(1) 系统基本能力子系统集：为分布式应用在 HarmonyOS 多设备上的运行、调度、迁移等操作提供了基础能力，由分布式软总线、分布式数据管理、分布式任务调度、方舟多语言运行时、公共基础库、多模输入、图形、安全、AI 等子系统组成。其中，方舟多语言运行时提供了 C/C++/JavaScript 多语言运行时和基础的系统类库，也为使用自研的方舟编译器静态化的 Java 程序(应用程序或框架层中使用 Java 语言开发的部分)提供运行时。

(2) 基础软件服务子系统集：为 HarmonyOS 提供了公共的、通用的软件服务，由事件通知、电话、多媒体、DFX、MSDP&DV 等子系统组成。

(3) 增强软件服务子系统集：为 HarmonyOS 提供了针对不同设备的、差异化的能力增强型软件服务，由智慧屏专有业务、穿戴专有业务、IoT 专有业务等子系统组成。

(4) 硬件服务子系统集：为 HarmonyOS 提供了硬件服务，由位置服务、生物特征识别、穿戴专有硬件服务、IoT 专有硬件服务等子系统组成。

根据不同设备形态的部署环境，基础软件服务子系统集、增强软件服务子系统集、硬件服务子系统集内部可以按子系统粒度裁剪，每个子系统内部又可以按功能粒度裁剪。

1.3.3 架构层

框架层为 HarmonyOS 的应用程序提供了 Java/C/C++/JavaScript 等多语言的用户程序框架和 Ability 框架，以及各种软硬件服务对外开放的多语言框架 API；同时为采用 HarmonyOS 的设备提供了 C/C++/JavaScript 等多语言的框架 API，但不同设备支持的 API 与系统的组件化裁剪程度相关。

1.3.4 应用层

应用层包括系统应用和第三方非系统应用。HarmonyOS 的应用由一个或多个 FA(Feature Ability)或 PA(Particle Ability)组成。其中，FA 有 UI 界面，提供与用户交互的能力，而 PA 无 UI 界面，提供后台运行任务的能力及统一的数据访问抽象。基于 FA/PA 开发的应用，能够实现特定的业务功能，支持跨设备调度与分发，为用户提供一致、高效的应用体验。

1.4 HarmonyOS 与 LiteOS

Huawei LiteOS 目前作为 HarmonyOS 内核的一部分，HarmonyOS 可以根据不同的 ROM 大小选择使用不同的内核版本。Huawei LiteOS 始于 2012 年，是为支持华为终端产品而开发的嵌入式操作系统，后来在华为 Mate 系列、P 系列、荣耀系列手机和可穿戴产品上批量应用。2016 年 9 月华为正式发布 LiteOS 开源版本。

Huawei LiteOS 是华为面向 IoT 领域的开源的物联网实时操作系统，如图 1.7 所示，支持 ARM(Advanced RISC Machine，进阶精简指令集机器)、RISC-V(RISC-V 是加州大学伯克利分校开发的一种特定指令集架构，严格地说，并不是一种全新的架构，它与 ARM 同属

RISC(Reduced Instruction Set Computer,精简指令集)等主流的CPU架构,遵循BSD-3开源许可协议,具备轻量级(最小内核大小仅为6KB)、低功耗、互联互通、组件丰富、快速开发等能力,可广泛应用于智能家居、个人穿戴、车联网、城市公共服务、制造业等领域,为开发者提供"一站式"完整的软件平台,有效降低开发门槛、缩短开发周期。

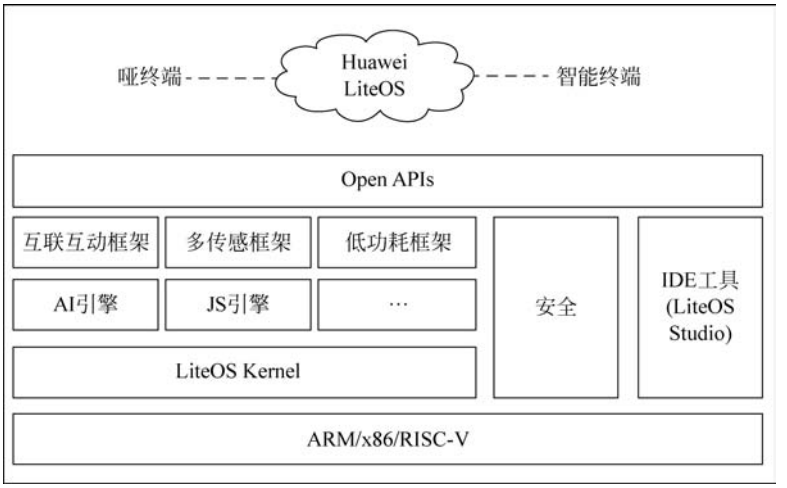

图1.7　华为LiteOS系统架构

Huawei LiteOS支持多种芯片架构,如表1.2所示,如Cortex-M series、Cortex-A series等,可以快速移植到多种硬件平台。Huawei LiteOS也支持UP(单核)与SMP(多核)模式,即支持在单核或者多核的环境上运行。

表1.2　Huawei LiteOS支持的架构

架　构	系　列
ARM	Cortex-M0
	Cortex-M0+
	Cortex-M3
	Cortex-M4
	Cortex-M7
	Cortex-A7
	Cortex-A9
ARM64	Cortex-A53
RISC-V	RV32
C-SKY	CK802

说明:ARM公司在经典处理器ARM 11以后的产品改用Cortex命名,并分成A、R和M三类,旨在为各种不同的市场提供服务。

Cortex 系列属于 ARM v7 架构,这是到 2010 年为止 ARM 公司最新的指令集架构。(2011 年,ARM v8 架构在 TechCon 上推出)ARM v7 架构定义了三大分工明确的系列:A 系列面向尖端的基于虚拟内存的操作系统和用户应用;R 系列针对实时系统;M 系列对微控制器。

LiteOS 既可以作为一款 RTOS(Real Time Operating System,实时操作系统)运行在资源受限的 MCU(Microcontroller Unit,微控制单元,又称单片微型计算机 Single Chip Microcomputer 或者单片机)上,也可以作为 HarmonyOS 的子内核运行在资源丰富的 SoC(System on Chip 的缩写,称为系统级芯片)平台上。根据硬件的资源情况,LiteOS 又可以分为 LiteOS-A(内存≥1MB)和 LiteOS-M(内存≥128KB)。

1.4.1 LiteOS-A 简介

LiteOS-A 内核是基于 Huawei LiteOS 内核演进发展的新一代内核,是面向 IoT 领域构建的轻量级物联网操作系统。新增了丰富的内核机制、更加全面的 POSIX 标准接口及统一驱动框架 HDF(Harmony Driver Foundation)等,为设备厂商提供了更统一的接入方式,为 HarmonyOS 的应用开发者提供了更友好的开发体验。

LiteOS-A 内核架构如图 1.8 所示。

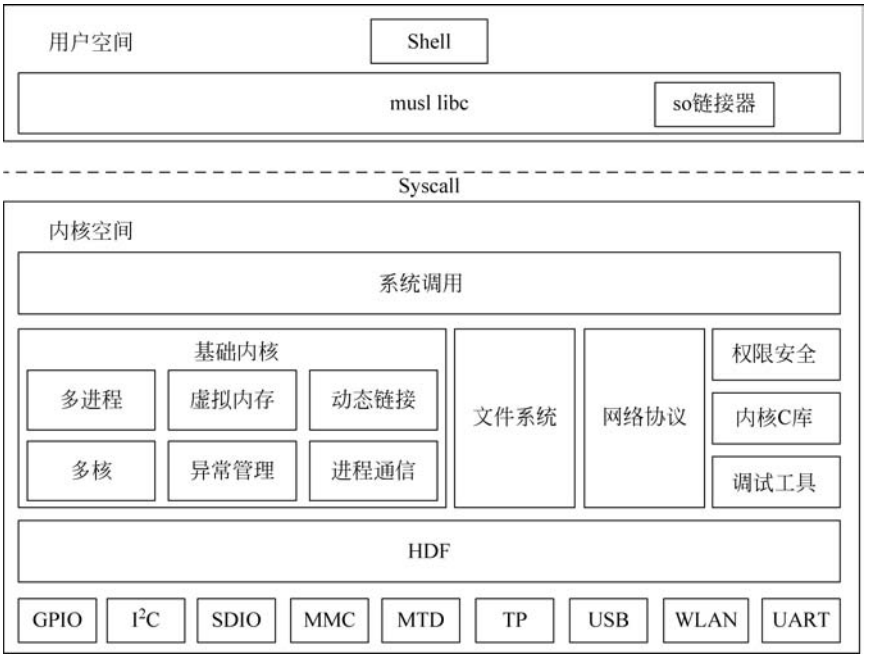

图 1.8 Huawei LiteOS-A 内核架构图

1.4.2 LiteOS-M 简介

LiteOS-M 内核是面向 IoT 领域构建的轻量级物联网操作系统内核,具有小体积、低功

耗、高性能的特点，其代码结构简单，主要包括内核最小功能集、内核抽象层、可选组件及工程目录等，分为硬件相关层及硬件无关层，硬件相关层提供统一的 HAL（Hardware Abstraction Layer）接口，提升硬件易适配性，以及不同编译工具链和芯片架构的组合分类，以此满足 AIoT 类型丰富的硬件和编译工具链的拓展。

LiteOS-M 内核架构如图 1.9 所示。

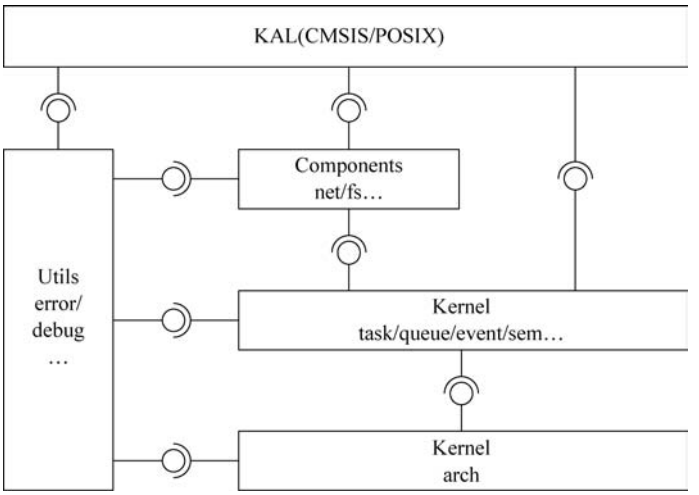

图 1.9 华为 LiteOS-M 内核架构图

1.5 OpenHarmony 生态

OpenHarmony 是 HarmonyOS 的社区开源版本，与谷歌公司的 AOSP（Android 开源项目）的作用是一致的。

OpenHarmony 项目由开放原子开源基金会负责社区化开源运营，HarmonyOS 是华为公司基于 OpenHarmony 开源版定制开发的商用发行版。开放原子开源基金会成立的目的是支持更多企业基于 OpenHarmony 定制开发自己的商用发行版本。

1.5.1 Android 与 AOSP

目前手机中运行的“安卓系统”通常是指 AOSP（Android Open Source Project，Android 开源项目）＋ GMS（谷歌服务框架，商用收费）。这两部分构成了安卓开发者使用的基础 SDK，也是所有安卓 App 的基础，如图 1.10 所示。

1.5.2 HarmonyOS 与 OpenHarmony

HarmonyOS 实际上由 3 个部分组成：OpenHarmony（开源鸿蒙操作系统）、HMS（华为移动服务）在内的闭源应用与服务，以及其他开放源代码，如图 1.11 所示。

图 1.10　Android 操作系统体系

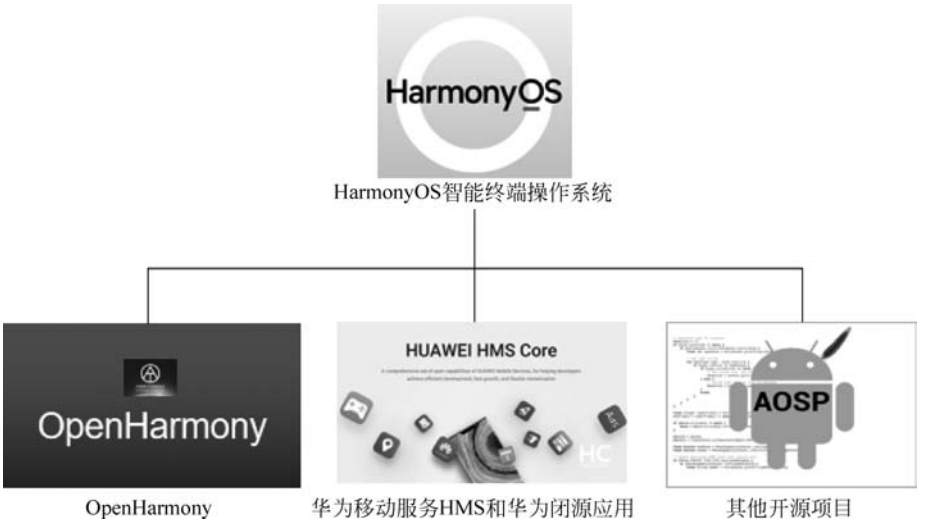

图 1.11　华为鸿蒙操作系统体系

华为手机操作系统包括 OpenHarmony(开源免费)＋HMS(华为移动服务,商用收费),HMS 是对标谷歌 GMS 的商业产品,用于支持开发者为华为手机开发 App。为了兼容现有的 Android 生态,HMS 的许多接口设计尽量兼容了 GMS。

开放原子开源基金会是在民政部注册的致力于开源产业公益事业的非营利性独立法人机构。开放原子开源基金会的服务范围包括开源软件、开源硬件、开源芯片及开源内容等,为各类开源项目提供中立的知识产权托管,保证项目的持续发展不受第三方影响,通过开放治理寻求更丰富的社区资源的支持与帮助,包括募集并管理资金,提供法律、财务等专业支持。

2020 年 9 月 10 日华为将 HarmonyOS 2.0 源码捐赠给开放原子开源基金会孵化,并为其命名为 OpenHarmony 1.0,通过 Gitee 托管代码并对外开放下载。OpenHarmony 开源项目主要遵循 Apache 2.0 等商业友好的开源协议,所有企业、机构与个人均可基于 OpenHarmony 开源代码开发自己的商业发行版,HarmonyOS 与 OpenHarmony 的发展路标如图 1.12 所示。

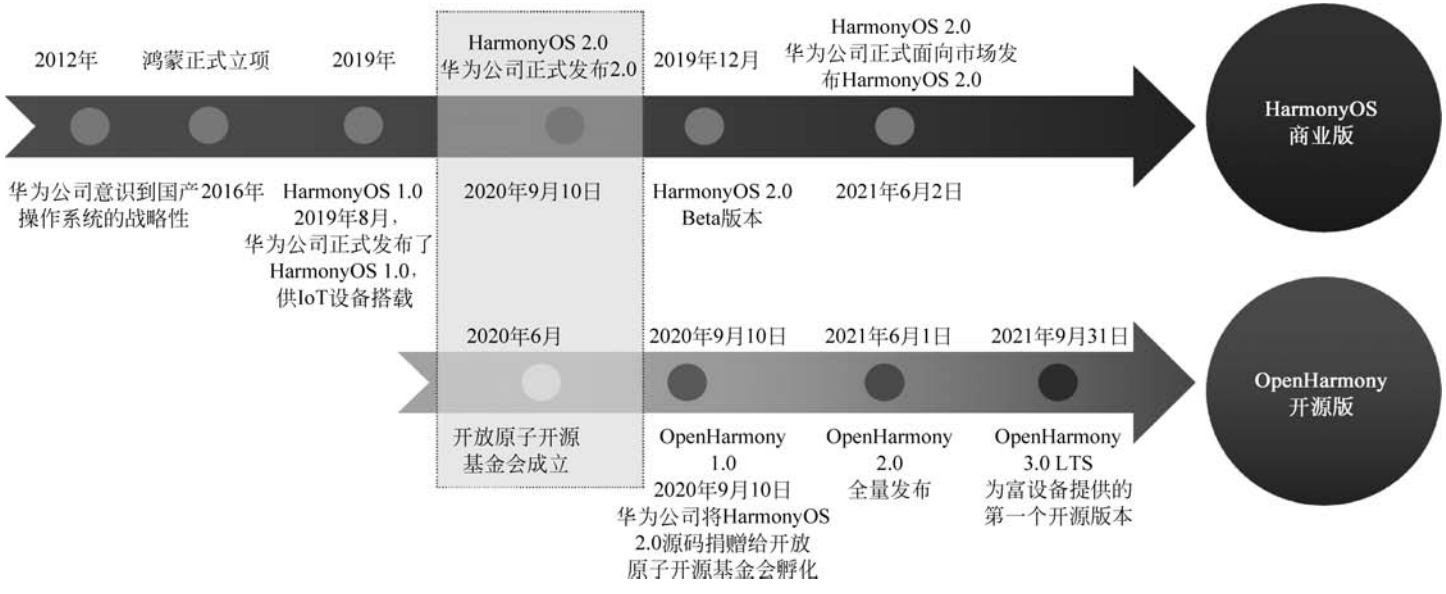

图 1.12　HarmonyOS 与 OpenHarmony 发展历程

OpenHarmony 代码以组件的形式开放，目前包含 32 个子系统、281 个代码仓库，如图 1.13 所示。

Gitee 网站 https://gitee.com/openharmony。

OpenHarmony 官网 https://openharmony.io/。

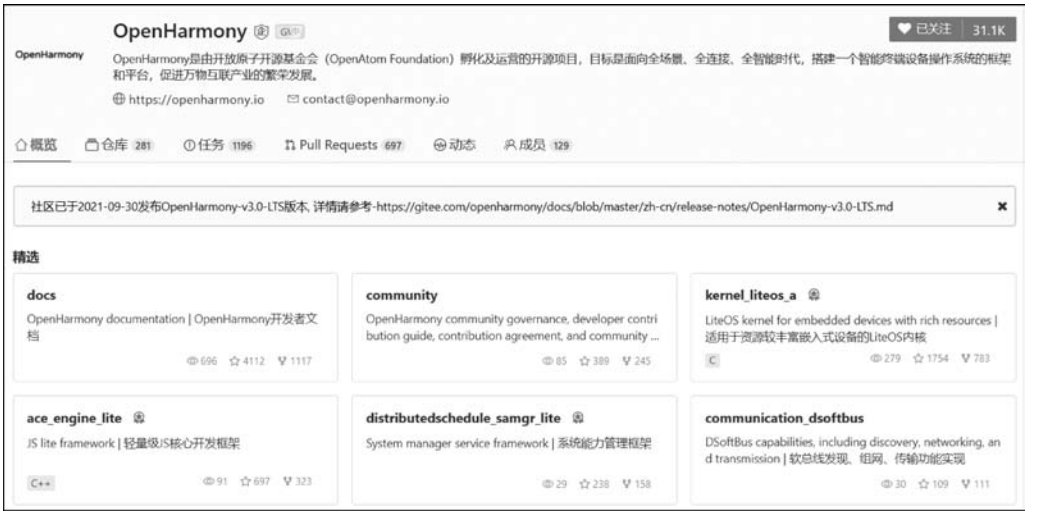

图 1.13　OpenHarmony 代码仓库

1.6　HarmonyOS 与 Fuchsia OS

Fuchsia OS 是谷歌开发的操作系统，与基于 Linux 内核的 Chrome OS 和 Android 等不同，Fuchsia OS 基于新的名为 Zircon 的微内核，受 Little Kernel 启发，用于嵌入式系统，主要使用 C 语言和 C++编写。Fuchsia OS 的设计目标之一是可运行在众多的设备上，包括手机和计算机。

Fuchsia OS 是谷歌使用单一的操作系统去统一整个物联网生态圈的一种新的操作系统架构模式。Fuchsia OS 的独特之处在于其微内核、模块化设计。

Fuchsia OS 主要有以下优点：

(1) 原生进程沙箱，解决应用安全和分发问题。

(2) 全新微内核架构设计，Fuchsia OS 不再基于宏内核 Linux，而是采用了谷歌自己研发的全新微内核 Zircon。

(3) 稳定的驱动接口，硬件厂商可独立维护硬件驱动。支持的架构包括 x86-64 和 ARM 64，支持的设备从 IoT 到服务器。

(4) 系统模块化、分层、设备厂商可以灵活定制专有系统，Fuchsia OS 的另一大特点是其模块化系统体系结构与模块化的应用程序设计，这使谷歌的合作伙伴与硬件制造商可以用自己的模块替代 Fuchsia OS 的单个系统级别，从而在不影响其他级别功能的同时改进或

扩展 Fuchsia OS，增强了系统可扩展性。

(5) Vulkan 图形接口、3D 桌面渲染 Scenic、全局光照。

(6) Flutter 应用开发框架，Flutter SDK 能在 Android 运行，兼容 Android 应用，是使用 Dart 和 Flutter 作为界面开发的语言和框架。此外，与安卓相比，Fuchsia OS 无论是存储器还是内存等硬件要求都大幅降低。

1.6.1 Fuchsia OS 系统架构

从 Fuchsia OS 技术架构来看，内核层 Zircon 的基础是 Little Kernel(简称 LK)，LK 是专为嵌入式应用中小型系统设计的内核，代码简洁，适合嵌入式设备和高性能设备，例如 IoT、移动可穿戴设备等，如图 1.14 所示。

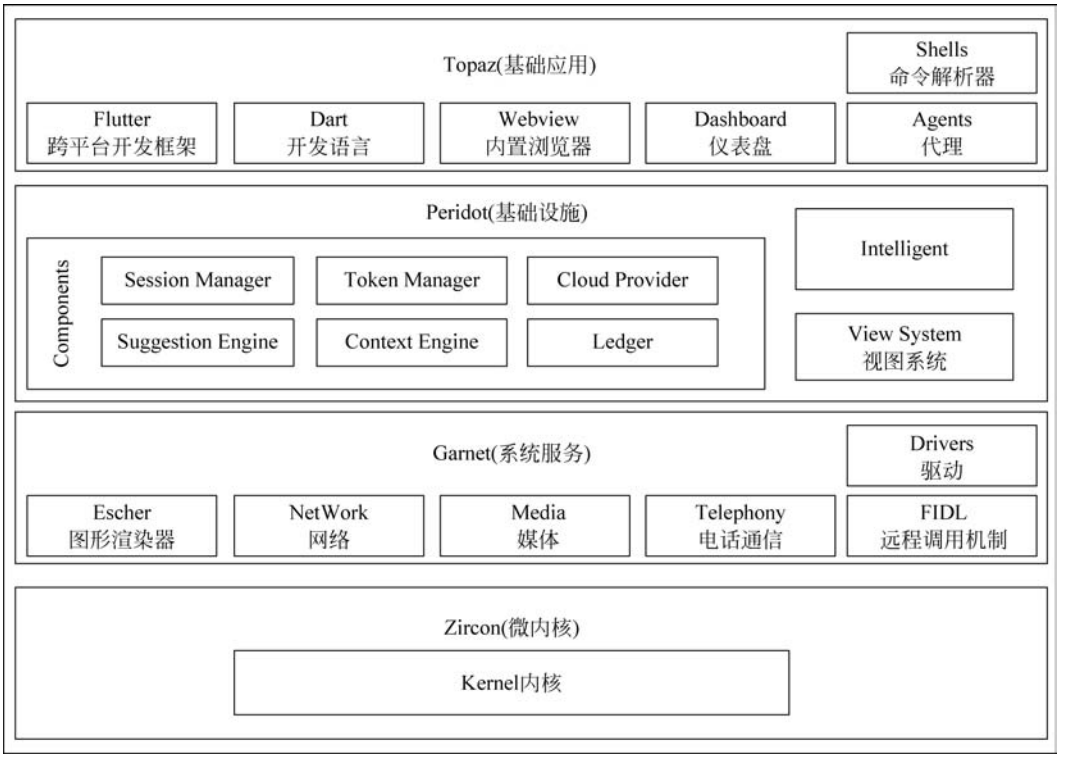

图 1.14 Fuchsia OS 系统架构图

Fuchsia OS 的系统架构也基于模块化。操作系统由 4 个独立级别模块组成，每个级别都有其自己的任务：Zircon、Garnet、Peridot 和 Topaz。

下面逐一认识一下 Fuchsia OS 的系统架构模块。

1. 微内核(Zircon)

Fuchsia OS 并非基于 Linux 内核，而是基于一个新的名为 Zircon /'zɜ:k(ə)n/ 的微内核架构开发，Zircon Kernel 最早由 LK(Little Kernel)的一个分支发展而来。

LittleKernel是一个适用于嵌入式设备和BootLoader(BootLoader是嵌入式系统在加电后执行的第一段代码,在它完成CPU和相关硬件的初始化之后,再将操作系统映像或固化的嵌入式应用程序装载到内存中,然后跳转到操作系统所在的空间,启动操作系统运行)等场景的微型操作系统(微内核),提供了线程调度、互斥量和定时器等支持。在嵌入式ARM平台,Little Kernel的核心大约15～20KB。部分芯片厂商(如高通、联发科等)的Android操作系统使用Little Kernel作为其BootLoader和TEE(Trusted Execution Environment)的安全区运行环境。

不同于为微控制器设计的Little Kernel(微控制通常只有非常有限的RAM,少量的外设及运行任务),Zircon的设计目标是运行在具备更强处理能力的智能手机及个人计算机上。Zircon仅支持64位处理器系统,在Little Kernel的基础上增加了进程的概念、添加了用户态、基于能力的安全模型、MMU(Memory Management Unit,内存管理单元)的支持及系统调用等。

Zircon包含Fuchsia OS的内核、设备管理器、最核心的第一层设备驱动程序及底层系统库(如libc和launchpad)。此外,Zircon还提供FIDL(Fuchsia OS接口定义语言),这是一种用于进程间通信的协议。FIDL是独立于编程语言的,能与流行的编程语言(如C、C++、Dart、Go和Rust)进行连接。

这里简单介绍一下微内核与宏内核的关系,微内核(Micro Kernel)是内核的一种精简形式,如图1.15所示。微内核中只有最基本的调度、内存管理。驱动、文件系统等都采用用户态的守护进程去实现。微内核的优点是超级稳定,驱动等的错误只会导致相应进程"死掉",不会导致整个系统都崩溃,做驱动开发时,如果发现错误,则只需Kill进程,修正后重启进程就行了,比较方便,但缺点是效率低。

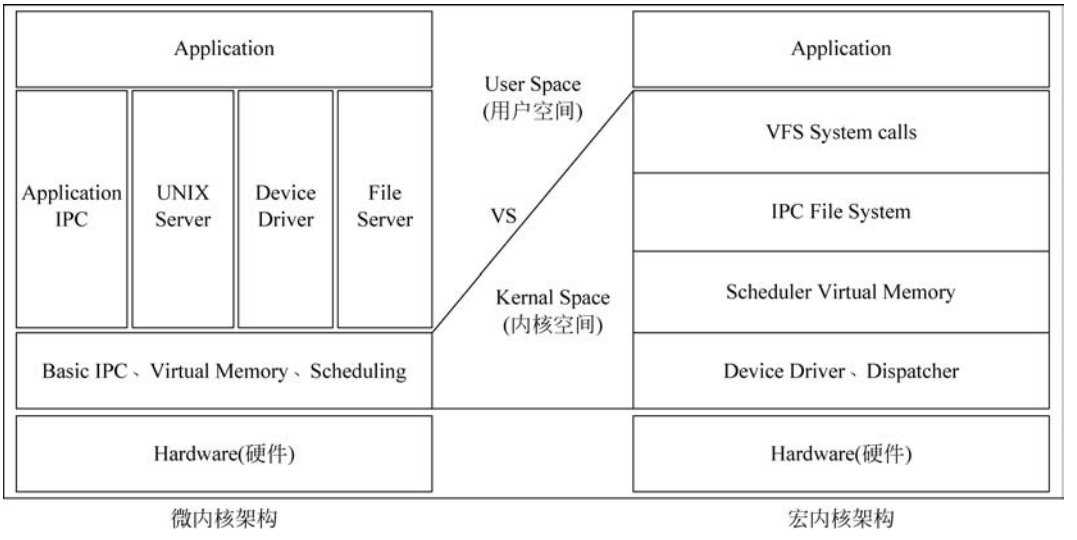

图1.15　微内核与宏内核比较图

宏内核(Monolithic Kernel)简单来讲,就是把很多东西都集成进内核,例如 Linux 内核,除了最基本的进程、线程管理、内存管理外,文件系统、驱动、网络协议等都在内核里面。宏内核的优点是效率高。缺点是稳定性差,开发过程中的 Bug 经常会导致整个系统崩溃,具体区别如表 1.3 所示。

表 1.3 微内核与宏内核比较

比较项	微内核	宏内核
基本概念	用户服务和内核服务运行在不同的地址空间中	用户服务和内核服务运行在相同的地址空间中
尺寸	比较小	比微内核大
执行速度	容易扩展	不容易扩展
可扩展性	单个服务崩溃不影响全局	单个服务崩溃往往意味着整个系统崩溃
代码开发	需要开发的代码量大	平台提供的代码多,相对需要开发的代码量少
系统举例	QNX、Symbian、L4Linux、Singularity、K42、Mac OS X、PikeOS、Minix、Coyotos	Linux、BSDS(FreeBSD、penBSD、NetBSD)、Microsoft Windows (95、98、Me)、Solaris、OS-9、AIX、DOS、HP-UX OpenvMS、XTS-400 等

2. 系统服务(Garnet)

Garnet 是基于 Zircon 内核之上的系统服务层。提供了设备级别的各种系统服务和网络,以及媒体和图形服务,例如:软件安装、系统管理及与其他系统的通信。Garnet 包含图形渲染器 Escher、程序包管理、更新系统 Amber 及文本和代码编辑器 Xi。

3. 基础设施(Peridot)

Peridot 用于处理模块化应用程序设计,Peridot 的另外两个重要组件可直接用于模块 Ledger 和 Maxwell。

Ledger:Ledger 是基于云的存储系统(分布式存储系统),它为每个 Fuchsia 组件(模块或代理)提供单独的数据存储,可在不同设备之间同步,这使用户可以在当前 Fuchsia OS 的设备上继续停留在其他 Fuchsia OS 设备上的相应位置。

Maxwell:通过 Maxwell,谷歌在 Fuchsia OS 中集成了一个组件,该组件向用户提供了人工智能。就像 Fuchsia OS 一样,Maxwell 具有模块化设计。AI 系统由一系列代理组成,这些代理分析用户的行为及其所使用的内容,在后台确定合适的信息,并将建议转发给操作系统。例如,应加载哪些模块或故事以适合用户在特定时间的行为。谷歌语言助手也是 AI 组件的一部分,该组件将在 Fuchsia OS 项目的框架内以代码 Kronk 的形式进一步开发。

4. 基础应用(Topaz)

Topaz 作为基础应用层,Topaz 提供 Flutter 支持,而有了 Flutter 的支持,各种华丽的应用程序可以提供日常使用的功能齐全的应用程序。例如,现在最令人印象深刻的当然是 Armadillo UI,它是 Fuchsia OS 主要用户界面和主屏幕,Armadillo 现在被 Ermine 取代,这

是一个面向开发人员的 shell,专为测试的明确目的而设计 Fuchsia OS 的应用。

1.6.2 Fuchsia OS 与产业

HarmonyOS 与 Fuchsia OS 均为搭建物联网生态而推出的系统,而搭建完善的生态体系需要接入不同领域的合作伙伴,因此,HarmonyOS 与 Fuchsia OS 未来竞争方向主要是在合作伙伴上。

谷歌目前计划将 Fuchsia OS 应用于智能手机、智能家居市场与笔记本电脑市场,由于 Fuchsia OS 的进展较慢,华为 HarmonyOS 目前已经在将近 2 亿台华为设备上安装并运行,同时华为通过 HarmonyOS Connect 计划已经接入了大量第三方产品,两者差异如表 1.4 所示。

表 1.4 HarmonyOS 与 Fuchsia OS 比较

比 较 项	HarmonyOS	Fuchsia OS
应用场景	手机、计算机、智能手表、手环、智慧屏、智能音箱、路由器等智能设备	智能家居、移动终端等智能嵌入式设备,未来可能被应用于智能手机与笔记本电脑
特征	实现跨终端无缝协同体验。统一的系统 IDE 支撑着开发人员只需一次开发,便可以实现将应用部署到不同的设备上,可大幅提高开发效率	与目前 Android 相比,无论是存储器还是内存之类的硬件要求都大幅降低,可以看出这是一款面向物联网的家用电器的系统
内核机制	目前基于多内核设计	基于微内核 Magenta(后期改名为 Zircon)的新内核

1.7 本章小结

鸿蒙操作系统作为国产物联网开源操作系统,将会不断推进中国物联网行业的发展,它不仅给行业带来了新的动力和发展机会,同时不断地影响和改变着人们的生活。学习和掌握鸿蒙操作系统开发给软件、硬件开发者提供了新的机会。