鸿蒙生态应用开发

白皮书V1.0

HUAWEI

版权所有 © 华为终端有限公司 2022。 保留一切权利。

本材料所载内容受著作权法的保护，著作权由华为公司或其许可人拥有，但注明引用其他方的内容除外。未经华为公司或其许可人事先书面许可，任何人不得将本材料中的任何内容以任何方式进行复制、经销、翻印、播放、以超级链路连接或传送、存储于信息检索系统或者其他任何商业目的的使用。

商标声明

以上为华为公司的商标（非详尽清单），未经华为公司书面事先明示许可，任何第三方不得以任何形式使用。

注意

华为会不定期对本文档的内容进行更新。

本文档仅作为使用指导，文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为终端有限公司

地址： 广东省东莞市松山湖园区新城路 2 号

网址： https://consumer.huawei.com

CONTENT

01 万物互联时代应用开发的机遇、挑战和趋势

02 鸿蒙生态应用核心技术理念

1）一次开发，多端部署 7

2）可分可合，自由流转 16

3）统一生态，原生智能 19

03 鸿蒙生态应用开发能力全景

1）赋能套件 24

2）鸿蒙开发套件 26

3）三方库 44

4）开发者支持平台 45

04 高效开发与测试

1）典型开发场景 48

2）设计 48

3）ArkTS 语言 49

4）ArkUI 框架 50

5）用户程序框架 61

6）SDK 64

7）集成开发环境 69

8）测试工具 78

05 统一上架与多端分发

1）快速上架 91

2）应用分发 95

3）服务分发 98

06 自由流转与分布式运行环境

1）价值与架构定义 104

2）跨端迁移 106

3）多端协同 108

07 全方位运维分析

08 全场景案例参考

09 附录：术语

Chapter 1

万物互联时代应用开发的机遇、挑战和趋势

经过十多年的发展，传统移动互联网的增长红利已渐见顶。万物互联时代正在开启，应用的设备底座将从几十亿手机扩展到数百亿 IoT 设备。GSMA 预测到 2025 年，全球物联网终端连接数量将达 246 亿个，其中消费物联网终端连接数量将达 110 亿个（注：数据来自于全球移动通信系统协会发布的《2020 年移动经济》报告）。IDC 预计到 2025 年，中国物联网总连接量将达到 102.7 亿个（注：数据来自于 IDC 发布的《中国物联网连接规模预测，2020—2025》报告）。全新的全场景设备体验，正深入改变消费者的使用习惯。 同时应用开发者也面临设备底座从手机单设备到全场景多设备的转变，通过全场景多设备作为全新的底座，为消费者带来万物互联时代更为高效、便捷的体验。

新的场景同时也带来了新的挑战。开发者不仅需要支持更加多样化的设备，还需要支持跨设备的协作。不同设备类型意味着不同的传感器能力、硬件能力、屏幕尺寸、操作系统和开发语言，还意味着差异化的交互方式。同时跨设备协作也让开发者面临分布式开发带来的各种复杂性，例如跨设备的网络通信、数据同步等。若采取传统开发模式，适配和管理工作量将非常巨大。当前移动应用开发中遇到的主要挑战包括：

- 针对不同设备上的不同操作系统，重复开发，维护多套版本。
- 多种语言栈，对人员技能要求高。
- 多种开发框架，不同的编程范式。
- 命令式编程，需关注细节，变更频繁，维护成本高。

与此同时，AI 时代全面来临，在 PC 互联网到移动互联网到智能化终端演进过程中，AI 计算主要在云端数据中心进行，非常依赖网络，具有一定的时延，且数据传输的安全性、私密性不能得到有效保证。随着人们对交互和信息获取的智能化要求越来越高，移动设备的计

算能力越来越强，在设备侧就能提供 AI 的相关能力，例如自然语言交互、环境智能感知、图像识别等。如何快速地使用设备侧的强大 AI 能力，使自己的应用更加智能化，进而更好的服务消费者，也是开发者面临的全新挑战。

移动终端上的应用生态发展到今天也面临着变革。传统厚重的 App，整体体验好，功能齐全，但开发成本高、周期长，且存在搜索，安装，升级，卸载等一系列需要用户主动关注的显性操作，这些显性操作给用户带来了实质性的使用成本。轻量化、可快速达成消费者意图、可独立执行、完成单一功能的程序实体正成为新的趋势，例如小程序、App Clips、快应用等。根据阿拉丁指数的统计，全网小程序已经突破 700 万个（注：数据来自于阿拉丁研究院发布的《2021 年度小程序互联网发展白皮书》），远超 App 数量。大型应用开发者普遍向用户提供轻量化程序实体。在很多特定的使用场景下，小程序等轻量化程序实体的使用占比已超过 App，成为面向用户的主要触达方式。

轻量化的程序实体所具备的“即用即走、无需安装卸载、永远最新”的特征，也推动了 App 基于搜索下载的“人找应用”的传统分发向“服务找人”的智慧分发的演进。App 遵循“搜索、下载、安装、使用”的模式，用户主动发现的成本高，拉新、促活、召回的全生命周期流程相对被动。轻量化的程序实体具有即用即走的体验，可通过各类终端的系统级智慧入口进行分发，甚至可以在三方 App 中分发，依托无所不在的入口流量和标签化识别，向用户主动提供精准服务。配合 CPS（Cost Per Sale）等商业模式，可以为开发者带来更高的 ROI（Return of Investment）。

为了更好的抓住机遇，应对万物互联所带来的一系列挑战，新的应用生态应该具备如下特征：

- 单一设备延伸到多设备：应用一次开发就能在多个设备上运行，软件实体能够从单一设备转移到其他设备上，且多个设备间能够协同运行，给消费者提供全新的分布式体验。
- 厚重应用模式到轻量化服务模式：提供轻量化的服务，最小化资源消耗，一步直达，快速完成消费者特定场景的任务。
- 集中化分发到 AI 加持下的智慧分发：为消费者提供智慧场景服务，实现“服务找人”。
- 纯软件到软硬芯协同的 AI 能力：提供软硬芯协同优化的原生 AI 能力，全面满足应用高性能诉求。

Chapter 2

鸿蒙生态应用

核心技术理念

- 1）一次开发，多端部署
- 2）可分可合，自由流转
- 3）统一生态，原生智能

在万物智联时代重要机遇期，鸿蒙系统结合移动生态发展的趋势，提出了三大技术理念：一次开发，多端部署；可分可合，自由流转；统一生态，原生智能。

图 2-1：核心技术理念

1）一次开发，多端部署

“一次开发，多端部署”指的是一套代码，一次开发上架，多端按需部署。目的是为了支撑开发者高效地开发多种终端设备上的应用。为了实现这一目的，鸿蒙系统提供了几个核心能力，包括多端开发环境，多端开发能力以及多端分发机制。

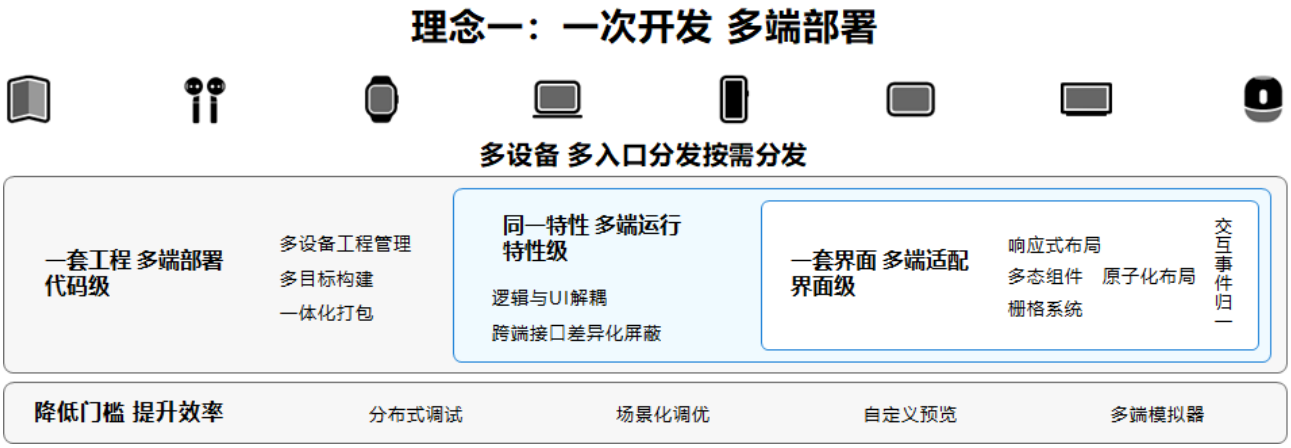

图 2-2：一次开发 多端部署

多端开发环境

HUAWEI DevEco Studio 是面向全场景多设备提供的一站式开发平台，支持多端双向实时预览、分布式调优、分布式调测、超级终端模拟、低代码可视化开发等能力，帮助开发者降低成本、提升效率、提高质量。HUAWEI DevEco Studio 提供的核心能力如下图所示：

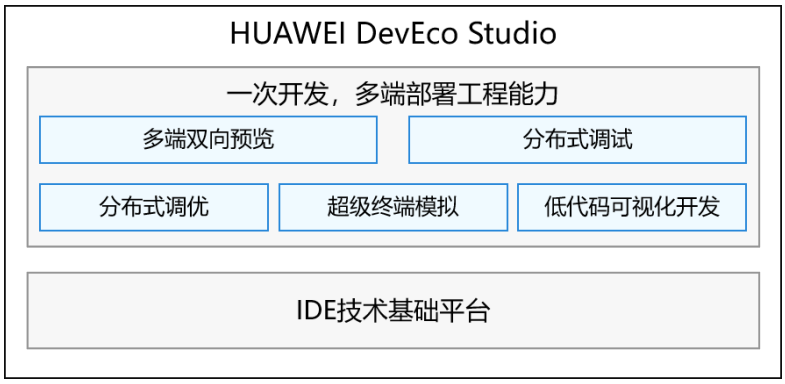

图 2-3：HUAWEI DevEco Studio 核心功能和特征

1. 多端双向预览

在鸿蒙生态应用的开发阶段，因不同设备的屏幕分辨率、形状、大小等差异，开发者需要在不同设备上查看界面 UI 显示，确保实现效果与设计目标一致。传统的开发模式下，开

发者需要获取大量不同的真机设备用于测试验证。HUAWEI DevEco Studio 提供了多种设备的双向预览能力，支持同时查看 UI 代码在多个设备上的预览效果，并支持 UI 代码和预览效果的双向定位修改。

2. 分布式调试

鸿蒙生态应用具有天然的分布式特征，体现在同一个应用在多个设备上会有大量的交互。开发过程中，对这些交互进行调试时，需要对每个设备分别建立调试会话，并且需要在多个设备之间来回切换，容易造成调试不连续、操作繁琐等问题。为了提升开发效率，HUAWEI DevEco Studio 提供了分布式调试功能，支持跨设备调试，通过代码断点和调试堆栈可以方便地跟踪不同设备之间的交互，用于定位多设备互动场景下的代码缺陷。

图 2-4：分布式调试交互图

3. 分布式调优

分布式应用的运行性能至关重要。在跨端迁移场景中，需要应用在目标设备上快速启动，以实现和原设备之间的无缝衔接；在多端协同场景中，需要应用在算力和资源不同的多个设

备上都能高效运行，以获得整体的流畅体验。以往开发者在分析分布式应用的性能问题时，需要单独查看每个设备的性能数据，并手动关联分析这些数据，操作繁琐，复杂度高。HUAWEI DevEco Studio 提供了分布式调优功能，支持多设备分布式调用链跟踪、跨设备调用堆栈缝合，同时采集多设备性能数据并进行联合分析。

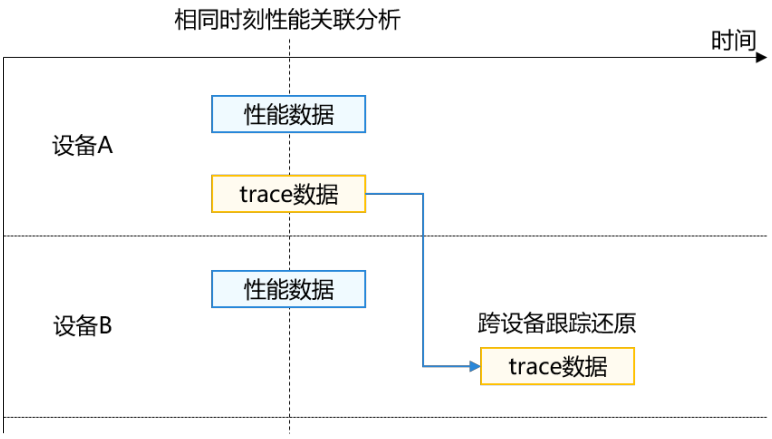

图 2-5：多设备联合分析

4. 超级终端模拟

移动应用开发时需要使用本地模拟器来进行应用调试，实现快速开发的目的。鸿蒙生态应用需要运行在多种不同类型的设备上，为此 HUAWEI DevEco Studio 提供了不同类型的终端模拟，支持开发者在多个模拟终端上进行开发调试，降低门槛、节约成本。同时，多个模拟终端、真机设备也可以自由地组成超级终端，进一步降低开发者获取分布式调测环境的难度。

5. 低代码可视化开发

低代码开发提供 UI 可视化开发能力，支持自由拖拽组件和可视化数据绑定，可快速预览效果，所见即所得。通过拖拽式编排、可视化配置的方式，帮助开发者减少重复性的代码编写，快速地构建多端应用程序。低代码开发的产物如组件、模板等可以被其他模块的代码引用，并且能通过跨工程复用，支持开发团队协同完成复杂应用的开发。

多端开发能力

应用如需在多个设备上运行，需要适配不同的屏幕尺寸和分辨率、不同的交互方式（如触摸和键盘等）、不同的硬件能力（如内存差异和外设差异等），开发成本较高。因此，多端开发能力的核心目标是降低多设备应用的开发成本。为了实现该目标，鸿蒙系统提供了以下几个核心能力，支持界面和业务逻辑代码复用，帮助开发者降低开发与维护成本，提高代码复用度。

1. 多端 UI 适配

不同设备屏幕尺寸、分辨率等存在差异，系统需要对屏幕进行逻辑抽象，包括尺寸和物理像素，并提供丰富的自适应/响应式的布局和视觉能力，方便开发者进行不同屏幕的界面适配。

屏幕逻辑抽象：鸿蒙系统提供虚拟像素 vp（virtual pixel）对分辨率进行抽象，为应用开发者提供统一单位，不同设备的系统会在显示时，在底层进行像素转化。不同设备的尺寸存在差异，鸿蒙操作系统根据设备的屏幕水平宽度，抽象和定义了四种尺寸：超小（xs）、小（sm）、中（md）、大（lg）。这四种抽象后的屏幕尺寸与日常使用的设备屏幕类型有一

定的对应关系，例如：超小对应智能穿戴设备，小对应手机和折叠屏，中对应平板，大对应 PC 与智慧屏。开发者可面向应用运行的目标设备进行所属屏幕类型的适配。

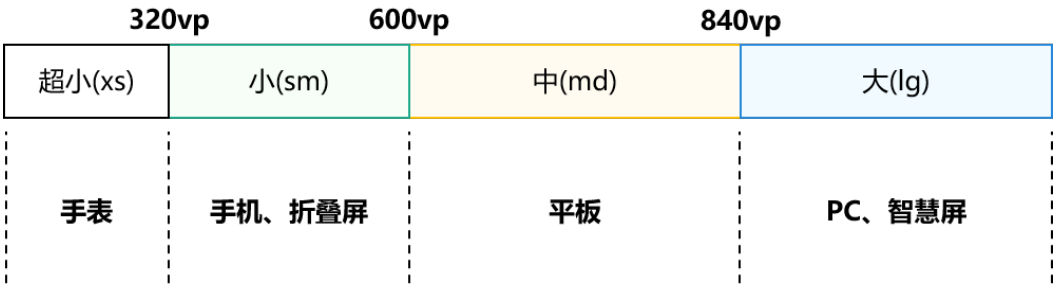

图 2-6：尺寸抽象化

布局：鸿蒙系统提供的布局主要分为自适应布局和响应式布局。自适应布局是当外部容器大小发生变化时，容器内元素可以根据相对关系自动变化以适应外部容器变化的布局能力。相对关系包含占比、固定宽高比、显示优先级等。当前自适应布局能力主要有 7 种：拉伸能力、均分能力、占比能力、缩放能力、延伸能力、隐藏能力、折行能力。自适应布局能力可以实现界面显示随外部容器大小连续变化。响应式布局是当外部容器大小发生变化时，元素可以根据断点、栅格或特定的特征（如屏幕方向、窗口宽高等）自动变化以适应外部容器变化的布局能力。当前响应式布局能力主要有 3 种：断点、媒体查询、栅格布局。

视觉：鸿蒙系统提供的视觉样式能力，包括分层参数、多态组件和主题。

2. 事件交互归一

不同设备间的交互方式等存在差异，如触摸、键盘、鼠标、语音、手写笔等，系统需要对不同输入方式进行统一处理，向开发者提供归一的逻辑交互事件。

以缩放交互为例，通过多指触控的张合来完成缩放动作，在多设备场景下，缩放交互会出现多种不同的操作输入方式。为了让应用更好的支持这些缩放交互，鸿蒙系统提供如下统一的缩放交互规则。

表 2-1：缩放交互的规则

操作方式	触屏双指捏合交互	键盘 Ctrl 键+鼠标滚轮交互	键盘 Ctrl 键+“+/-”键交互	触控板双指捏合交互	表冠旋转交互
上报事件	触屏双指捏合事件	按键+滚轮组合事件	按键组合点击事件	触控板双指捏合事件	表冠旋转事件

3. 设备能力抽象

不同设备间的软、硬件能力等存在差异，如设备是否具备定位能力、是否具备摄像头、内存从百 KiB~GiB 等，系统需要对设备能力进行逻辑抽象，并提供接口来查询设备是否支持某一能力，方便开发者进行不同软、硬件能力的功能适配。在鸿蒙系统中，使用 SystemCapability（简写为 SysCap）定义每个部件对应用开发者提供的系统软硬件能力。应用开发者基于统一的方式访问不同设备的能力。

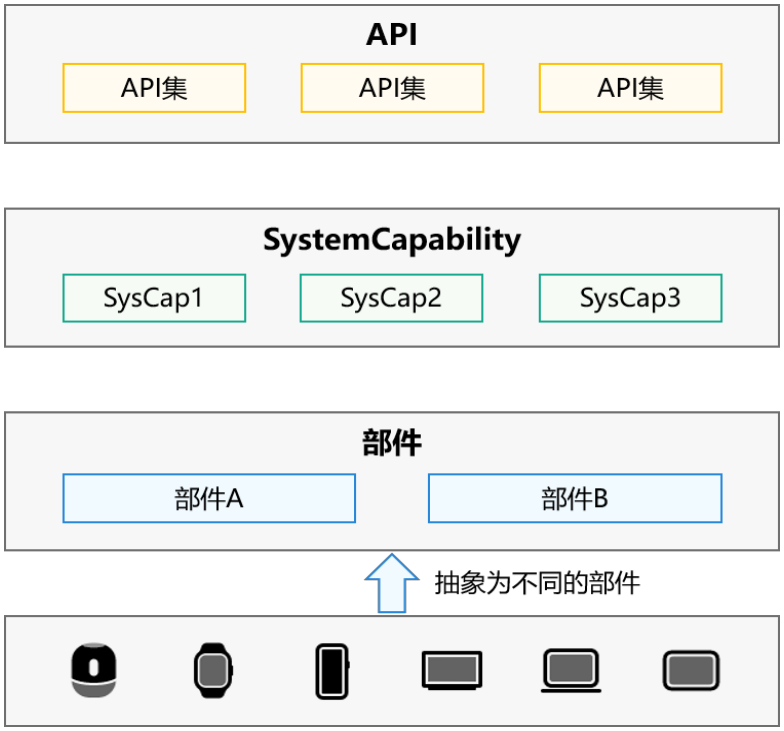

图 2-7：API、SystemCapability、部件和设备的关系

多端分发机制

如果需要开发多设备上运行的应用，一般会针对不同类型的设备多次开发并独立上架。开发和维护的成本大，为了解决这个问题，鸿蒙系统提供了“一次开发，多端部署”的能力，开发者开发多设备应用，只需要一套代码，一次打包出多个 HAP，统一上架，即可根据设备类型按需进行分发。

除了可以开发传统的应用，开发者还可以开发原子化服务。原子化服务是一种面向未来的服务提供方式，具有独立入口的、免安装的、可为用户提供一个或多个便捷服务的应用程序形态。鸿蒙系统为原子化服务提供了更多的分发入口，方便用户获取，同时也增加了原子化服务露出的机会。

1. 多设备按需分发

鸿蒙系统提供了两种模式帮助开发者基于“一次开发，多端部署”能力分发应用和原子化服务到不同设备上。

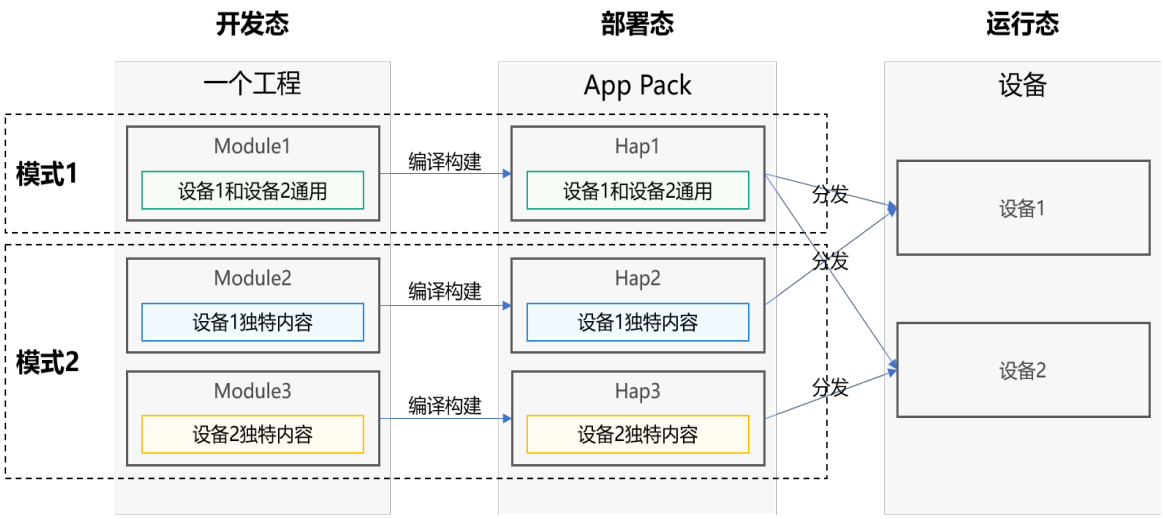

图 2-8：多设备按需分发的两种模式

- 模式 1：应用或服务的 UI 自适应不同尺寸的设备屏幕，并且在不同设备的功能相同，可以实现多设备共享一个 HAP 包。这种场景下建议开发者通过一个模块来开发，并配置该模块支持多设备，然后再编译构建生成一个 HAP，分发到不同类型的设备上运行。
- 模式 2：应用或服务的 UI、功能在不同设备间存在差异，无法实现 HAP 包多设备归一。可根据实际情况设置不同模块适用的设备类型，编译构建多个 HAP 包，一起上架。HUAWEI AppGallery Connect 会自动提取 HAP 中的设备类型的配置信息，为对应的设备自动分发正确的 HAP 包组合。

2. 多入口按需分发

鸿蒙系统为原子化服务提供了多设备、多入口的分发能力，基于场景和用户意图拉起原子化服务，实现“服务直达”。鸿蒙生态提供的丰富入口如下图所示：

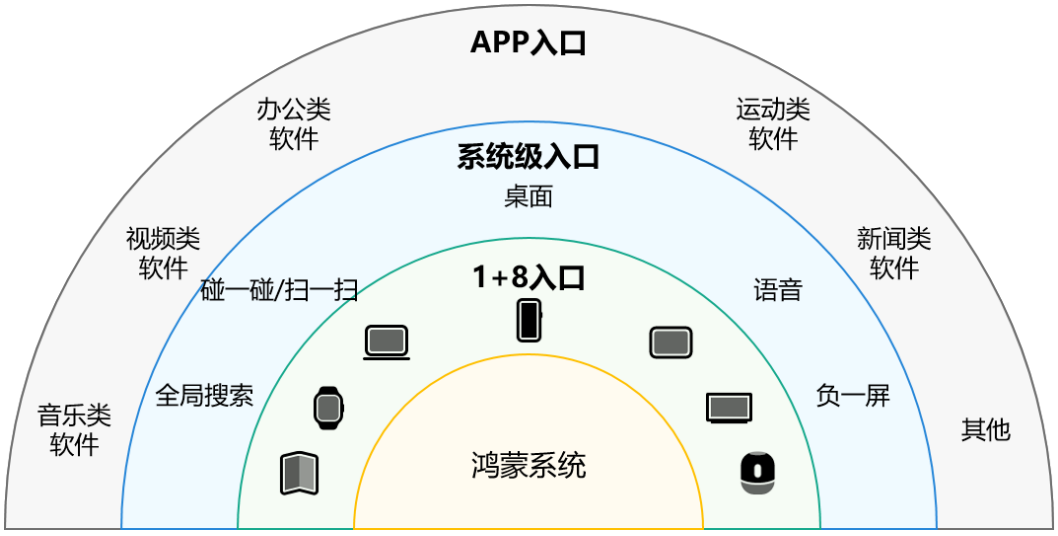

图 2-9：多入口按需分发

2）可分可合，自由流转

原子化服务是鸿蒙系统提供的一种全新的应用形态，具有独立入口，用户可通过点击、碰一碰、扫一扫等方式直接触发，无需显式安装，由程序框架后台静默安装后即可使用，可为用户提供便捷服务。

传统移动生态下，开发者通常需要开发一个原生应用版本，如果提供小程序给用户，往往需要开发若干个独立的小程序。鸿蒙生态下，鸿蒙原生支持原子化服务开发，开发者无需

维护多套版本，通过业务解耦将应用分解为若干原子化服务独立开发，按需根据场景组合成复杂应用。

原子化服务基于鸿蒙系统 API 开发，支持运行在 1+8+N 设备上，供用户在合适的场景、合适的设备上便捷使用。原子化服务是支撑可分可合，自由流转的轻量化程序实体，帮助开发者的服务更快触达用户。具备如下特点：

- 触手可及：原子化服务可以在服务中心发现并使用，同时也可以基于合适场景被主动推荐给用户使用，例如用户可在服务中心和小艺建议中发现系统推荐的服务。
- 服务直达：原子化服务无需安装卸载，“秒开体验”，即点即用，即用即走。
- 服务卡片：支持用户无需打开原子化服务便可获取服务内重要信息的展示和动态变化，如天气、关键事务备忘、热点新闻列表。
- 自由流转：原子化服务支持运行在多设备上并按需跨端迁移，或者多个设备协同起来给用户提供最优的体验。例如手机上未完成的邮件，迁移到平板继续编辑，手机用作文档翻页和批注，配合智慧屏完成分布式办公；例如分布式游戏场景，手机可作为手柄，与智慧屏配合玩游戏，获得新奇游戏体验。

可分可合

在开发态，开发者通过业务解耦，把不同的业务拆分为多个模块。在部署态，开发者可以将一个或多个模块自由组合，打包成一个 App Pack 统一上架。在分发运行态，每个 HAP

都可以单独分发满足用户单一使用场景，也可以多个 HAP 组合分发满足用户更加复杂的使用场景。

开发者可以在以下两种模式中选择，进行鸿蒙生态应用、原子化服务的打包和上架。

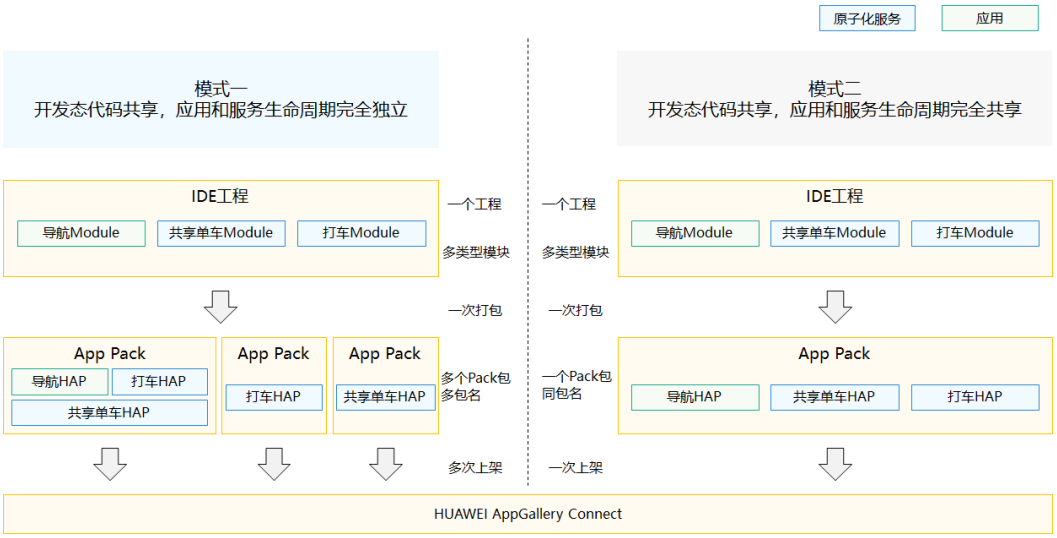

图 2-10：两种打包上架模式

- 模式一：打包成多个 App Pack，不同 App Pack 的包名是不一样的，每个 App Pack 都需要单独上架。在运行态，应用和服务的生命周期完全独立。
- 模式二：打包成一个 App Pack，App Pack 里面的 HAP 包名相同，统一上架。在运行态，应用生命周期完全共享。

自由流转

传统应用只能在单个设备内运行，当用户有多个设备，且要完成多个任务时，则需要在多个设备间来回切换。因此应用能够在设备之间流转，不间断给用户提供服务的能力就变得非常重要。

鸿蒙系统提供了自由流转的能力，使得开发者可以方便地开发出跨越多个设备的应用，用户也能够方便地使用这些功能。

自由流转可分为跨端迁移和多端协同两种情况。它们分别是时间上的串行交互和时间上的并行交互。自由流转不仅带给用户全新的交互体验，也为开发者搭建了一座从单设备时代通往多设备时代的桥梁。关于跨端迁移和多端协同详细说明，会在第六章中详细展开。

3）统一生态，原生智能

由于应用仅能运行在支持其运行环境的操作系统上，开发者要使其应用能运行在多操作系统上，则需要对不同的系统进行单独的开发或者适配。为了降低开发成本、提高代码复用率、减少多个平台重复开发工作量，业界推出了很多跨平台三方框架。

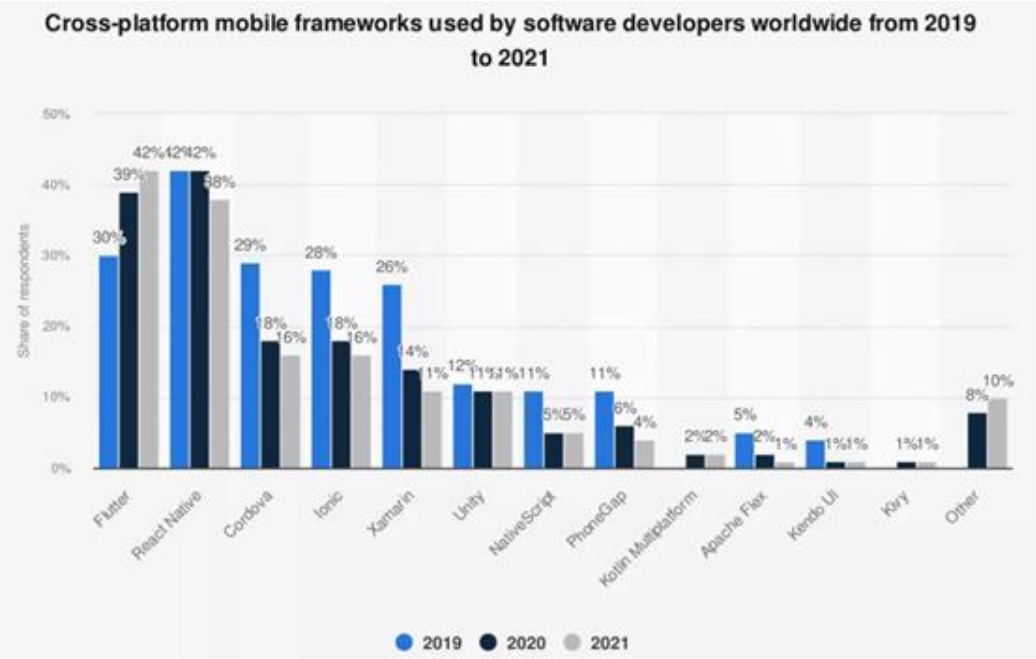

图 2-11：业界跨平台三方框架使用率（数据来源于 statista）

鸿蒙系统倡导应用生态统一、多方共建，支持开发者根据自身的业务场景，自由选择原生框架、三方跨平台框架来进行鸿蒙生态应用开发。

同时为了满足日益增长的应用智能化诉求，鸿蒙系统内置了多层次、丰富的 AI 开放能力，对开发者提供简洁易用的 API，帮助开发者快速集成 API，助力应用智能化。

统一生态

鸿蒙系统支持业界主流跨平台开发框架，通过多层次的开放能力提供统一接入标准，实现三方框架快速接入，支撑快速丰富鸿蒙生态应用、原子化服务。

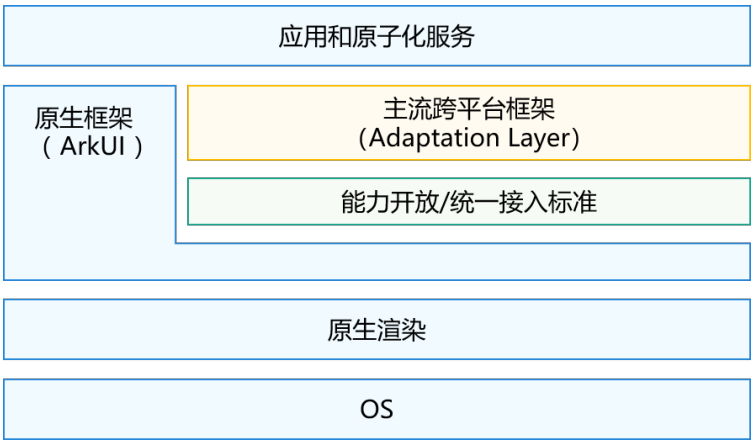

图 2-12：统一生态

原生智能

鸿蒙系统提供开箱即用的原生 AI 能力，降低智能应用的开发门槛，帮助开发者快速实现应用智能化。同时也提供软硬芯协同优化的系统级推理框架并预留扩展，满足开发者的高阶性能优化诉求。

图 2-13：AI 能力开放

AI 能力开放具体包括：

服务能力开放层：为原子化服务提供多样化、场景化的智能入口，让开发者的服务能够更加精准地触达用户。

应用能力开放层：面向应用开发者提供的简单易用、功能强大的场景化 AI 能力。

- 自然语言交互：包含 ASR、TTS、意图识别、语种检测、文本翻译，分词、词性标注、实体识别、关键字提取等。
- 智慧语音：可视可说，提供通过说出界面文字、图标、角标信息即可轻松操控界面的能力。
- 计算视觉：包含图片识别、文本识别、视频内容分析、人脸识别、人体检测等常用的能力。
- 情景感知：提供对设备使用场景的智能感知能力，例如地理围栏、快照等。

芯片能力开放层：向应用开发者开放芯片 AI 计算（含 NPU/CPU/GPU 多计算单元）能力，兼容 TensorFlow、Caffe、 MindSpore、Paddle、ONNX 等主流框架。同时提供统一的推理框架 MindSpore Lite 开放接口，开发者无需单独预置推理框架，减少应用、服务

包大小。MindSpore Lite 是一个极速、极智、极简的 AI 引擎，为用户提供端到端的解决方案。其具有三大优势：

- 极致性能：高效的内核算法和汇编级优化，最大化发挥硬件算力，最小化推理时延和功耗。
- 轻量化：提供超轻量的解决方案，支持模型量化压缩，模型更小跑得更快，使能 AI 模型极限环境下的部署执行。
- 高效部署：支持 MindSpore/TensorFlow Lite/Caffe/Onnx 模型，提供模型压缩、数据处理等能力，统一训练和推理 IR（Intermediate Representation），方便用户快速部署。

Chapter 3

鸿蒙生态应用开发能力全景图

- 1）赋能套件
- 2）鸿蒙开发套件
- 3）三方库
- 4）开发者支持平台

围绕开发者旅程，鸿蒙系统为开发者提供了端到端的开发能力支持。如下图所示，鸿蒙系统为开发者提供了赋能套件、鸿蒙开发套件、三方库、开发者支持平台。具体能力全景图如下图所示：

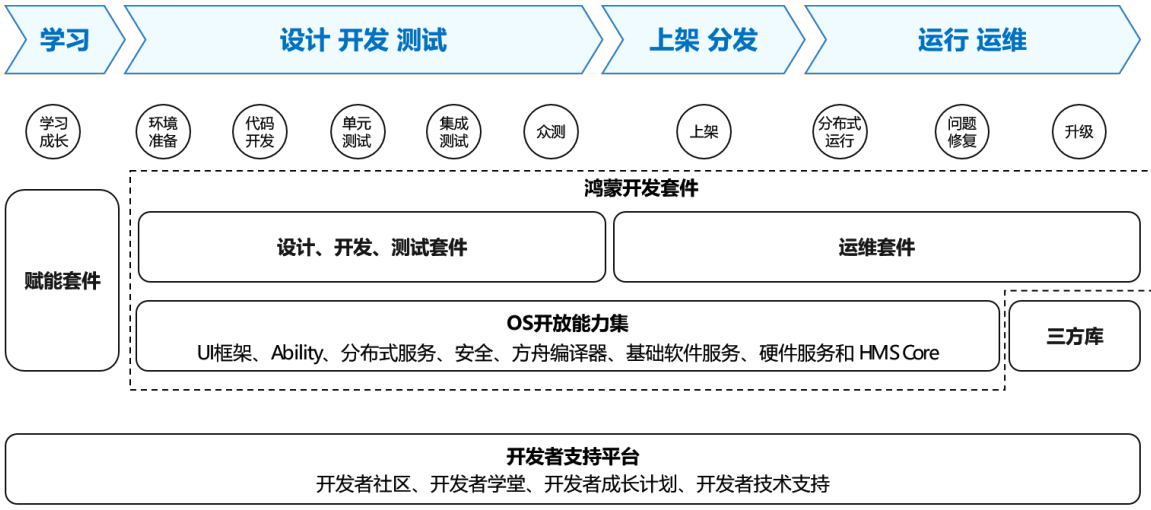

图 3-1：鸿蒙生态应用开发能力全景图

1）赋能套件

开发者了解和学习鸿蒙系统的各类资源，覆盖开发者全旅程，内容包含 Codelabs、视频课程、技术文章、指南、UX 设计资源与指南、API 参考、Sample Code 与 FAQ。

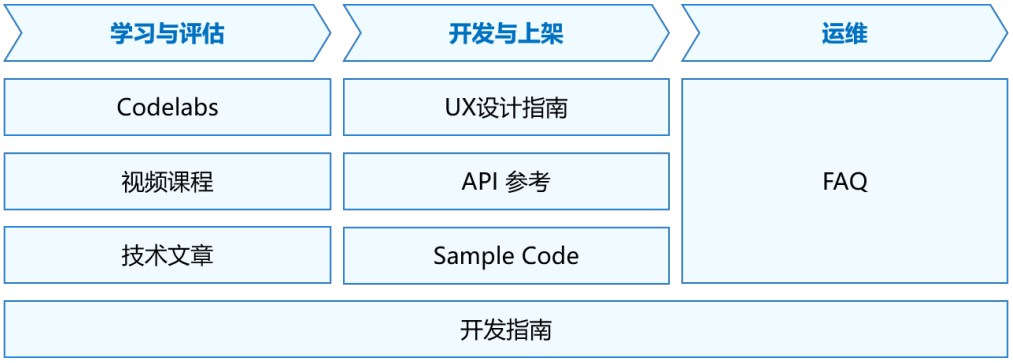

图 3-2：赋能套件全景图

Codelabs：以教学为目的的代码样例及详细的开发指导，帮助开发者一步步地完成指定场景的应用开发并掌握相关知识。Codelabs 将最新的鸿蒙生态应用开发技术与典型场景结合，让开发者快速地掌握开发高质量应用的方法。同时支持互动式操作，通过文字、代码和效果联动为开发者带来更佳的学习体验。

视频课程：基于真实的开发场景，提供向导式学习，多维度融合课程等内容，给开发者提供全新的学习体验。

技术文章：针对新发布特性及热点特性提供详细的技术解析和开发优秀实践。

开发、测试及上架指南：提供系统能力概述、快速入门，用于指导开发者进行场景化的开发。指南涉及到的知识点包括必要的背景知识、符合开发者实际开发场景的操作任务流（开发流程、开发步骤、调测验证）以及常见问题等。

UX 设计资源与指南：提供开发鸿蒙生态应用所需的 UX 设计规范、指导文档以及推荐的设计资源，满足各种场景的设计要求，可以帮助开发者设计出体验一致的鸿蒙生态应用。

API 参考：面向开发者提供鸿蒙系统开放接口的全集，供开发者了解具体接口使用方法。API 参考详细地描述了每个接口的功能、使用限制、参数名、参数类型、参数含义、取值范围、权限、注意事项、错误码及返回值等。

Sample Code：面向不同类型的开发者提供的鸿蒙生态应用开发优秀实践，每个 Sample Code 都是一个可运行的工程，为开发者提供实例化的代码参考。

FAQ：开发者常见问题的总结，开发者可以通过 FAQ 更高效地解决常见问题。FAQ 会持续刷新，及时呈现最新的常见问题。

具体的内容请访问 https://developer.harmonyos.com/ 获取。

2）鸿蒙开发套件

鸿蒙开发套件包含设计、开发、测试、运维套件以及 OS 开放能力集。通过鸿蒙开发套件，开发者可以高效开发鸿蒙生态应用、原子化服务。

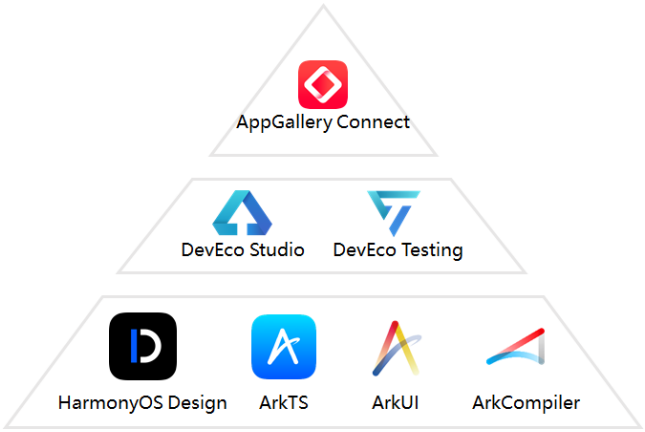

图 3-3：鸿蒙开发套件全景图

设计套件

HarmonyOS Design 是面向万物互联的设计系统，为用户带来全新交互体验。其秉承万物归一，和谐共生，衍生万物的设计理念。设计套件涵盖全面的全场景设计规范、丰富的设计资源，以及设计工具，帮助开发者提升设计和开发效率。

开发套件

开发者在应用开发过程中使用到的产品集合，包含 DevEco Studio 以及 DevEco Studio 集成的性能调优、设备模拟、命令行工具和 SDK。

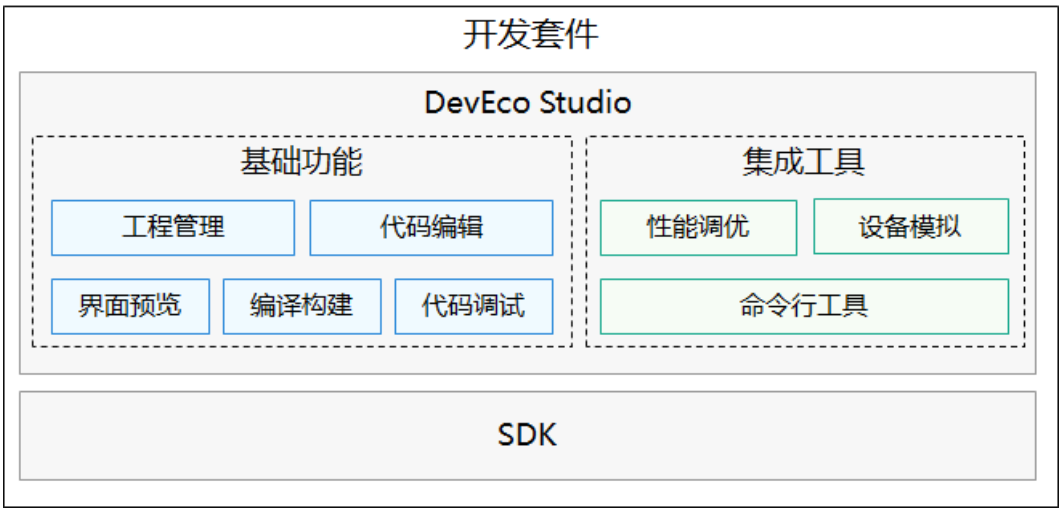

图 3-4：开发套件全景图

DevEco Studio：鸿蒙生态应用、原子化服务开发配套的集成开发环境（IDE），提供了工程管理、代码编辑、界面预览、编译构建、代码调试等基础功能，同时还集成了性能调优工具、设备模拟工具、命令行工具等帮助开发者解决特定领域的问题。

SDK：集成在 DevEco Studio 中，包含开发者可以使用的 API 定义以及调试编译等基础的工具链。

请访问 https://developer.harmonyos.com/cn/develop/deveco-studio 获取最新的 DevEco Studio 以及 SDK。

测试套件

包括测试标准和测试工具两部分。

1. 测试标准

覆盖鸿蒙生态应用性能、功耗、稳定性、兼容性、UX、安全、流转、游戏等测试规范，帮助开发者解决测什么的问题。

表 3-1：测试标准覆盖范围

测试标准名称	测试标准覆盖范围说明
性能测试	任务启动时间、界面刷新帧率、应用内存占用及 CPU 占用等。
功耗测试	后台长驻任务/托管任务场景功耗、后台硬件器件资源/软件系统资源占用场景功耗及分布式资源占用场景功耗。
稳定性测试	长时间运行故障率（App Crash/App Freeze）、长时间运行内存泄漏及长时间运行踩内存等异常场景。
兼容性测试	应用和 OS 兼容、应用升级兼容、应用交互兼容及应用分布式兼容。
UX 测试	应用 UX 规范一致性、控件截断、文字截断、布局变形、图片模糊、黑边及白块等 UX 显示异常。
安全测试	基础安全、用户隐私、权限管理及跨设备安全。
流转测试	流转交互一致性、跨端迁移功能及多端协同功能。
游戏测试	游戏音效、消息免打扰、帧率、屏幕点时延、GPU 使用率及后壳温度等。

2. 测试工具

提供鸿蒙生态应用开发、调试、单元测试、集成测试、上架测试等各开发阶段所需的测试工具集，支持手机、折叠屏、平板、智慧屏、手表、音箱等 1+8+N 设备，帮助开发者全面高效测试。

图 3-5：鸿蒙生态应用测试工具概览

典型测试工具能力简介见下表：

表 3-2：典型测试工具能力简介

测试能力名称	简介
专项测试套件	覆盖性能、功耗、稳定性、兼容性、UX、安全共 6 项专项自动化测试用例，基于测试标准实现自动化，并以测试服务化方式提供。
测试框架	包括单元测试框架和 UI 测试框架， 并支持 ArkTS 语言，单元测试框架提供支撑用例运行的基础能力，UI 测试框架提供 UI 控件查找、点击、检视、按键注入等模拟用户操作的 API。

测试能力名称	简介
性能测试工具	应用性能能效调优和测试工具平台；提供测试过程中实时采集 FPS、RAM、CPU、GPU 等性能数据的采集能力，同时提供性能能效数据分析可视化分析等能力。
稳定性测试工具	应用 UI 随机压测工具，提供 UI 随机事件注入、控件级事件顺序或随机注入、用户操作录制回放、异常日志捕获、可视化报告生成等应用稳定性测试基础能力。
分布式设备录制回放	支持多设备多模输入操作录制，生成基于控件的脚本，开发者添加检查点后形成自动化测试用例脚本，快速自动执行。
应用与服务体检	本地速测工具，支持兼容性、设计约束、性能等专项测试，无需编写用例；集成于 DevEco Studio 一键式测试。
云测平台	云测平台提供性能、功耗、稳定性、兼容性、UX、安全自动化测试能力，支持流转、服务卡片等鸿蒙系统关键特征自动化测试，支持华为 1+8+N 多设备运行。

测试套件获取途径如下表：

表 3-3：测试套件获取途径

名称	获取途径
测试套件	华为官网访问路径： https://developer.harmonyos.com/cn/docs/documentation/doc-guides/app-testing-overview-0000001198515507 社区官网访问路径： https://gitee.com/openharmony/docs/tree/master/zh-

名称	获取途径
	cn/application-dev/application-test

运维套件

主要包括由 HUAWEI AppGallery Connect 提供的上架分发测试和运维分析两大能力。

1. 上架分发测试能力

提供多种上架分发测试能力，满足开发者在不同阶段的上架分发测试诉求。具体如下表介绍：

表 3-4：上架分发测试能力介绍

分发阶段	简介
云测试/调试	快速获取目标机型，便捷远程测试，零脚本、低成本，通过自动化测试快速发现应用的兼容性、性能、稳定性、功耗、安全等问题，出具详细报告，复现与修复应用问题。
开放式测试	可以让开发者的应用在正式发布给所有用户前，面向特定用户群组发布测试版本。参与测试的用户可以向开发者反馈，帮助开发者及时发现技术问题或用户体验问题，以在应用/服务正式上架前完成改进，从而在此过程中最大限度地降低对用户的影响。
全网上架	开发者在开发测试验证完成后，正式提交应用上架申请，审核人员审核通过后应用就会变为“已上架”状态，用户可在设备上搜索到该应用/服务。

分发阶段	简介
分阶段发布	在当前上架版本为全网发布时，开发者可以采用分阶段发布的方式进行升级。采用分阶段发布，可以先向一定比例的用户发布更新的版本，然后再逐步提升用户比例，最终实现全网发布。通过小范围的版本更新，可以快速获取用户对新版本的反馈意见，降低全网发布后版本出现问题的风险。

2. 运维分析

提供崩溃服务、性能管理及云服务监控，支撑开发者精准定位问题，同时支持多维度分析，智能诊断问题并给出解决方案。

表 3-5：运维分析能力介绍

能力名称	简介
崩溃服务	帮助开发者快速发现、定位、解决应用崩溃（又称闪退）问题。无需开发任何代码，即可实时查看可视化数据报告并检测到应用在每个设备上的运行状态，及时快速发现或者定位、解决应用崩溃问题，从而确保应用稳定运行，避免崩溃给用户带来糟糕体验。
性能管理	性能管理（APM，App Performance Management）服务提供分钟级应用性能监控能力，检测应用在每个设备上的运行性能数据，帮助开发者快速发现、定位、解决应用性能问题。
云服务监控	云服务监控是面向云函数、云数据库等云服务的质量监控解

能力名称	简介
	决方案，帮助开发者快速发现、定位、解决云服务的业务层性能问题。

OS 开放能力集

OS 开放能力集通过 SDK 的形式对开发者呈现，提供应用开发所需的一系列系统开放能力，包括 UI 框架、Ability、分布式服务、安全、方舟编译器、HMS Core、基础软件服务、硬件服务等。

1. ArkUI 框架

基于 ArkTS 语言 UI 框架，有类 Web 开发和声明式两种开发范式。其中声明式开发范式采用更接近自然语义的编程方式，开发者可以直观地描述 UI 界面，无需关心框架如何实现 UI 绘制和渲染，实现极简高效开发。UI 框架从组件、动效和状态管理三个维度来提供 UI 能力，同时还提供了系统能力接口，实现系统能力的极简调用。

- 开箱即用的组件：提供丰富的系统预置组件，可以通过链式调用的方式设置组件的呈现效果。开发者可以组合预置组件为自定义组件，通过这种方式将页面组件转化为一个个独立的 UI 单元，实现页面不同单元的独立创建、开发和复用。
- 丰富的动效接口：提供多种绘制图形能力，同时开放了丰富的动效接口，开发者可以通过封装的物理模型或者调用动画能力接口实现自定义动画。

- 多维度状态与数据管理：状态与数据管理作为声明式开发范式的特色，不同的装饰器给开发者提供了清晰的页面更新渲染流程和管道。状态管理包括组件和应用状态管理，合理使用两种状态管理机制，可以在不同场景中仅通过改变数据，自动刷新UI。
- 系统能力接口：封装了丰富的系统能力接口，开发者可以通过简单的接口调用，实现高效的业务开发。

2. Ability

应用所具备能力的抽象，是应用程序的基本组成部分，主要包括组件生命周期回调、系统环境变化通知、应用跳转、卡片开发等能力。具备如下核心的技术特征：

- 基于 MVVM（Model-View-ViewModel）模型：充分结合 ArkUI 的声明式 UI 特性，应用更易于实现界面与逻辑解耦。
- 原生分布式：自带分布式接口，支持跨端迁移和多端协同。
- 支持多设备：Ability 实体与窗口实体解耦，可扩展窗口形态，适应不同的设备形态。

3. 分布式服务

由于具体场景的差异，以及技术演进、生态构建等各种复杂因素的存在，多设备之间的通信方式各不相同（比如 Wi-Fi、蓝牙、ETH、PLC、NFC、USB、Zigbee、红外、超声波等），为了使开发者可以方便快捷地进行分布式开发，鸿蒙系统将跨设备交互涉及的关键能力统一封装并开放给开发者。根据不同的使用场景，分布式服务具体可分为：

- 分布式软总线：支持多设备间协同实现异构组网，对开发者屏蔽不同通信介质和通信协议带来的差异。分布式软总线通过软硬件协同，提供高吞吐、低时延、高可靠、安全可信的通信通道，克服无线通信不可靠、不稳定的挑战，为开发者提供接近本地化访问效果的通信能力。
- 分布式文件：基于分布式软总线，为应用跨设备文件相互访问提供完整解决方案。分布式文件系统是一个直接构建在内核态，无中心、高性能的文件系统。提供了标准的 POSIX 操作能力，使开发者可以像使用本地文件一样访问分布式文件系统。
- 分布式数据库：基于分布式软总线实现数据的分布式管理。用户数据不再与单一物理设备绑定，跨设备的数据处理如同本地数据处理一样方便快捷，对外呈现一份全局唯一的数据视图，让开发者能够轻松处理多设备下的数据存储、共享和访问。
- 分布式硬件：打破单一设备的硬件边界，是软件定义各种新产品形态和体验的“新硬件”所需的关键技术。分布式硬件能够将硬件设备化整为零，形成“超级终端”硬件资源池，供多个设备共享使用，真正达到软件定义硬件、设备间实现系统级融合并灵活按需适应不同场景的目的。
- 融合感知：构筑在传感器技术和多模感知技术上的系统开放能力，提供了对设备、环境、用户等的各类状态的感知能力，例如设备的摆放姿态、操作行为、多设备间的空间拓扑关系、用户的移动状态、位置信息，甚至是用户的情绪等。融合感知目前提供了六种不同的感知能力：

表 3-6：融合感知能力

分类	描述
空间感知	多设备空间感知模块，基于综合传感处理平台的空间感知原子算法能力，提供多设备间的空间感知结果，包括设备间距离、角度、方位关系等。
移动状态感知	移动感知模块，通过加速度传感器、陀螺仪、磁力计、气压计等传感器及 Modem 基站信息，判断用户所处移动状态：乘车、步行、乘坐直梯、自行车、走、跑、静止、快步走、高铁、步行、电梯、相对静止、手持步行、躺卧、智能飞行、飞机、车载(不包含地铁、火车、高铁等轨道交通和自行车之外的交通工具)、停留、地铁。
设备状态感知	设备状态服务，通过加速度传感器、陀螺仪、接近光等传感器及 Modem 基站、WIFI 扫描信息，判断用户手机状态：高精度静止、中精度静止等。
手势感知	能够感知操作手势，比如拿起、翻转、靠近耳朵、摇一摇、旋转、口袋模式、拿离耳朵、落腕、抬腕、招一招。
地理围栏	地理围栏服务，负责围栏监控及围栏进出事件上报。
时间线	时间线服务，根据移动感知的信息，区分家和公司的场景，保存一天内用户停留和移动的行为，用于智慧画像。

4. 安全

鸿蒙系统提供严格的隐私保护和数据安全系统能力，保护消费者智能终端安全。具体的安全隐私能力包括：

- 设备互信认证服务：为保证分布式系统的连接安全，实现用户数据在分布式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确可信，即设备和设备之间建立信任关系，并能够在验证信任关系后搭建安全的连接通道，实现用户数据的安全传输。设备之间的信任关系包括同帐号设备之间的可信关系，以及点对点绑定的设备可信关系。
- 用户身份认证：鸿蒙系统除提供数字密码、图形密码的传统身份认证方式，还提供指纹识别、人脸识别等生物认证手段。根据不同认证方式的安全能力和特点，可应用于相应的身份认证场景，如设备解锁、应用锁，移动支付等。同时，针对分布式业务场景，为提升用户认证的便捷性，鸿蒙系统提供分布式协同认证能力，使用户可便捷地以近端设备为入口完成用户身份认证。
- 应用程序隔离和权限管理：权限访问控制是基于 Access Token 构建的应用权限管理能力，系统化地规范应用程序的行为准则与权限许可并强制执行。由于应用通过沙箱机制彼此隔离，默认情况下，应用只能访问有限的系统资源。但应用为了扩展功能，需要访问沙箱外的系统或其它应用的数据或能力，系统或应用也需要具备共享数据或能力。为了保证这些数据或能力不被不当或恶意使用，Access Token 权限管理机制提供了程序操作某种对象的许可，在应用层面，使用显式定义且经用户授权的权限控制机制。
- 数据分级访问控制架构：为用户数据提供了全生命周期的安全防护措施，确保在每一个阶段，数据都能获得与其个人数据敏感程度、系统数据重要程度和应用程序数据资产价值匹配的保护措施。数据创建时即指定数据分级标签，基于标签关联全生命周期的访问控制权限和策略。在数据存储时，基于不同分级标签，采取不同的加

密措施。在数据传输时，高敏感等级的数据禁止向低安全能力的设备上传递，同时禁止低安全能力的设备发出指令控制高敏感等级的资源和外设。

- 数据防泄露保护：数据生命周期范围内，数据的存储、访问和传输过程中数据泄漏风险比较大。数据防泄漏保护服务保证数据跨设备传输到另一设备后，依然具有相应的访问管控能力。

5. 方舟编译器（ArkCompiler）

支持多种编程语言、多种芯片平台联合编译、运行而设计的统一编译运行时平台。支持包括动态类型和静态类型语言在内的多种编程语言，如 ArkTS、JS、TS。

方舟编译器是鸿蒙系统作为手机、PC、平板、电视、车机和智能穿戴等多种设备统一操作系统的编译运行时底座。主要分成两个部分，编译工具链与运行时。编译工具链以 ArkTS/TS/JS 源码作为输入，将其编译生成为 ABC（ArkCompiler Bytecode，即方舟字节码）文件。运行时直接运行字节码文件，实现对应语言规范的语义逻辑。架构图如下图所示：

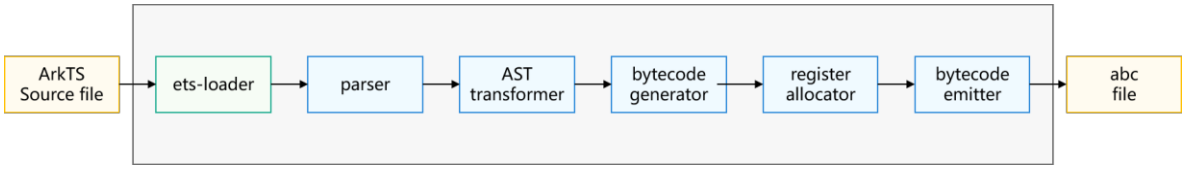

图 3-6：方舟编译器

方舟编译器具备如下核心技术特点：

- 原生支持类型：目前业界引擎执行 TS 的方式是先把 TS 转化为 JS，再运行 JS 源码来完成对应的语义逻辑。方舟编译器的编译工具链编译 TS 源码时，会分析推导 TS 的类型信息并将其传递给运行时。运行时直接使用类型信息在运行前预生成内联缓

存（Inline Cache）以加速字节码执行。另外，TSAOT（Ahead-of-Time）Compiler，可以利用字节码文件中的类型信息，直接编译生成优化机器码，使得应用可以直接运行优化机器码，获得高性能运行体验。

- 并发模型优化与并发 API：ECMAScript 规范没有提供并发语义表述，业界引擎，如浏览器或者 Node.js，通常会提供基于 Actor 并发模型的 WorkerAPI 来支持多线程开发。Actor 模型下执行体之间不共享任何数据对象，通过消息机制进行通信。因此 Web 引擎或者 Node.js 引擎的 Worker 都有启动速度慢、内存占用高这些问题。针对这些问题，方舟编译器运行时已经实现了 Actor 实例中的不可变或者不易变的对象（方法和字节码）的共享，较大程度地优化了 Actor 的启动性能和启动内存。
- 简洁的并发 API：方舟编译运行时不只提供了业界通用的 Worker API，还提供了 TaskPool，作为并发 API 的增强。TaskPool 是一个支持优先级调度、工作线程自动扩缩容的任务池功能库。开发者无需关心并发实例的生命周期，也无需关心任务负载变化时需要创建或者销毁并发实例，极大地简化了高性能多线程鸿蒙应用的开发。
- 安全：方舟编译器前端编译工具链将 ArkTS/TS/JS 程序预先静态编译为方舟字节码，并且还提供了多重混淆能力的增强，有效地提升了开发者代码资产的安全强度。同时出于安全的考虑，ArkCompiler 不支持 sloppy 模式的 JS 代码，也不支持 eval 等运行动态字符串的功能。

6. 基础软件服务

鸿蒙系统为开发者提供了通用的基础软件服务，包括多媒体、通信、图形、文件存储、升级、无障碍等，下面介绍三个常用的服务。

- 多媒体服务是指用户听觉、视觉信息的表达、存储和还原的处理过程。多媒体服务确保在不同设备上运行的性能和体验，同时也为应用开发者提供统一的接口，让开发者更多专注于业务开发，轻松使用多媒体资源。

  ✧ 相机服务：提供精确控制相机镜头，采集视觉信息的能力。

  ✧ 视频服务：提供听觉和视觉信息的解压播放和压缩录制的能力。

  ✧ 音频服务：提供音频播放、音频采集、音量管理和短音播放的能力。

  ✧ 图片服务：提供单张图片信息解压还原和压缩的能力。

  ✧ 数据服务：提供音频文件、视频文件、图片文件等数据高效管理能力。

- 通信服务为各种各样的终端设备提供多样信息的传递，鸿蒙系统提供业界主流和常见的通信方式，涵盖短距离的无线通信（NFC、蓝牙和 WLAN 等）、长距离的蜂窝通信，以及有线的以太网通信等服务，并且为其提供网络管理服务。

  ✧ WLAN 服务：提供 WLAN 基础功能、P2P（peer-to-peer）功能和 WLAN 消息通知的相应服务，让应用可以通过 WLAN 和其他设备互联互通。

  ✧ 传统蓝牙：提供蓝牙版本 3.0 以下的传统蓝牙服务。

✧ 低功耗蓝牙：提供蓝牙版本 4.0 以上的低功耗蓝牙服务。

✧ NFC 服务：提供近距离的、非接触式识别和互联技术，让移动设备、消费类电子产品、PC 和智能设备之间可以进行近距离无线通信。

✧ 电话服务：提供无线蜂窝网络通信和 SIM 卡管理服务。

✧ 网络管理服务：提供数据连接管理、流量统计和网络协议栈服务。

- 图形服务提供图形渲染与显示输出的功能，内部通过对系统硬件资源的合理利用，为系统提供流畅高效的显示体验。图形系统按功能维度分为渲染服务、绘制、动画、效果、显示与内存管理、2D 图形库和 3D 图形引擎这几个子模块。

✧ 绘制：提供高性能的 2D 渲染服务。

✧ 动画：提供轻量的、链式的、物理连续的动画实现。

✧ 效果：提供高性能，基于物理的多类型的动效能力。

✧ 渲染服务：提供应用的界面显示（包括控件、动效等 UI 元素），将不同应用渲染的图层进行合成，送显的过程。

✧ 显示与内存管理：提供了硬件合成、送显、Vsync 以及显示设备、Surface、Bufferqueue 轮转、本地平台化窗口等能力。

✧ 2D 图形库：提供 2D 渲染库 SKIA 和轻量并行渲染等能力。

- ✧ 3D 图形引擎：提供 3D 图形场景管理、渲染系统、插件平台和物理引擎等能力。

7. 硬件服务

硬件服务通过屏蔽硬件厂家接口差异，为应用提供统一的接口规范，从而使开发者便捷地控制外设，其中包括电源、USB、泛 sensor 和位置服务等，下面介绍两个常用的服务。

- 泛 sensor 服务是应用访问底层硬件传感器的一种设备抽象概念。根据鸿蒙系统提供的 Sensor API 查询设备上支持的传感器类型，并订阅指定传感器的数据，通过定制相应的算法，开发各类应用，比如指南针、运动健康、游戏等。
  - ✧ 订阅：提供数据订阅能力，系统将获取到的传感器数据上报给订阅者。
  - ✧ 控制：提供设置传感器的数据采样间隔和数据上报间隔等控制能力。
  - ✧ 服务管理：提供各类传感器列表的查询管理能力。
  - ✧ 数据上报：提供数据接收、解析及分发的能力。
  - ✧ 权限管控：提供传感器权限管控能力。
  - ✧ 维测：提供传感器的调试和打点能力。
- 位置服务（LBS，Location Based Services）又称定位服务，是由移动通信网络和卫星定位系统结合在一起提供的一种增值业务，通过一组定位技术获得移动终端的位置信息（如经纬度坐标数据），提供给移动用户本人或他人以及通信系统。

✧ 全球导航卫星系统定位：提供 GNSS 定位服务的全能力，包含 GNSS 定位请求管理、GNSS 芯片参数设置、GNSS 芯片参数上报订阅、GNSS 缓存等功能。

✧ 网络定位：提供基站、WLAN 和蓝牙等网络定位框架能力。

✧ 地理编码：提供地理编码转换服务框架能力。

✧ 被动定位：提供被动定位功能，允许应用进行被动定位，该种类型定位不会主动发起 GNSS 定位或者网络定位，只有当其他应用有 GNSS 或者网络定位请求时，被动定位的应用才会收到位置信息。

✧ 地理围栏：提供地理围栏的能力，支持围栏添加和删除、围栏事件订阅和取消、围栏事件主动通知等功能。

✧ 区域管理：提供国家码、城市码、区域码等信息的检测能力，对外提供国家码查询接口，主动监听国家码等信息变化并通知给应用。

8. HMS Core

华为移动服务开放能力是为鸿蒙生态应用开发提供场景化服务的平台。同时，依托华为云服务，HMS Core 也为这些服务提供云端能力，用于各服务的开通、业务实现及运营。每个服务可单独为开发者提供独立且完善的产品能力，也可以多个服务相互组合，为多个领域提供行业解决方案。下面介绍一些常用的服务：

- 帐号服务：支持用户在手机、平板、智慧屏等设备上，使用华为帐号快速便捷地登录应用。支持用户信息授权、一键授权登录、多帐号切换、家长管控、数字遗产继承等。

- 推送服务：多种推送样式，支持大文本、Inbox多行文本、带按钮等样式，帮助开发者更好地提高消息对用户的吸引力。提供精细化人群划分的能力，可以根据用户属性、特定标签、订阅主题，将用户群进行不同维度的细分，实现精细化运营。
- 应用内支付服务：提供支付服务，让开发者聚焦应用本身能力，更关注于应用创新。大大降低支付渠道、全球化合规等开发引入和产品上线环节的投入，助力商业变现。
- 分析服务：免费的跨平台、多设备数据分析产品，清晰理解用户行为方式，轻松实现数据驱动的精细化运营。
- 广告服务：开发者可以通过流量变现服务在应用内广告中获得更多收益；通过广告标识服务，可以保护用户隐私，也可以帮助广告平台和三方监测平台合规地开展个性化广告和转化归因。

3）三方库

鸿蒙生态三方库，是在鸿蒙系统上可重复使用的软件库，可帮助开发者重用技术资产，快速开发鸿蒙生态应用、原子化服务，提升开发效率。根据不同的开发语言分为两种：

- ArkTS/JS/TS/语言的三方库，可直接导入并使用。
- C/C++语言的三方库，在应用开发中通过 NAPI 的方式来使用。

鸿蒙生态三方库发布与使用完整的流程如下图所示：

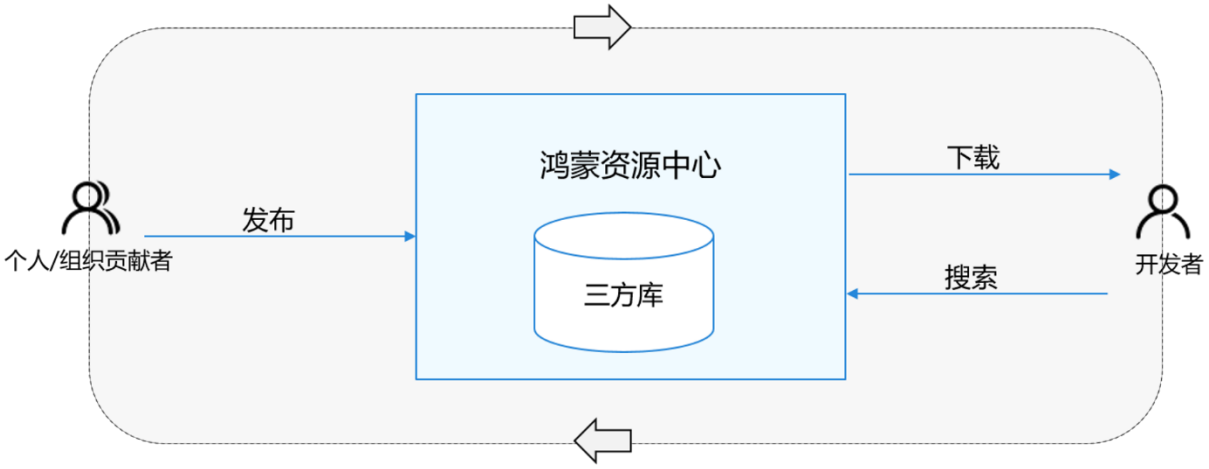

图 3-7：鸿蒙生态三方库管理

开发者资源中心聚合了丰富的鸿蒙生态开发资源包，方便开发者一站式获取所需组件。个人/组织贡献者通过 DevEco Studio 将开发好的三方库发布到资源中心。开发者遵循如下的步骤使用三方库：

1. 应用开发者登录资源中心，通过分类和关键字搜索需要的三方库信息；

2. 应用开发者在应用开发时，通过包管理工具，将搜索到的三方库引入到应用依赖清单中。

4）开发者支持平台

为了能更好地连接、服务开发者，开发者支持平台向开发者提供了以下能力，帮助开发者快速成长并融入鸿蒙生态。

- 开发者社区：开发者技术交流平台，帮助开发者探索开发实践、交流心得经验、获悉业界动态、答疑解惑。

- 开发者学堂：聚合官方鸿蒙生态课程，课程有慕课、微课、直播课、训练营等多种形式，内容有入门、基础、进阶分级，面向开发者提供学、练、考、证一站式服务，满足开发者不同阶段的学习诉求。
- 开发者成长计划：设置校园开发者计划（HSD）、布道师计划（HDE）等开发者成长计划，助力不同类型开发者提升技能，帮助开发者相互连接、共享能力、彼此启发、协同激励。
- 开发者技术支持：针对开发者在开发实践中的具体问题提供问题反馈平台，其中智能客服提供 7*24 小时自助式智慧技术问答；工单系统由技术客服为开发者提供问题处理和在线技术支持。

Chapter 4

高效开发与测试

- 1）典型开发场景
- 2）设计
- 3）ArkTS 语言
- 4）ArkUI 框架
- 5）用户程序框架
- 6）SDK
- 7）集成开发环境
- 8）测试工具

1）典型开发场景

基于鸿蒙系统原生开发框架，开发者可以根据自己实际的业务情况选择：

- 独立开发一个应用
- 独立开发一个原子化服务
- 同时开发应用和原子化服务

开发者可以选择开发简单、场景聚焦的原子化服务，渐进迭代演进，按需组合原子化服务成为一个复杂的应用。对于大型游戏类应用，可以直接开发鸿蒙生态应用，针对智慧屏、车机、手表等设备，可以考虑开发原子化服务并提供卡片，让信息更加直接的呈现给用户。

2）设计

HarmonyOS Design 支持跨设备的超级终端一拖即连，万能卡片轻轻一滑即可获取所需信息，文件中转站、智慧视觉等创新功能，带来全场景智慧生活新体验。

HarmonyOS Design 涵盖全面的全场景设计规范，丰富的设计资源，以及设计工具，帮助开发者提升开发效率：

- 全面的全场景设计规范：包括设计理念、人因研究、应用架构、人机交互、视觉风格、动效、声音、多态控件、界面用语、全球化、无障碍、隐私设计等。
- 丰富的设计资源：提供上千种的图标资源，媒体音效专项分类，快速开发调用；字体再升级，支持新版国标汉字完整覆盖 GB 18030—2022 实现级别 2 的汉字。

- 高效的设计工具：动态响应式布局，六种动态布局能力，控件元素自由组合，无缝适配多尺寸界面。首创自适应 UI 引擎，自动学习优化布局 ，提升开发效率与实现效果；提供支持手机、平板、折叠屏、智慧屏、智能座舱等多设备多品类的响应式布局模板，支撑快速设计开发。

3）ArkTS 语言

ArkTS 是鸿蒙生态应用的开发语言。它在 TypeScript（简称 TS）的基础上，提供了声明式 UI、状态管理等相应的能力，让开发者以更简洁、更自然的方式开发高性能应用。TS 是 JavaScript（简称 JS）的超集，而基于 TS 的 ArkTS 会结合应用开发和运行的需求持续演进，包括引入分布式开发范式、并行和并发能力增强、类型系统增强等方面的语言特性。

目前 ArkTS 针对 TS 的扩展主要是从 UI 框架的需求角度，在 TS 基础上做了进一步的扩展，包括定义了各种装饰器、自定义组件和 UI 描述机制，再配合 UI 框架中的 UI 内置组件、事件方法、属性方法等共同构成了应用开发的主体。

在应用开发中，除了 UI 的结构化描述之外，还有一个重要的方面是状态管理。ArkUI 是基于 ArkTS 的 UI 框架，基于 ArkTS 提供的扩展语法， ArkUI 框架中提供了多维度的状态管理机制。例如 UI 相关联的数据，不仅可以在组件内使用，还可以在不同组件层级间传递，比如父子组件之间，爷孙组件之间，也可以是全局范围内的传递，还可以是跨设备传递。另外，从数据的传递形式来看，可分为只读的单向传递和可变更的双向传递。开发者可以灵活地利用这些能力来实现数据和 UI 的联动。

总体而言，ArkTS 通过扩展成熟语言、结合语法糖支持 ArkUI 提供简洁高效的声明式开发范式，再结合 UI 组件、状态管理等方面设计，统一鸿蒙生态应用的开发范式。

4）ArkUI 框架

ArkUI 是鸿蒙生态原生的 UI 开发框架。主体结构如下图所示：

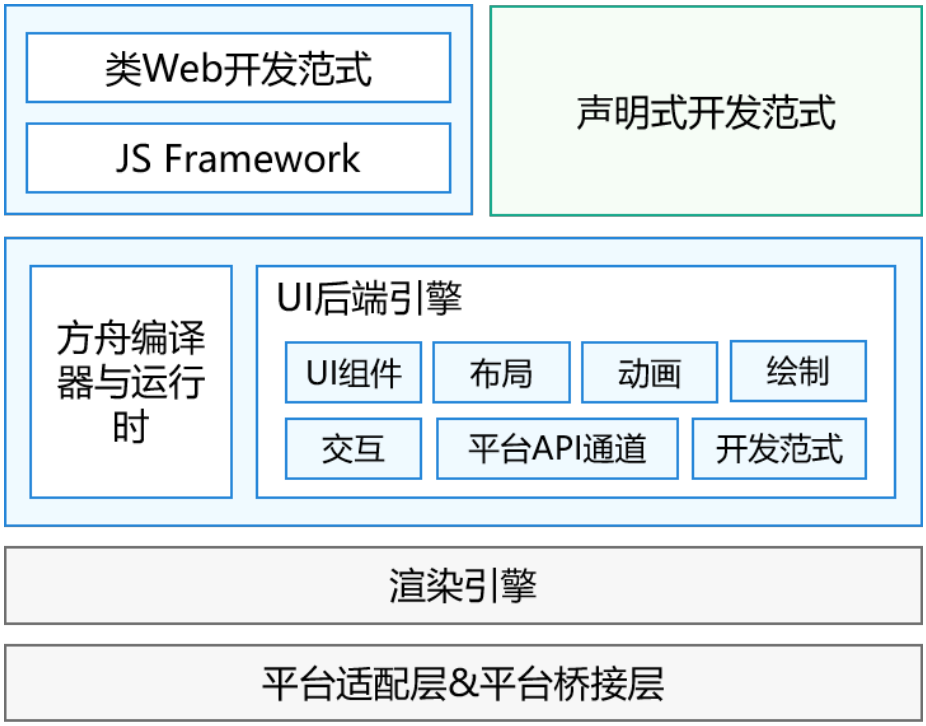

图 4-1：ArkUI 框架组成结构

ArkUI 框架提供给开发者的主要能力如下：

- 组件：多种开箱即用的 UI 组件，如文本显示、图片显示、按键等。
- 布局：多种布局方式，如弹性布局、列表、宫格、栅格布局等。
- 动画：包含了属性动画、转场动画和自定义动画能力。
- 绘制：包含了多种绘制能力，支持图形绘制、颜色填充、文本绘制、图片绘制等。

- 交互：提供了多种交互能力，应用在不同平台通过不同输入设备均可正常进行 UI 交互响应。
- 平台 API 通道：提供 API 扩展机制，通过此种机制进行封装风格统一的 JS 接口。
- 开发范式：分别是基于 JS 扩展的类 Web 开发范式和基于 ArkTS 的声明式开发范式。两种范式会长期共存，类 Web 开发范式对于 Web 及前端开发者更友好，声明式开发范式开发更加简洁高效。

类 Web 开发范式

使用 HML 标签文件进行布局搭建，使用 CSS 文件进行样式描述，使用 JS 文件进行逻辑处理。UI 组件与数据之间通过单向数据绑定的方式建立关联，当数据发生变化时，UI 界面自动触发更新。此种开发范式，对 Web 前端开发者更为友好。

类 Web 范式的整体接口采用与传统 Web 页面开发相似的设计理念，采用 HML 、CSS 与 JS 三种类型的文件进行页面开发，开发者可以基于此范式方便地进行 UI 构建，同时提供数据绑定机制，支持通过 JS 进行数据更新，进而更新 UI。

- HML 语法：是一套类 HTML 的标记语言，通过组件、事件构建出页面的内容。页面具备数据绑定、事件绑定、列表渲染、条件渲染和逻辑控制等高级能力。在 HML 文件中不仅可以进行架构描述，也可以进行数据绑定，通过{{}}方式进行数据绑定后，也需要在 JS 文件中进行数据的定义，运行时将使用 JS 文件中提供的数据 content 进行替换。

- CSS 语法：CSS 是描述 HML 页面结构的样式语言。所有组件均存在系统默认样式，也可在页面 CSS 样式文件中对组件、页面自定义不同的样式。ArkUI 开发框架提供标准 CSS 语法的核心功能集，满足应用开发者的诉求。
- JS 语法：在类 Web 开发范式中，提供了一系列的全局方法与全局对象，进行数据操作与逻辑处理。

框架后端采用 C++开发语言实现，提升了框架的运行性能，使用方舟编译器运行时作为 JS 引擎，具有更优的 JS 执行性能，同时还提供了一套完整的包含 UI 组件、布局机制、动画能力的渲染框架，通过渲染引擎对 UI 元素进行绘制。

类 Web 范式实现层面采用了轻量化设计的思路，将 JS Framework 下沉到 C++层，以减小 JS 的内存占用，使用 C++进行更为严格的内存分配管理，并采用更为轻量的 JS 引擎，UI 部分采用轻量的 UIKit 并结合轻量图形引擎最终实现百 K 级别设备的支持，从而在轻量化设备上可执行的应用，也可以在硬件规格更高的设备上执行，而无需重新开发。这也就是采用类 Web 开发范式的优势所在，采用统一的开发范式，开发者无需关心具体运行时的前端框架、JS 引擎与后端 UI 组件，系统会根据运行平台不同，采用最佳的模块，保障应用在不同平台都可具有最佳的运行性能。具体的实现原理如下图所示：

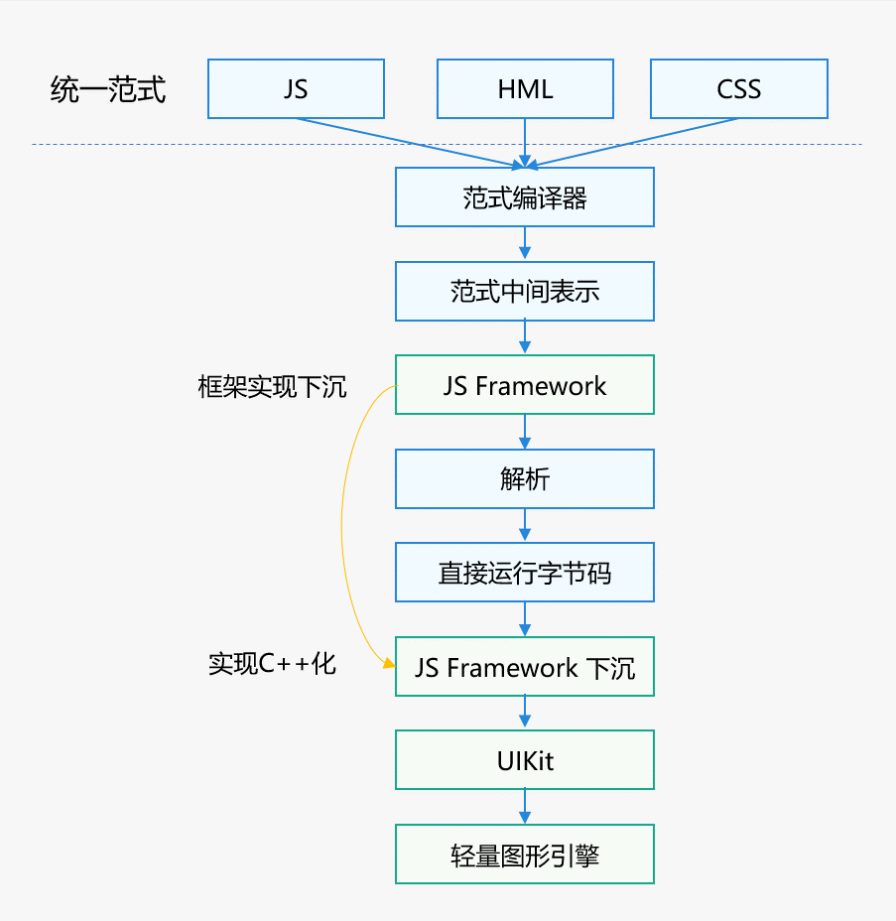

图 4-2：类 Web 开发范式

声明式开发范式

声明式开发范式与类 Web 范式采用相同的后端，通过语言增强、架构精简等手段，在功能和性能方面对比类 Web 开发范式有了全面提升。采用声明式开发范式进行应用开发，相同场景下，对比类 Web 开发范式代码更为精简，并且在性能、内存方面进一步优化提升。下面针对重点功能进行详细介绍说明。

1. 状态管理

声明式开发范式的核心思想是数据驱动 UI 变化，通过提供的状态进行数据管理，这里状态管理指的是，管理数据发生变化时，UI 组件更新的范围，如下状态图表达了 UI 框架的状态管理手段。

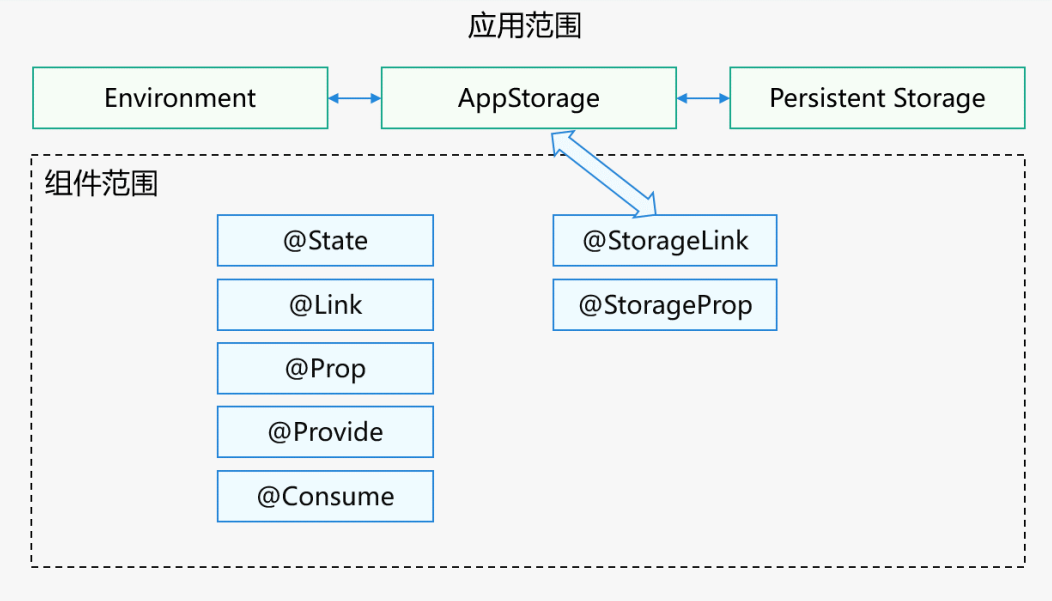

图 4-3：状态管理

状态管理从生效范围的维度可以分为应用范围和组件范围。应用范围的数据是以 AppStorage 为中心进行管理，根据不同的使用场景分为提供系统环境数据管理的 Environment，提供持久化存储支持的 Persistent Storage。组件范围的数据通过装饰器的方式提供管理机制，称为状态变量装饰器：

- @State：组件拥有的状态属性，当@State 装饰的变量更改时，组件会重新渲染更新 UI。
- @Link：组件依赖于其父组件拥有的某些状态属性，当任何一个组件中的数据更新时，另一个组件的状态都会更新，父子组件重新渲染。

- @Prop：类似@Link，但子组件所做的更改不会同步到父组件上，属于单向传递。
- @Provide：作为数据的提供方，可以更新其子孙节点的数据，并触发页面渲染。
- @Consume：在感知到@Provide 数据更新后，会触发当前自定义组件的重新渲染。

同时为了使应用数据变化能够触发组件的更新，基于 AppStorage 还提供了两个状态变量装饰器：

- @StorageLink 装饰器：组件通过使用@StorageLink(key)装饰的状态变量，与 AppStorage 建立双向数据绑定，key 为 AppStorage 中的属性键值。当创建包含@StorageLink 的状态变量的组件时，该状态变量的值将使用 AppStorage 中的值进行初始化。在 UI 组件中对@StorageLink 的状态变量所做的更改将同步到 AppStorage，并从 AppStorage 同步到任何其他绑定实例中，如 PersistentStorage 或其他绑定的 UI 组件。
- @StorageProp 装饰器：组件通过使用@StorageProp(key)装饰的状态变量，将与 AppStorage 建立单向数据绑定，key 标识 AppStorage 中的属性键值。当创建包含@StoageProp 的状态变量的组件时，该状态变量的值将使用 AppStorage 中的值进行初始化。AppStorage 中的属性值的更改会导致绑定的 UI 组件进行状态更新。

2. 组件化

组件是 ArkUI 框架中的基础显示单元，一切 UI 显示的内容都是组件，由框架直接提供的称为预置组件，由开发者定义的称为自定义组件，其具有以下特点：

- 可组合：允许开发人员组合使用内置组件、其他组件、公共属性和方法；

- 可重用：自定义组件可以被其他组件重用，并作为不同的实例在不同的父组件或容器中使用；
- 配置化生命周期回调：生命周期的回调方法可以在组件中配置，用于业务逻辑处理；
- 数据驱动更新：由状态变量的数据驱动，实现 UI 自动更新。

3. UI 元素装饰器

使用自定义组件的场景中，通常会遇到需要动态传入不同的 UI 元素的情况，为了满足该场景 ArkUI 框架同时提供了动态构建 UI 元素的能力。

- @Builder：可通过@Builder 装饰器进行描述，该装饰器可以修饰一个函数，此函数可以在 build 函数之外声明，并在 build 函数中或其他@Builder 修饰的函数中使用，在一个自定义组件内快速生成多个布局内容。
- @Style：声明式范式为了避免开发者对重复样式的设置，通过@Styles 装饰器可以将多条样式设置提炼成一个方法，直接在组件声明的位置使用。@Styles 装饰器将新的属性函数添加到基本组件上，如 Text、Column、Button 等，当前@Styles 仅支持通用属性。通过@Styles 装饰器可以快速定义并复用组件的自定义样式。@Styles 可以定义在组件内或组件外，在组件外定义时需在方法前添加 function 关键字，组件内定义时不需要添加 function 关键字。
- @Extend：为了满足开发者拓展原生组件的诉求，提供了@Extend 装饰器，可以将新的属性函数添加到内置组件上，如 Text、Column、Button 等。通过@Extend 装饰器可以快速地扩展原生组件。

4. 动效

声明式范式中一大特点体现在动效的使用上，与传统开发方式不同，声明式的动画是由数据变化驱动动画启动，而不再是直接控制动画的播放。UI 框架根据开发者的配置，自动地进行动画执行，根据动画场景不同进行如下分类：

- 属性动画：组件的某些通用属性变化时，可以通过属性动画实现渐变效果，提升用户体验。
- 显式动画：全局 animateTo 显式动画接口，指定由于闭包代码导致的状态变化插入过渡动效，开发者可以在事件回调中通过显式动画对指定数据变化增加动画效果。
- 转场动画：转场动画包括页面间转场、组件内过渡转场和共享元素转场三种，通过路由接口进行页面路由时，会触发动画的执行。

5. 事件交互

ArkUI 框架提供了很多交互事件，这些事件提供了不同的信息用于处理相关程序交互逻辑，目前提供了 UI 组件事件以下几类事件：

- UI 组件事件：由 UI 组件内置交互逻辑触发，不同的 UI 组件由不同的 UI 组件事件，比如 TextInput 输入框产生的 onEditChange 输入文本变更事件，List 列表组件产生的 onScrollIndex 列表项滚动事件，这类事件属于非冒泡事件（非冒泡事件指的是当一个组件上的事件被触发后，该事件不会向父节点传递，下同）；挂载卸载事件，当 UI 组件挂载到 UI 组件树或者从 UI 组件树上卸载时触发，典型的场景比如通过 if 渲染语法控制 UI 组件的显隐状态，该事件属于非冒泡事件。

- 交互事件：点击事件，拖拽事件，焦点事件，触摸事件，按键事件，鼠标事件，手势事件等。

6. 绘制能力

ArkUI 框架提供两种 2D 自定义绘制能力。一种是通过图形组合的方式，利用布局、绝对定位和各种图形进行组合实现；另一种是通过绘制 API 在 Canvas 画布上进行绘制。

7. 混合开发

应用的场景是多样的，部分场景直接采用 UI 组件组合无法满足诉求，例如游戏、地图这种需要依赖 C++ SDK 进行独立渲染，又或者开发相机、视频播放器这种需要独立纹理填充的场景，因此需要框架提供一种能够在C++侧进行自定义绘制的组件。ArkUI 框架提供了 XComponent 组件，支持加载应用动态库、NAPI 跨语言调用，进行 C++绘制能力的开发。

可视可说开发

可视可说框架提供“系统级”和“应用级”两种实现方式。其中，“系统级”无需应用适配自动支持标准控件文本的语音操控功能；“应用级”接入方式允许开发者对控件场景、角标、别名、个性化播报等元素进行适配，从而提供最佳的用户体验。应用级和系统级两种实现是互补关系，应用级优化用户体验，系统级保证覆盖率。

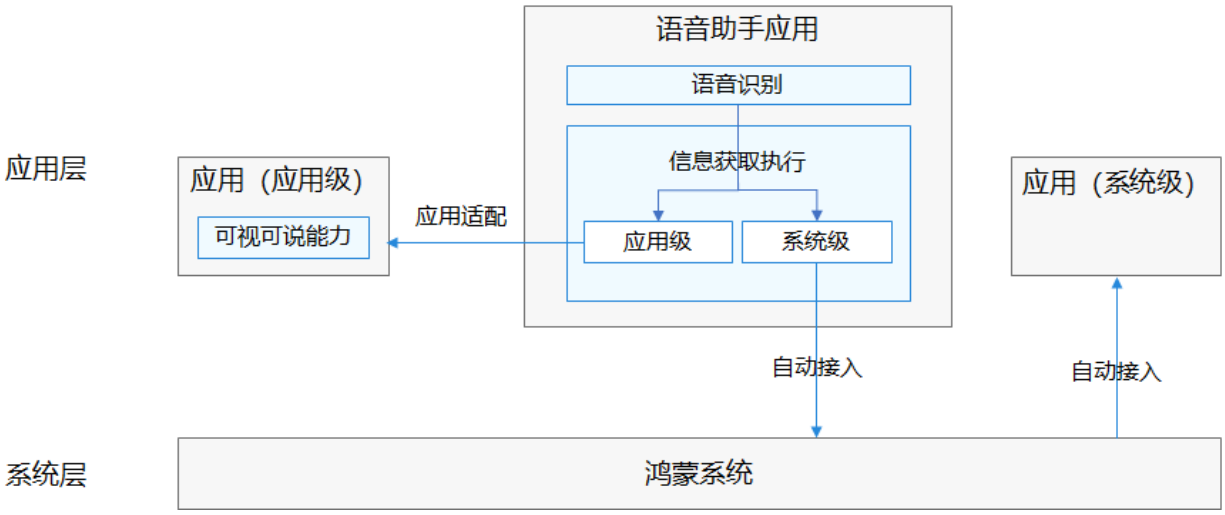

图 4-4：可视可说框架

1. 系统级

应用使用标准控件开发，无需额外适配，自动识别标准控件界面文本和位置执行，天然支持界面文本可视可说基础体验。

2. 应用级

系统级基础体验无法满足体验目标时，应用可以按照业务特征进行灵活定制适配，以此获得可视可说最佳体验。

3. 语音交互生命周期

可视可说分为信息获取和识别执行，信息获取模块基于界面变化用户监听界面变化获取信息热词，识别执行模块将信息热词传递到语音系统进行 AI 识别。

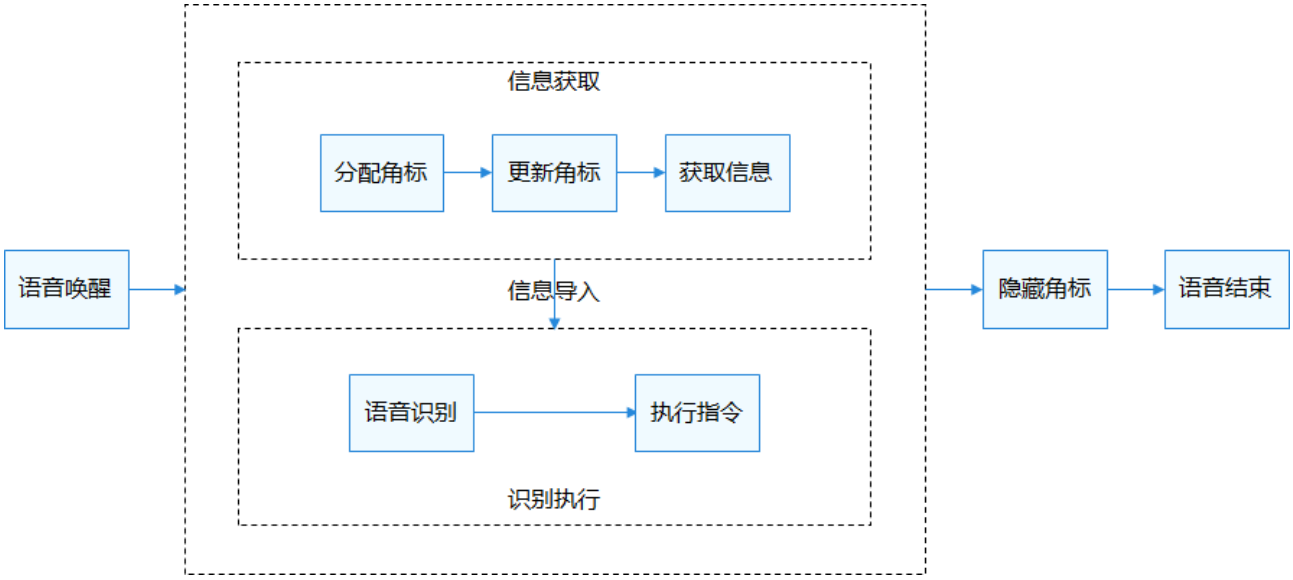

图 4-5：单次语音交互生命周期

- 语音唤醒：语音助手唤醒后，发送建立连接给前台应用，应用收到请求后启动可视可说初始化。
- 信息获取
  - ✧ 分配角标（可选）：应用反馈界面需要展示的角标数量给智慧语音。
  - ✧ 更新角标（可选）：界面变化时，自行管理展示角标刷新。
  - ✧ 获取信息：智慧语音获取界面热词信息，包括文本，图标和角标信息。
- 识别执行
  - ✧ 语音识别：智慧语音系统根据用户语音和界面信息热词，识别出可视可说意图和指令。
  - ✧ 执行指令：收到可视可说意图和指令触发执行。
- 隐藏角标：通知应用隐藏界面角标（隐藏角标动作也由适配应用自己完成）。

- 语音结束：语音助手释放可视可说，发送解除绑定给前台应用，应用收到解除连接后停止可视可说相关动作（比如角标未隐藏则隐藏角标信息）。

5）用户程序框架

用户程序框架是操作系统对应用程序架构的抽象，通过它可以完成鸿蒙应用、原子化服务的编写。在鸿蒙系统中，Ability 是应用最基本的抽象单位，是能够完成一个独立功能的应用组件，Ability 可能有用户界面，也可能没有用户界面仅执行后台功能，这由其具体的子类型决定。由于鸿蒙系统的历史发展，系统中提供了两种编程模型，无论是哪种模型，都是以 Ability 作为基础类型：

- FA 模型：自鸿蒙系统早期版本就有的模型，适合简单应用。Ability 衍生出了三个子类型的 Ability：
  - ✧ PageAbility：负责用户界面与用户交互。
  - ✧ ServiceAbility：负责后台服务。
  - ✧ DataAbiltity：负责数据存储。
- Stage 模型：自 API9 新增的模型。面向复杂场景，是目前主推并且今后会长期演进的模型。

Stage 模型

Stage 模型的设计基于如下三个出发点：

- 规范化后台进程管理：为了保障用户体验，鸿蒙系统上的运行环境对后台应用进行了有序治理，应用程序不能随意驻留在后台。系统定义了四类后台任务：

  ✧ 短时任务：对于所有应用，在退到后台之后，系统提供了一个短期的可运行时间，在这个时候，应用可以进行数据保存的操作。

  ✧ 长时任务：对于音乐播放，投屏，导航这类场景，系统提供了长时任务的能力。需要注意的是，长时任务的具体类型也是由系统明确定义的，应用应当根据实际需要来申请，而不应当过度滥用。

  ✧ 延时任务：对于一些实时性要求不高的场景，系统提供了延时任务。这种情况下，会由系统统一周期，对齐多个应用的任务激活时间。

  ✧ 托管任务：除此之外，系统还提供了托管任务，这类任务是由系统完成，完成之后再通知到应用。例如，下载，提醒和定位这类场景。

- 原生支持组件级的跨端迁移和多端协同：鸿蒙系统是原生支持分布式的操作系统，应用框架需要从架构设计上使得组件更易于实现迁移和协同。Stage 模型通过 Ability 与 UI 分离，以及 UI 展示与服务能力合一等模型特性，实现这一设计目标。
- 支持多设备和多窗口形态：为了支持多种设备形态和更易于实现多种不同的窗口形态，需要组件管理服务和窗口管理服务在架构层面上是解耦的，从而方便裁剪，更有利于定制不同的窗口形态。Stage 模型通过重新定义 Ability 生命周期定义，设计组件管理服务和窗口管理服务的单向依赖来解决这一问题。

Stage 模型与 FA 模型最大的区别在于：Stage 模型中多个应用组件共享同一个虚拟机，而 FA 模型中，每个应用组件独享一个虚拟机。因此在 Stage 模型中，组件之间可以方便的共享对象和状态。Stage 模型作为鸿蒙系统的主要应用编程模型，开发者通过它能够更加便利地开发出分布式场景下的复杂应用。其主要优势包括：

- 为复杂应用设计：多应用组件在运行时共享同一个虚拟机引擎，从而减少复杂应用运行内存的占用。采用面向对象的开发方式，使得复杂应用代码可读性高、易维护好、可扩展性强。
- 程序逻辑与用户界面解耦：窗口部分可单独销毁和重建，窗口与 Ability 可跨设备运行，Ability 可在不启动界面的情况下响应请求。
- 开放的扩展能力点：支持卡片、输入法、快捷开关、分享、壁纸、长时任务等应用开发。

与 FA 模型相同的是 Stage 模型中的应用组件也是由 Ability 这个基础概念演化而来。在 Stage 模型中，有两类 Ability：

- UIAbility：负责用户界面和用户交互。
- ExtensionAbility：负责 UIAbility 之外的事情。事实上，ExtensionAbility 有很多的具体类型，例如：FormExtension。

下图展示了 Stage 模型的主要结构：

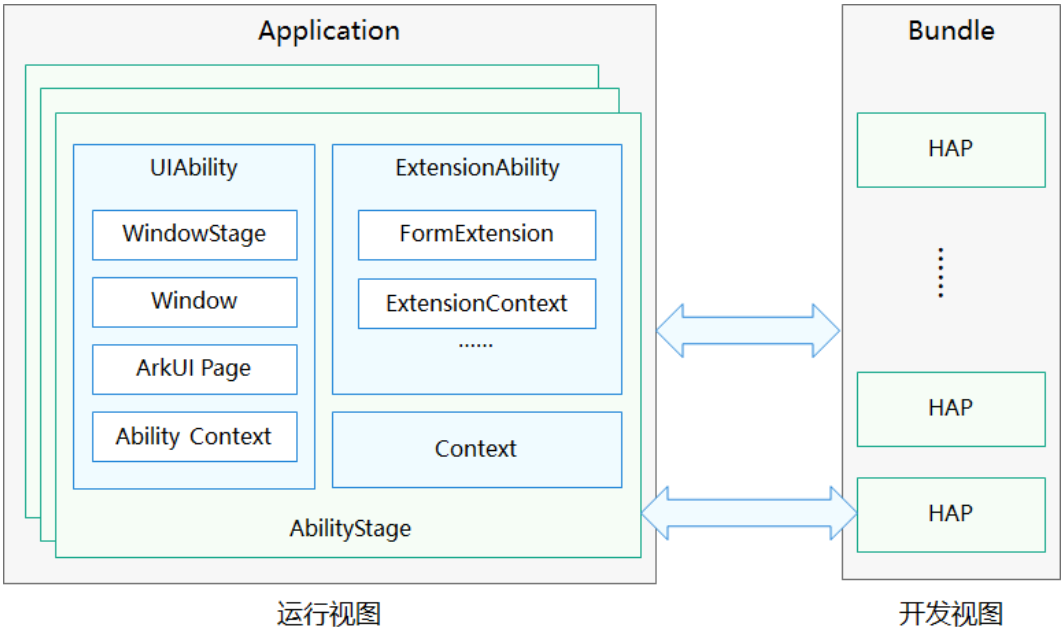

图 4-6：Stage 模型 Ability 相关概念

6）SDK

SDK 包含了应用开发所需的 API 定义和基础工具集。

ArkTS API

为了使得鸿蒙生态能够覆盖尽可能多的设备，鸿蒙系统使用 ArkTS 作为主要的应用开发语言。鸿蒙系统提供的每个接口都属于一个特定的版本，例如：API Version 8，API Version 9 等。每个 API 版本包含了若干新增的 API，也可能会出现 API 的行为变更以及少量的废弃 API。为了生态的一致性，开发者应当尽可能使用最新的 API 版本进行开发。已经上架的应用也应当定期地进行 API 版本的升级。

系统在设计上会保证 API 的质量和兼容性，关于 API 的详细定义，请参见 ArkTS API 参考。

C API

虽然鸿蒙生态应用的主要开发语言是ArkTS，但同时也提供使用Native语言开发ArkTS模块的扩展方式，鸿蒙系统中支持这种开发方式的C语言接口叫C API。C API也包含在鸿蒙SDK中，方便开发者使用C或者C++语言实现应用相应功能。

C API只覆盖了部分鸿蒙基础底层能力，如libc，图形库，窗口系统，多媒体，压缩库等，并没有完全提供类似于ArkTS API上的完整鸿蒙平台能力，开发者可以使用C API开发支持鸿蒙应用框架的扩展动态库，通过import语句导入到ArkTS环境中使用。

建议使用C API的场景：应用性能敏感场景，如游戏，物理模拟等计算密集型场景；复用已有的C或C++库场景；需要针对CPU特性进行专项定制场景，如neon加速等。

C API接口组成如下表所示：

表4-1：C API接口组成

接口分类	接口功能
标准C库	以鸿蒙C库为基础提供的标准C库接口，当前包含了1500+的接口
标准C++库	C++运行时库，应用使用C++语言，需打包到应用中
hilog	日志打印接口
N-API	应用框架提供的，方便应用开发接入ArkTS语言环境的一组类Node的C接口

接口分类	接口功能
XComponent	应用框架 XComponent 组件中的 surface 与触屏事件接口，供开发者开发高性能图形应用使用
libuv	应用框架集成的三方的异步 IO 库接口
libz	zlib 库，提供基本的压缩，解压接口
Drawing	系统提供的 2D 图形库，可以在 surface 上进行绘制
OpenGL	系统提供的 OpenGLv3 接口
Rawfile	应用资源访问接口，可以读取应用中打包的各种资源
OpenSLES	用于 2D，3D 音频加速的接口库
包管理	包服务接口，方便查询应用包信息

N-API 接口

N-API 接口提供了使用 C/C++封装操作 ArkTS 对象的能力，使用类 Node 的 N-API 接口命名。开发者使用 C/C++开发业务，通过 N-API 接口实现跨语言调用，方便开发者使用高性能 C 语言能力。开发者开发一个 C/C++的 ArkTS 扩展库后，在 ArkTS 侧可以通过 import 引入这个扩展库。

方舟工具链

传统的 JS 程序开发中，应用程序往往带的是经过前端打包工具处理过的 JS bundle 文件，在程序运行阶段进行解释执行；这种运行方式需要设备有强大的计算能力。鸿蒙系统能够支持的设备范围广泛，覆盖从低端的 IoT 设备到高性能手机设备。采用传统的方式，无法保证多类型设备的体验一致性。

在鸿蒙开发环境中，应用代码是通过前端编译器完成编译的。前端编译器按照语言规范解析源代码，编译成方舟运行时能够理解的二进制字节码格式（ABC，ArkCompiler ByteCode），最后打包到应用中。前端编译器是鸿蒙应用框架与其它 JS 应用框架最主要的差别之一。下图展示了两种编译运行方式的差别，方舟前端工具链把解析源码、编译字节码的过程从运行时迁移到编译时，降低运行时的开销。

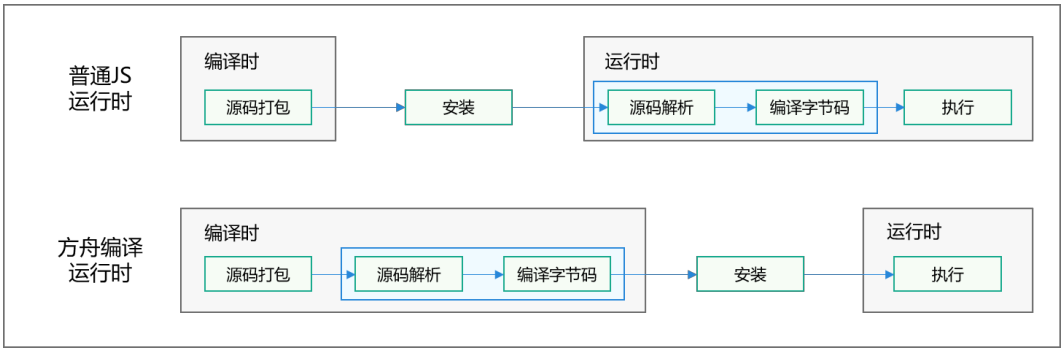

图 4-7：普通 JS 运行时与方舟编译运行时比对图

前端流水线在发起编译时，进行工程参数解析，依赖分析，语法校验，语法转换，代码编译等各个编译动作的编排。前端编译器负责编译流水线中源代码编译，提供对应的触发接口给编译流水线。

下图为鸿蒙生态应用的编译流水线流程：

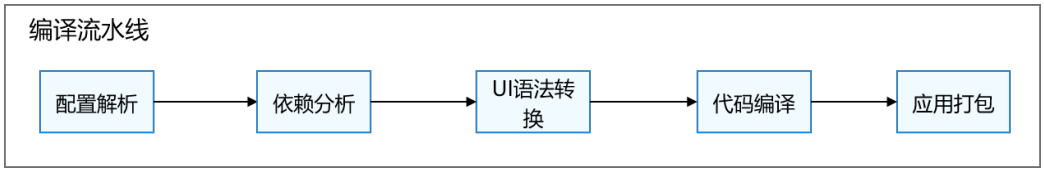

图 4-8：编译流水线

- 配置解析：解析 IDE 工程中的配置文件，解析程序组件，入口组件，组件包名，代码位置等信息
- 依赖分析：根据代码中的 import 等语句，进行各个代码的依赖关系分析
- UI 语法转换：对 UI 中的各种语法糖进行转换
- 代码编译：输入转换后代码，解析编译，输出 ABC 字节码文件
- 应用打包：获取应用的资源，ABC 字节码文件，应用配置文件等，使用用户签名进行应用打包，输出应用包

前端编译器负责将 ArkTS 代码编译成方舟字节码 ABC，鸿蒙生态应用编译流程中，分为两种编译模式。分别是 bundle 和 esmodule 编译模式。两者的区别主要在源码文件的处理上，bundle 编译把各个有依赖关系的源代码通过打包方式打成一个 bundle 文件，然后通过前端编译器编译成 ABC 字节码文件；而 esmodule 编译是保持用户写的 ArkTS 模块不变，通过前端编译器编译成 ABC 字节码文件，字节码文件内保留各个模块的代码段，依赖关系等信息；当前推荐开发者使用 esmodule 模式，保持模块语义。

前端编译器架构

前端编译器是根据输入的 ArkTS 源码，进行词法，语法解析、转换、编译、输出字节码文件；在这个过程中会提取代码中标注的类型信息，进行类型检查，类型绑定，最终作为元数据生成到字节码 ABC 文件中。

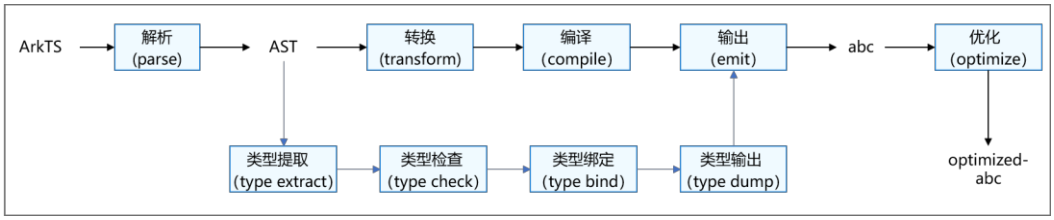

图 4-9：前端编译器架构

- 解析：前端编译器读取 ArkTS 源码，进行词法，语法解析，输出抽象语法树（AST）
- 转换：前端编译器识别语法糖，转换成基础语法
- 编译：根据抽象语法树，生成对应的中间表示（IR）
- 输出：收集 IR，字符资源，常量，等各种元素，按照 ABC 文件格式生成字节码文件
- 优化：读取 ABC 文件中的字节码信息，生成 IR 表示，进行优化处理，重新生成更优的字节码文件。

7）集成开发环境

HUAWEI DevEco Studio 是面向鸿蒙生态的集成开发环境，提供了一站式的鸿蒙生态应用、原子化服务开发能力，详细能力如图所示。

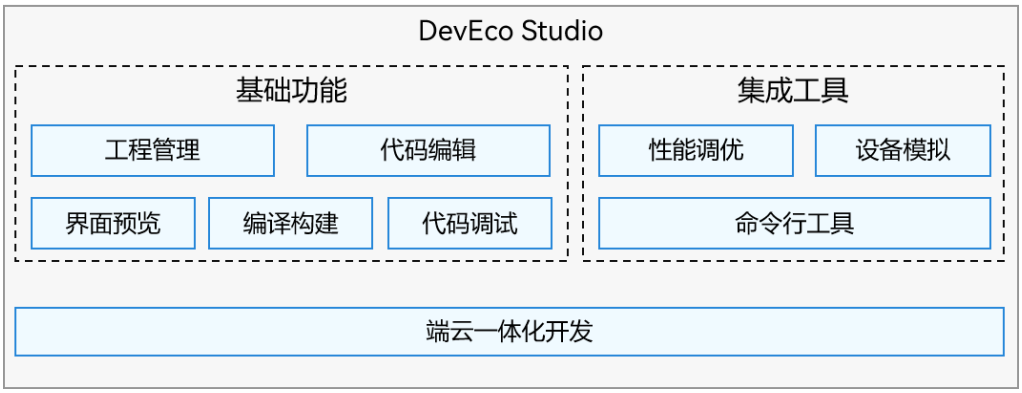

图 4-10：DevEco Studio 集成开发环境

工程管理

提供基础的工程管理能力，包括工程向导，工程模板，鸿蒙视图，SDK 管理，样例导入等，并提供模板市场，支持扩展丰富的模板。开发者可以方便地安装和更新鸿蒙 SDK，利用模板创建鸿蒙生态应用、原子化服务，使用鸿蒙视图聚焦到关键文件及配置，也能导入样例快速学习了解鸿蒙 API 的用法：

- 工程向导：开发者基于模板，方便地创建出工程（Project），应用模块（Module），库模块（Library），Ability，服务卡片（Service Widget）等开发态元素，快速得到鸿蒙生态应用开发所需的项目结构。
- 鸿蒙视图：通过鸿蒙视图，可以过滤掉应用开发中无需特别关注的文件，如工具自动生成的文件，聚焦鸿蒙开发的代码文件及配置文件。
- SDK 管理：开发者可以在 HUAWEI DevEco Studio 和命令行工具上管理鸿蒙 SDK，包括安装，更新，卸载等。开发的时候，如果 SDK 缺失，编译构建也能自动检测缺少的 SDK 并下载。
- 模板市场：模板市场提供了丰富的工程模板，支持模板的发布及更新，HUAWEI DevEco Studio 可以检测到新版本并更新。开发者也可以通过模板市场分享自己开发的工程模板，供其他开发者下载使用。
- 样例导入：样例提供了常用鸿蒙 API 的使用指导，开发者可以将样例工程导入到 HUAWEI DevEco Studio，学习常用 API 的使用，也可以基于样例工程快速开始开发。

代码编辑

针对 ArkTS 语言及 ArkUI 框架，HUAWEI DevEco Studio 提供了代码补全、跳转、校验、重构、高亮、折叠、格式化等一系列编辑功能，辅助开发者便捷地阅读代码，高效地编写代码，实时地纠正代码错误。相较于传统的代码编辑，HUAWEI DevEco Studio 还结合了人工智能技术，根据待补全位置的上下文代码特征进行预测和推荐，使补全项更精准，推荐内容更完整，开发人员可以更快速地完成鸿蒙生态应用、原子化服务开发。同时，HUAWEI DevEco Studio 内置鸿蒙生态应用、原子化服务开发最佳编程规范校验功能，实时提示代码错误，支持快速纠错，可高效地将建议修复结果应用于代码中。

界面预览

在开发过程中，开发者需频繁修改 UI 代码，查看对应的呈现效果，确保开发与实现目标一致。传统的开发模式下，开发者每次修改代码后，执行编译构建，并推送应用到设备上重新运行，才能查看到界面的呈现效果，整个过程冗长，产生极大的时间浪费。HUAWEI DevEco Studio 提供了界面预览能力，使开发者更方便快速地调测应用界面，大幅提升界面开发效率。

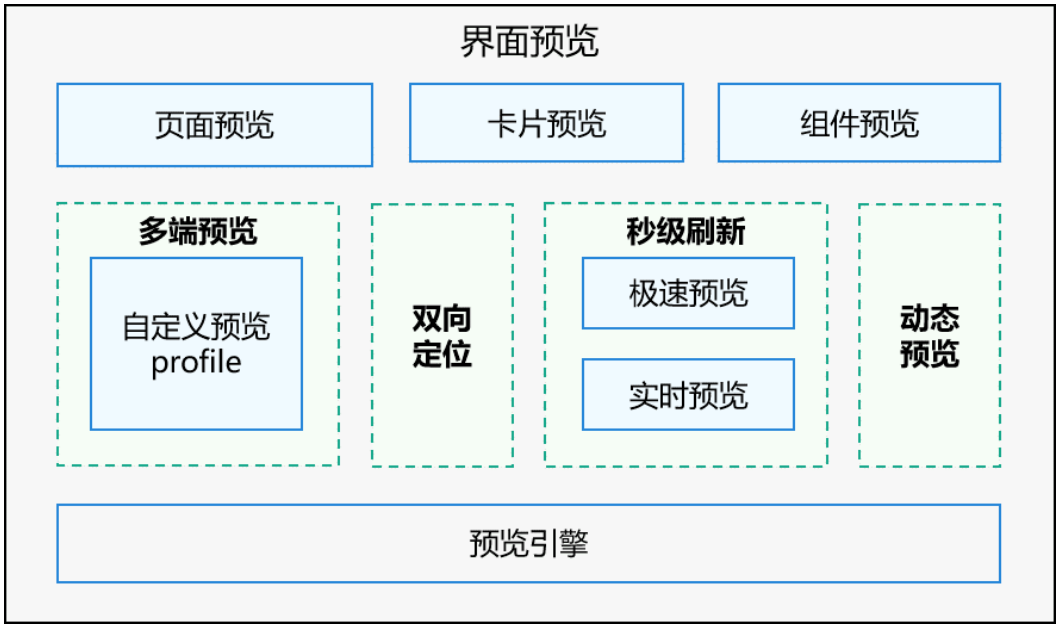

图 4-11：界面预览全景

- 页面预览：可快速查看应用/服务中页面 UI 代码的呈现效果。
- 卡片预览：可查看多种卡片规格、多种卡片尺寸（最小/标准/最大）的呈现效果。
- 组件预览：可独立查看组件的呈现效果，支持开发者注入组件参数，灵活查看组件在不同上下文中的预览效果。
- 自定义预览 profile：预览 profile 是设备显示能力的抽象定义，典型的 profile 信息有设备名称、 设备类型、屏幕分辨率、屏幕密度、语言、亮暗模式、横竖屏状态等。通过自定义预览 profile，开发者能自由组合设备显示能力，查看 UI 代码在不同设备上的呈现效果。
- 双向定位：支持 UI 代码和预览效果的双向定位修改。
- 极速预览：对于组件属性的调整，无需保存和编译，极速呈现预览效果。
- 实时预览：开发者修改 UI 代码，保存后可秒级刷新预览界面，界面调测效率倍增。

- 动态预览：在预览界面中提供点击、滑动、键盘输入等交互能力，操作体验与真机设备一致。

编译构建

HUAWEI DevEco Hvigor 是一款华为自研轻量级编译构建工具，将编译操作进行任务化管理，为开发者提供自动化的构建服务。其具备强大的构建能力，支持多种语言（ArkTS、C/C++等）、多种文件（低代码描述文件、资源文件等）的快速编译，最终生成 HAP/App 包。此外，Hvigor 还具备以下特点：

- 高效编译：充分利用系统资源，并行执行编译请求，提升编译效率；综合历史信息，精确增量检查，高度复用往次构建产物，缩短编译时长；最优化编排任务序列，异步化执行编译操作，减小等待间隙，加速构建流程。
- 差异构建：内置多目标构建机制，允许开发者灵活选择源码文件、资源信息、部署设备等应用要素，形成多种组合。通过简易配置，匹配自定义构建目标，一键打包生成不同产物，实现“一套代码，多种产物”。
- 灵活扩展：支持开发者自定义编写构建任务，匹配自身业务需求，扩展编译构建流程。
- 独立运行：拥有完善的命令行工具，兼备良好的跨平台能力，可以脱离 HUAWEI DevEco Studio，独立运行在 Windows、Mac、Linux 等操作系统上，支持不同环境下的流水线搭建。

代码调试

在开发过程中，代码调试是使用频率最高的功能之一，开发者可以使用断点跟踪或日志分析，快速定位代码缺陷。HUAWEI DevEco Studio 提供了常用的代码调试功能，如设置断点（普通断点、条件断点、异常断点、符号断点等）、断点跳转（Step Over/Into/Out）、变量值查询、表达式计算、调试堆栈、命令行工具等。此外，基于鸿蒙系统的特点，还提供了以下四点功能，进一步提升效率：

- 分布式调试：分布式调试用于定位跨设备交互场景下的代码缺陷。使用跨设备的断点调试，可以使断点在不同设备的代码间跳转，自动寻找目标设备并建立调试会话。通过查看跨设备的调试堆栈，可以快速准确地跟踪设备交互的详细情况，包括每次流转的起点（设备、函数栈）、终点（设备、函数栈）和详细交互数据。
- 跨语言调试：支持 ArkTS 和 C/C++两种语言同时调试，并支持断点从 ArkTS 语言跳转到被调用的 C/C++语言；提供统一的调试堆栈，便于快速查阅两种语言代码的调用层次关系，整体操作体验与单一语言调试一致。
- Hot Reload：修改代码后，无需重新创建调试会话和启动鸿蒙生态应用、原子化服务，即时生效，大幅缩短调试时间。
- 多维日志：查看系统消息日志时，可根据设备、进程、日志级别以及自定义的规则灵活过滤，快速筛选，协助定位代码缺陷。在分布式场景下，可以同时查看多个设备的系统消息日志。其中特别重要的异常日志在独立的窗口呈现，避免淹没在大量系统消息日志中。

性能调优

应用的运行性能至关重要，一旦出现卡顿、发热、电量消耗过快等问题，便会导致体验急速下降，造成用户流失。性能调优是鸿蒙生态应用开发阶段中非常重要的一环，然而性能优化过程充满挑战，需要开发者了解应用程序框架、系统、硬件各方面知识，并对多维度性能数据进行综合分析。为了降低性能调优技术难度，HUAWEI DevEco Studio 推出了场景化调优工具 DevEco Insight，提供以下四点关键能力：

- 场景化调优模板：针对各类典型场景的性能问题，提炼出对应的场景化调优模板，自动采集相应维度性能数据。常用场景化调优模板如表 4-2 所示。
- 模板自动推荐：根据实时监控观测到的性能异常事件，自动推荐对应的场景化模板。
- 高效数据分析：关联分析不同维度性能数据，结合同一时刻的代码调用栈，快速分析代码和性能问题之间的因果关系。
- 一键定位代码行：分析结果中代码堆栈并一键跳转至编辑器中的对应代码行。

表 4-2：常用场景化调优模板

模板名称	用途
Time Insight	耗时分析模板：通过周期性采集调用栈，识别 CPU 耗时高的热点代码段，用于分析卡顿、CPU 占用高、运行速度慢等问题。
Allocations Insight	内存分析模板：录制和分析内存分配记录和内存堆快照，用于分析内存峰值高，内存泄漏，内存不足导致应用被强杀等问题。
CPU Insight	CPU 分析模板：录制 CPU 调度事件、线程运行状态、CPU 核频率、Trace 等数据，可用于分析卡顿、运行速度慢、应用无响应等

模板名称	用途
	问题。
Energy Insight	能耗分析模板：录制和分析能耗异常事件、硬件资源使用记录、功耗关联的系统状态，可用于分析电量消耗过快的问题。
Launch Insight	启动分析模板：录制和还原从点击应用图标，到显示首帧过程中的CPU、内存等资源使用情况，用于分析启动耗时长的问题。

设备模拟

HUAWEI DevEco Studio 提供了设备模拟的能力，解决鸿蒙生态应用、原子化服务开发过程中遇到的真机设备不足、无分布式应用调试环境等问题，为开发者提供低成本、易获取的调测验证环境。

- 超级终端模拟：支持对手机、智慧屏、手表等多种终端进行模拟，针对不同模拟终端提供了差异化的交互界面，方便开发者快速在多个模拟终端上开发调试应用。此外，开发者可以一键配置分布式组网，组成模拟超级终端，支持对分布式应用进行调试。未来还将支持模拟器与真机组合成超级终端。
- 丰富的器件模拟：提供了多终端常用器件、外设、传感器的模拟，包括电池、Wi-Fi、移动网络、GPS、Camera、陀螺仪、心率等，支持开发者调用模拟器件的能力，进行特定功能的开发。
- 场景化数据注入：通过场景化的数据注入能力，开发者能快速模拟一些常见的设备使用场景，方便调试应用在特定场景下的功能。包括低电量、弱网络信号、摇一摇、GPS 导航、户外跑步运动等场景。

- 统一设备管理：支持对模拟设备、超级终端、本地真机设备、远程真机设备的统一管理，方便开发者快速切换调试设备，一键配置超级终端。

命令行工具

HUAWEI DevEco Studio 提供了一系列命令行工具，辅助开发者更高效的管理 SDK、设备，提升调试、调优的效率。目前提供以下工具供开发者使用：

- sdkmgr：查看、安装和卸载 HarmonyOS SDK。
- hdc：管理设备、本地和设备之间传输文件、安装和卸载应用、启动和终止应用。
- bytrace：对内核 ftrace 进行了封装和扩展，配合应用打点，追踪进程轨迹，分析应用性能。

端云一体化开发

HUAWEI DevEco Studio 在传统的“端开发”基础上新增了“云开发”能力，支持开发鸿蒙生态应用、原子化服务的云侧服务，提供端云一体的开发体验。

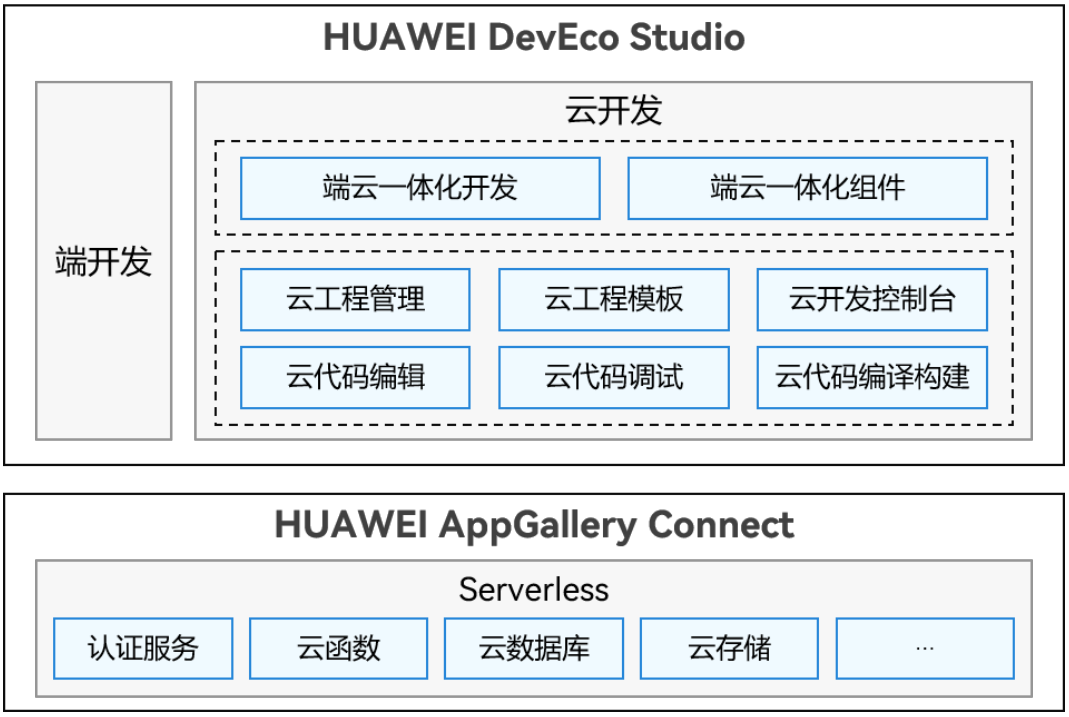

图 4-12：端云一体化开发全景图

- 端云一体化开发：支持端侧代码和云侧代码的协同开发，统一管理端侧和云侧代码目录，进行端云代码的端到端开发、调试和部署。
- 端云一体化组件：内置完整的云侧逻辑，开发者在集成 UI 组件的同时即可自动实现云侧逻辑，快速实现特定场景的功能。
- Serverless：为鸿蒙生态应用、原子化服务的云侧服务提供 Serverless 化托管服务，具有开箱即用、一键部署、自动弹性伸缩、免运维等特点，开发者可聚焦业务逻辑本身，降本增效。

8）测试工具

鸿蒙生态应用、原子化服务的测试分层模型分为：单元测试、集成测试、专项测试。

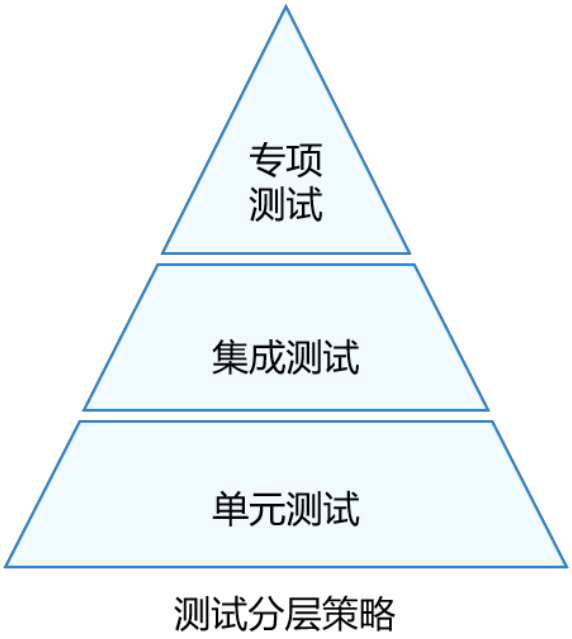

图 4-13：鸿蒙生态应用、原子化服务分层测试模型

单元测试

单元测试是函数级别的验证。函数是产品开发实现的最基本单位，单元测试通过验证产品代码的函数输入输出，最终保证整个产品的质量。

单元测试框架

DevEco Testing hypium 单元测试框架可以在真机或者模拟器上运行。单元测试框架采用插件化机制开发，具备空间占用最小化、功能可定制、语法兼容特点。测试框架的整体结构如下图所示：

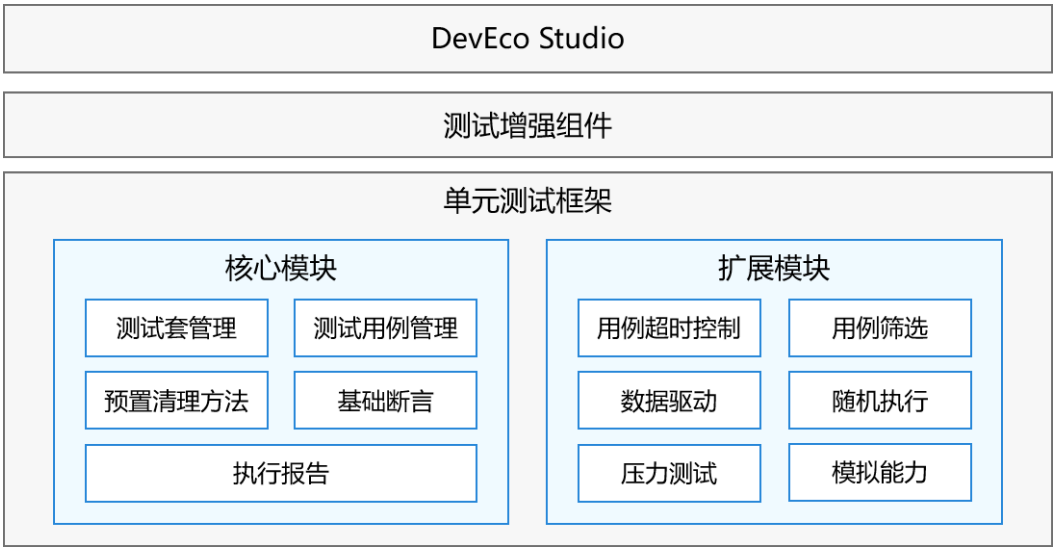

图 4-14：单元测试框架

测试框架由核心模块和扩展模块组成。其中核心模块是测试框架的最小集，包含执行必备核心接口和逻辑。扩展模块是在核心模块的基础上增加一些常用能力，例如用例超时控制、用例筛选、数据驱动、压力测试、随试执行等。核心模块采用插件化机制，提供接入能力和运行时上下文，扩展模块通过插件的方式接入。

DevEco Testing hypium 单元测试框架具备以下特点：

- 语法统一：使用声明式或者类 Web 范式，针对不同的开发范式，单元测试框架提供统一的测试接口。
- 可定制：扩展模块即插即用，开发者根据不同的场景动态组合定制测试框架能力。
- 轻量化：核心模块代码量少，打包编译之后小于 10KB，支持 ROM 资源有限的设备灵活部署。

集成测试

集成测试分为模块测试和特性测试。模块测试把若干个单元组装，发现模块缺陷；特性测试把若干个模块集成，发现特性缺陷。鸿蒙生态为开发者提供多种集成测试的能力，方便开发者针对不同的集成测试场景，快速便捷的进行测试。

1. UI 测试框架

通过简洁易用的 API 提供查找和操作界面控件能力，支持开发者编写基于界面操作的自动化测试脚本。UI 测试框架的整体结构图如下图：

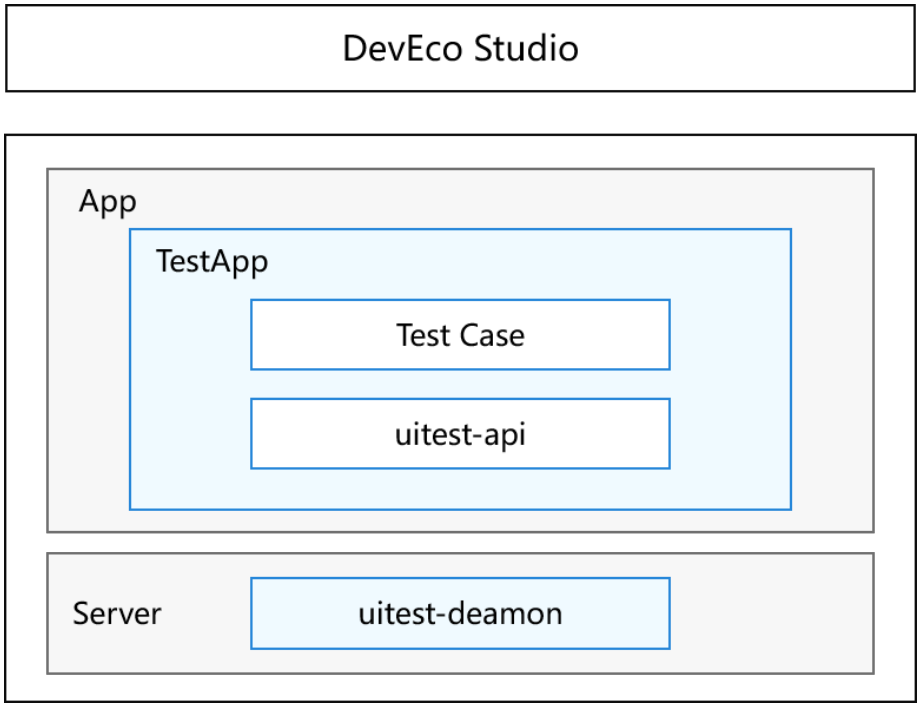

图 4-15：UI 测试框架

整个测试框架组主要分为两个部分，一部分是 uitest-api，用来提供接口，封装 UI 测试场景的 API，另外一部分是 uitest-deamon，是 UI 测试核心模块，对接系统服务，提供控

件树获取、解析、查找、操作能力。uitest-api 提供核心接口类 Driver、On 和 Component，具体功能如下表：

表 4-3：uitest-api 核心类功能列表

类名	功能
Driver	UI 测试能力入口类，实现与设备交互，提供页面控件查找、检查存在性及注入按键能力。
On	页面元素查找条件，用于描述目标控件特性（文本、id、类型等），UiDriver 根据 On 描述控件特征查找控件。
Component	页面元素实体类，返回查找的控件对象，提供控件属性查询，滑动查找等触控和检视能力。

2. 性能测试工具

DevEco Testing SmartPerf 提供一套完整的应用测试和调优工具，为应用开发者提供方便快捷的应用性能测试手段。主要包括三个部件：

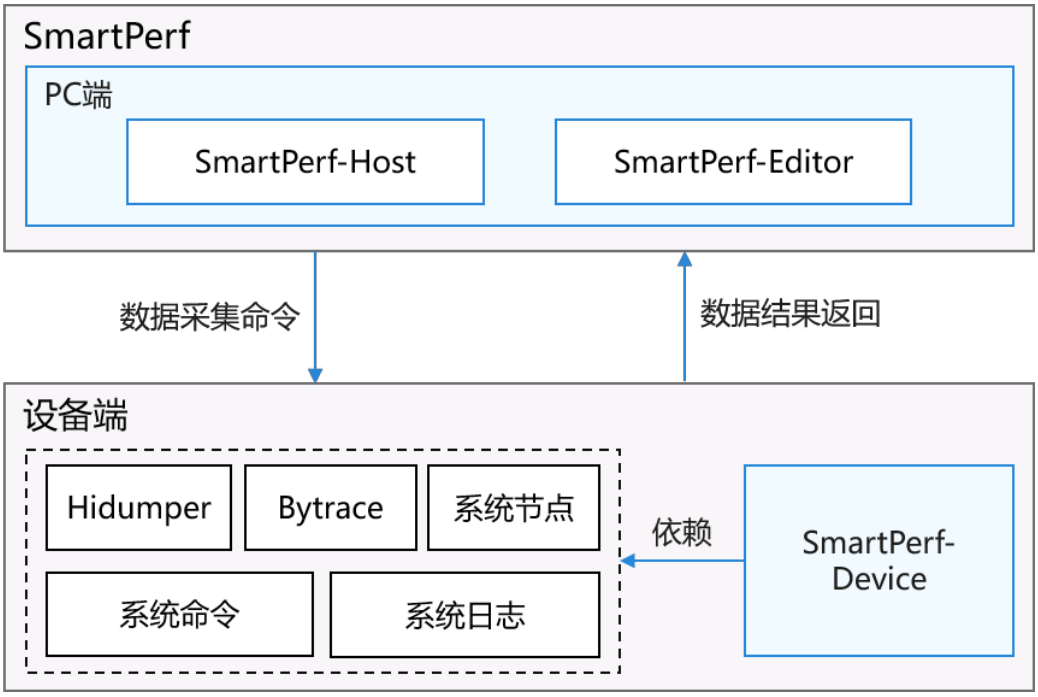

图 4-16：SmartPerf 主要结构

- SmartPerf-Host：可视化展示设备侧抓取的性能数据，帮助开发者快速发现和分析系统性能瓶颈。
- SmartPerf-Device：支持设备侧 FPS、功耗、热、Soc 信息的实时采集、实时展示和数据报告生成。预置或安装在设备中使用，针对带屏设备，测试过程中在设备悬浮窗实时展示测试过程中的性能数据；对于无屏设备，通过命令行获取测试过程中生成的性能数据。
- SmartPerf-Editor： 独立的 PC 客户端，支持用户操作录制回放，定点采集性能，trace 自动抓取与报告展示，用户自定义采集场景及采集指标等功能。

具体的功能点，请参考下表：

表 4-4：SmartPerf 功能介绍

部件名	功能点	功能描述
SmartPerf-Host	CPU 分析	CPU 线程运行时状态分析。
		CPU 调用栈信息分析。
	内存分析	Native memory 申请和调用栈分析。
	IO 分析	FileSystem 流量和调用栈分析，BIO 申请空间与调用栈分析。
	自定义 trace 抓取	用户自定义抓取的数据结构和展示方式。
	查询	SQL 语句查询。
SmartPerf-Device	基础能力采集	CPU、GPU、DDR、Temperature、Power、应用 RAM、FPS 等数据采集。
	UI 可视化操作	可操作采集任务开始、暂停、继续、停止。
	悬浮框	实时监控采集数据的数值变化，通过折线图查看数据走势。
	报告展示	查看 csv 报告，报告页查看图形化展示。
	性能数据采集	通过采集的 trace 分析冷热启动，点击完成时延、点击响应时延，滑动帧率、滑动响应时延的数据。
SmartPerf-Editor	自动化性能测试	应用启动完成时延、响应时延、点击滑动响应时延、界面滑动帧率等自动化测试。

部件名	功能点	功能描述
	自定义录制回放测试	用户操作脚本的录制、回放、采集点设置。
	报告展示和数据拉取	性能采集数据结果的报告呈现及数据异常的 Trace 自动拉取。
	多轮测试	性能测试页支持冷启动多轮测试功能。
	数据管理和查询	历史测试报告查询功能，支持根据 TaskID 模糊查询和测试类型分类查询。

3. 稳定性测试工具

DevEco Testing wukong 是 UI 自动化稳定性压测工具，用于帮助开发者快速发现应用稳定性问题。支持随机压测、控件顺序遍历、事件录制回放等测试能力，wukong 主要功能模块如下图所示：

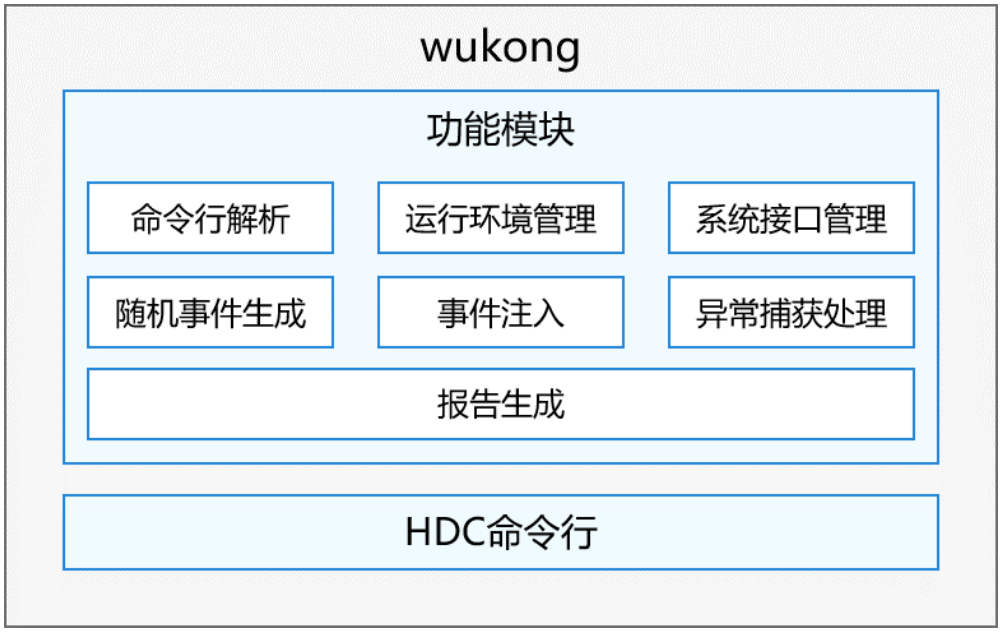

图 4-17：wukong 功能全景图

- 命令行解析：支持命令行获取参数并解析命令行参数。
- 运行环境管理：根据命令行初始化 wukong 整体运行环境。
- 系统接口管理：检查并获取指定的 mgr，注册 controller 和 dfx 的 faultlog 的回调函数。
- 随机事件生成：通过 random 函数生成指定种子数的随机序列，生成事件。
- 事件注入：根据支持的事件类型向系统注入事件，依赖窗口、多模、安全等子系统。
- 异常捕获处理：通过 DFX 子系统获取运行中的异常信息。
- 报告生成：保存运行时的异常信息并记录执行日志，生成报告。
- HDC 命令行：依赖 hdc 将命令下发至设备端，支持 exec 和 special 两类命令集。

4. 分布式设备录制回放

提供基于开发者对设备操作序列的监听，识别所操作的控件，转换生成测试脚本的能力，支持多设备分布式场景自动化脚本录制，基本架构原理如图所示：

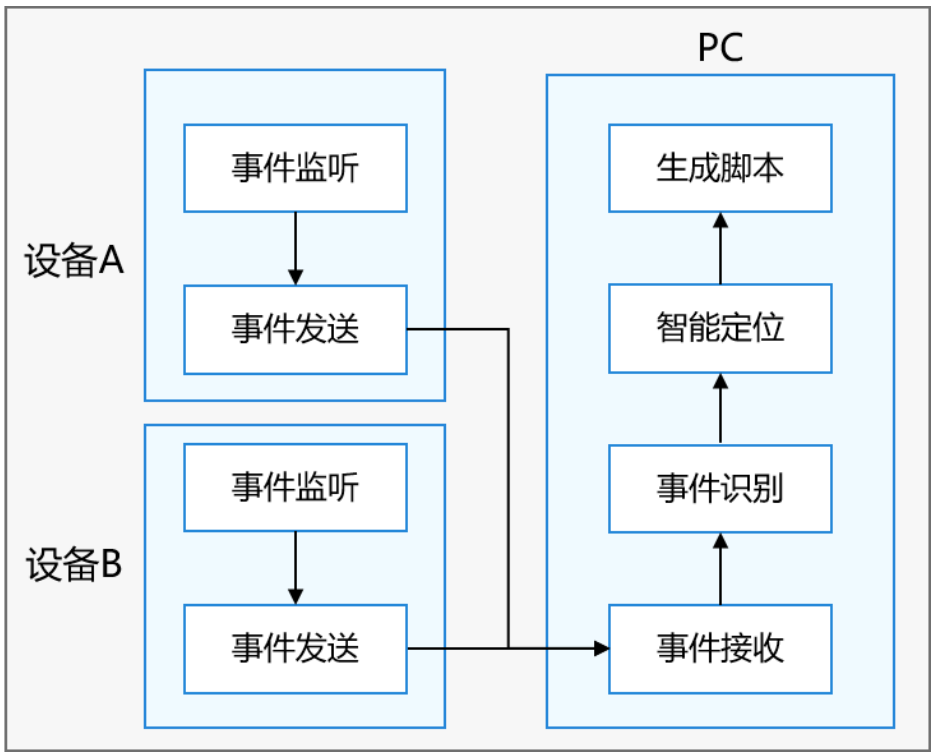

图 4-18：分布式录制架构原理

- 事件监听：监控设备操作状态，抓取设备操作事件流，同步到主控 PC。
- 事件识别&智能定位：多个设备通过 USB 同步时钟，按时间顺序处理监听到的设备事件，根据事件的坐标等信息结合当前设备显示页面的结构智能识别还原真实的操作，生成对应的 DSL 操作描述。
- 生成脚本：解析 DSL 描述，生成对应脚本，通过自主编辑增加检查点，就形成了完整的脚本。

专项测试

专项测试是应用/服务的多维度测试，包括全方位的质量体检，同时也提供性能、功耗、稳定性、兼容性、UX、安全等专项自动化测试。

1. 应用与服务体检

应用与服务体检用于检测并评价应用与服务的质量，提供评估结果和改进建议，帮助开发者提升产品质量。通过 DevEco Studio 连接本地设备或模拟器，自主遍历应用或者服务。检测维度目前包含：设计约束、兼容性、性能、稳定性。

- 设计约束：代码实现需要遵循的设计约束和规范，比如：App 中每个 FA（hap）必须明确支持的设备类型。
- 兼容性：在设备上是否可以正常的安装、启动、卸载。
- 性能：在设备上启动时间、滑动帧率、前台内存占用。
- 稳定性：运行时出现崩溃、Crash、冻屏。

最终的检测报告中按照测试维度进行分层展示，根据每条规则的权重，综合算出应用/服务的评分。

2. 专项测试云测平台

DevEco Testing 专项测试服务以云端服务的方式提供了多维度的专项测试能力，无需人工干预，自动完成应用/服务的测试。专项测试报告可以帮助开发者提前识别和定位问题，为消费者带来更佳的使用体验。

云端的 DevEco Testing 专项测试服务提供了多设备环境，可用于一次开发多端部署测试，降低开发者获取测试设备的门槛，并提供自动化的测试能力。

云端测试包含设计约束、安全漏洞检测、隐私合规检测、兼容性测试、稳定性测试、性能测试、功耗测试、UX 测试。可检测应用或服务从安装、启动、运行、卸载全生命周期的问题，如应用崩溃、冻屏、启动响应耗时长、前后台内存/CPU 占用高、启动/卸载异常等。

Chapter 5

统一上架与多端分发

- 1）快速上架
- 2）应用分发
- 3）服务分发

HUAWEI AppGallery Connect 为开发者提供全球化、全场景一站式应用分发能力，并为开发者提供质量、安全、工程管理等领域的能力，大幅降低应用开发与运维难度，提升版本质量，帮助开发者获得用户并实现收入的规模增长。

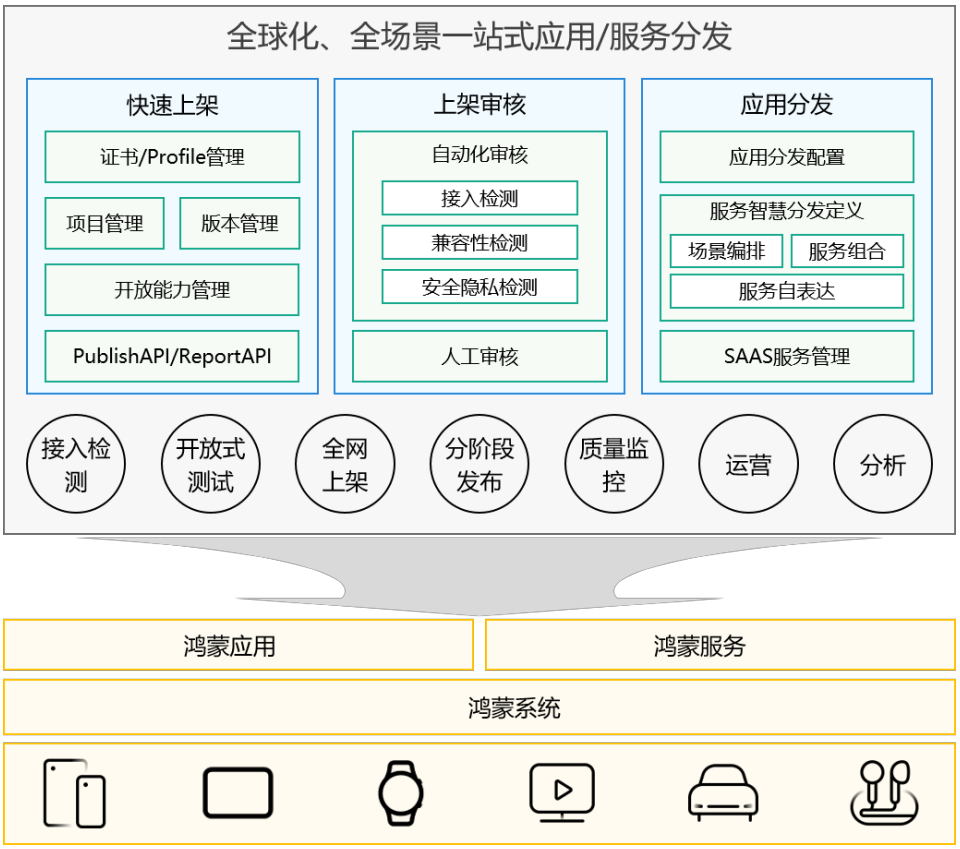

图 5-1：一站式分发能力

1）快速上架

HUAWEI AppGallery Connect 作为开发者统一提交入口，集成证书管理、项目管理、版本管理等功能，支持鸿蒙生态应用、原子化服务的快速上架与分发。

证书颁发

提供统一的证书颁发入口，支持软件证书和 Provision Profile 管理，保障开发者合法性与应用合法性。

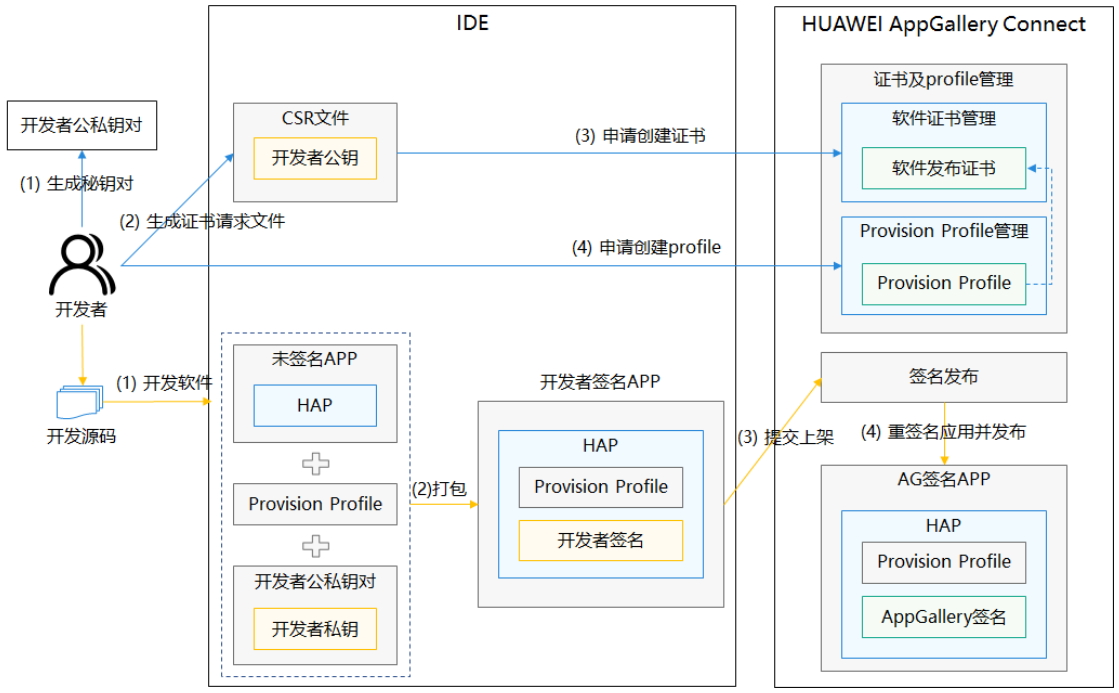

图 5-2：签名方案

- 开发者生成自己的公私钥对（Key Store），并且使用 IDE 生成软件证书请求文件（CSR）。
- 开发者上传软件证书请求文件，生成软件发布证书，选择对应的证书可以创建应用的 Provision Profile。
- 开发者使用 Provision Profile 和开发者私钥完成应用的本地签名，打包应用并提交。
- 只有经过 HUAWEI AppGallery Connect 合法性检测、拥有开发者所属软件证书签名的应用才允许提交。
- 经过合法验签之后，使用 HUAWEI AppGallery Connect 的秘钥对应用进行重签名。

- 允许通过以上步骤的应用安装到搭载鸿蒙系统的设备上。

统一上架

开发者开发完成之后，上传包体、描述信息、素材等，提交上架审核。也可委托 SaaS 厂商为其代理完成开发上架。

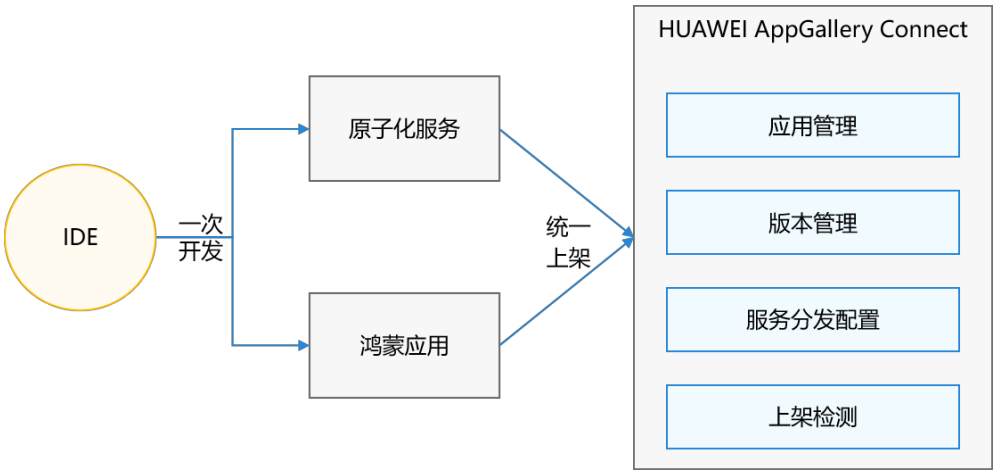

图 5-3：统一上架能力

- 应用管理：提供应用的基本信息管理，支持配置全球化名称，管理图标、截图、描述、应用分类、开发者服务信息等内容。
- 版本管理：维护应用的版本信息，支持配置发布区域、发布范围、应用资费、内容分级、隐私声明、版权信息等内容。
- 上架检测：支持对应用进行基础信息检测，包括 API、包名、签名、资源等，以及对 SDK 接入情况进行检查。

上架审核

为了给用户提供更安全且出色的体验，HUAWEI AppGallery Connect 对开发者提交的鸿蒙生态应用、原子化服务进行严格的审核与测试。开发者需了解并遵循《华为应用市场审核指南》。了解更多详细信息请访问：

https://developer.huawei.com/consumer/cn/doc/distribution/app/50104

完成实名认证才能享受开发者联盟开放的各类能力和服务。

表 5-1：开发者检测

检测	简介
实名认证	只有实名认证过的开发者，才允许进行应用上架分发；应用市场支持个人开发者和企业开发者认证，认证方式多样化。个人开发者：银行卡认证、身份证认证、华为云授权认证；企业开发者：对公银行认证、企业资料认证、华为云授权认证

开发者需提供资质文件以证明其内容符合法律、法规或政策的要求，同时为保障软件在设备上具备良好的使用体验，会对其兼容性、安全性、稳定性、隐私、性能、功耗等进行全方位检测。

表 5-2：应用/服务检测

检测	简介
行业资质检测	支持全行业资质自动化检测，包括游戏版号、计划及软件著作权证书、支付业务许可证等
公司资质检测	对公司的合法性进行检测，包括合同、协议、授权书、免责函、安全评估报告等资质的自动化检测
安全检测	进行安全风险问题检测，包括病毒木马、恶意行为、代码安全、逃逸对抗等
隐私检测	通过动/静态检测，识别是否存在隐私风险，比如违规收集个人信息、超范围收集个人信息、违规使用个人信息、强制/频繁/过渡索取权限、强制用户使用定向推送等问题
兼容性检测	通过真机检测，保障分发设备的兼容性，支持检测是否存在崩溃、无响应、运行错误、功能异常、界面异常等问题
合规检测	通过 AI 技术，识别图片、描述、文本等信息，自动检测内容是否存在色情、暴恐违禁、赌博、毒品、政敏、低俗、禁播等违规行为

2）应用分发

HUAWEI AppGallery Connect 提供了灵活的分发能力，支持按阶段、维度、场景等多种形式，高效、精准地分发到用户设备上。

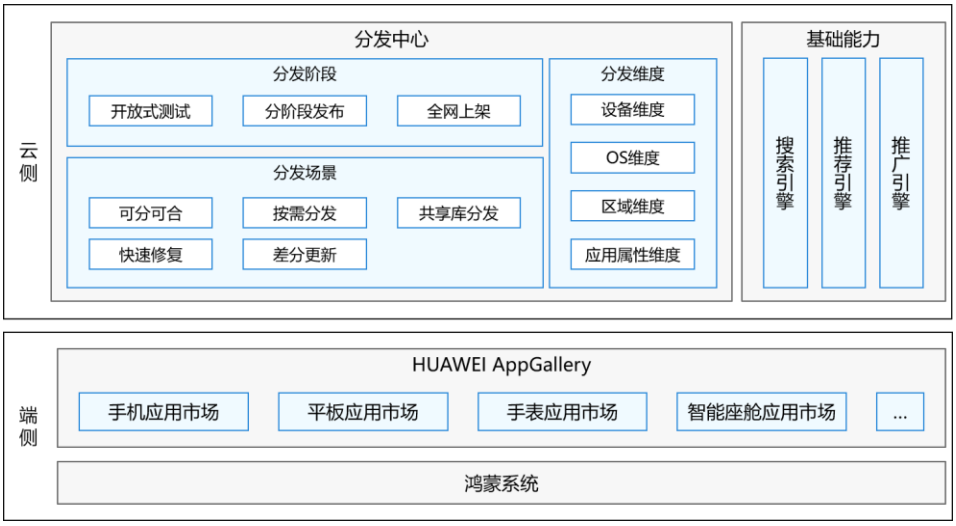

图 5-4：应用分发全景图

分发阶段

开发者可以在应用的不同成熟阶段采用不同的分发手段，结合应用的运行数据与用户声音，不断改进应用质量，持续提供优质服务。

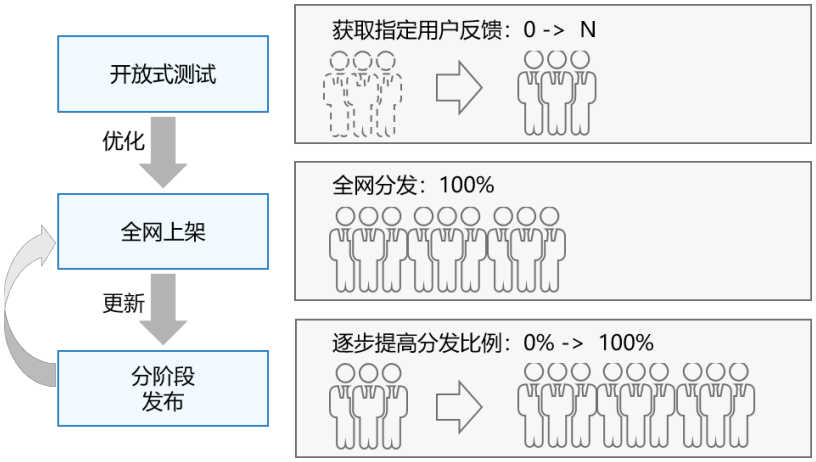

图 5-5：分发阶段

分发维度

开发者可以通过多种维度灵活分发应用：

- 设备维度：设备类型、PCID、屏幕类型等。
- OS 维度：API Level 等。
- 区域维度：发布国家区域、漫游分发等。
- 应用属性维度：年龄等级、应用分类、软件版本等。

分发场景

开发者上架应用之后，HUAWEI AppGallery 会根据不同的分发场景，选择最优的分发方式，从而实现高效分发，提升用户体验。

- 按需分发：开发者可以将应用进行合理拆分，将非核心的功能做成动态特性；用户首次下载应用时，只下载基本功能模块，仅在执行到动态特性时才按需下载，既满足了业务功能，也减少了存储空间的占用，消耗更少的网络流量，提升下载转换率。
- 快速修复：当开发者发现代码 bug 或者关键漏洞时，HUAWEI AppGallery 提供代码方法级的快速修复方案，支持快速解决问题。

智慧分发

HUAWEI AppGallery 构建了搜索、推荐、推广三大引擎，从海量数据中构建丰富的画像和知识图谱，基于 AI 能力实现精准送达。

表 5-3：引擎介绍

能力	简介
搜索引擎	基于精准的用户意图识别与丰富的鸿蒙生态应用、原子化服务标

能力	简介
	签体系，构建以用户体验为中心的多场景、多模态、全球化搜索引擎，高效连接鸿蒙生态和全球消费者
推荐引擎	基于海量用户数据，使用机器学习和深度学习算法，提供千人千面的个性化推荐服务，精准触达目标用户
推广引擎	面向合作伙伴提供精准、优质、高效的推广服务，支持面向安装、激活、次留、付费等目标的投放，助力合作伙伴快速精准获量，实现商业成功

3）服务分发

鸿蒙系统作为万物互联时代泛终端服务的载体，面向跨设备多终端环境， 实现“服务随身”的跨设备无缝体验， 原子化服务与 AI 算法深度配合，实现多入口、场景化分发。多入口分发包括设备入口、系统入口、应用入口，场景化分发是系统在理解用户的基础上，结合用户旅程的一种多服务组合分发，从用户场景出发，围绕用户旅程的场景化闭环。例如用户想去旅游，出行前要查看天气、预定机票酒店、购买门票，旅途中要去机场、打车、结束后要照片分享，发表感受等，在整个用户旅程中，都需要用户自己规划，自己寻找服务完成对应的操作。多服务的场景化分发是在理解用户意图后，将用户旅程中需要用到的服务组合起来，在用户需要的时候分发给用户，比如查看天气、预定机票酒店、购买门票、打车、航班提醒、分享感受等服务，让用户在旅程中感受到智慧便捷，实现场景的闭环。

入口丰富

1. 协同配合

鸿蒙系统的多入口不仅体现在数量多，层次多，并且体现在用户场景上的协同配合。如用户自驾去某景点，先使用手机导航选定路线，进入车后，导航从手机流转到车机上，并基于用户意图途中语音介绍景点，接近景点时，启动购票服务。

2. 流量矩阵

流量矩阵是入口流量的有机组合，能实现服务的触达、留存、召回全生命周期管理。

- 触达：系统根据用户偏好和所处的时间、地点等场景，识别用户意图，匹配用户所需内容主动推荐服务，使用户能发现所需服务。
- 留存：系统提供优质原子化服务和内容，吸引用户将服务留存在桌面，并为用户提供统一的原子化服务查看、搜索、收藏和管理功能。
- 召回：通过系统能力再次推荐优先匹配服务，唤醒用户再次使用，召回用户。

智能分发

原子化服务数量的持续增加给用户带来查找服务不方便、选择困难等问题，若无法提供精准快捷的服务触发，用户将面临信息过载和获取服务时间成本过大的困扰。AI 的“感知”、“理解”、“推理&决策”等能力，能有效解决用户查找服务不方便及选择困难问题。智慧分发核心能力分为感知、知识与理解、推理三层：

- 感知层：精确感知用户场景，是服务智能分发的基础。根据多个终端的硬件传感信号和软件感知能力，感知层可以感知时间、空间、动作等信息进一步支撑对场景的理解。结合用户偏好，辅以知识图谱提供的结构化数据，系统实现了场景的精准融合感知。随着用户使用时长和次数的增加，场景的感知能力也将更加精准，推荐的服务将更加贴心，更好地满足用户的需求。场景感知分为如下几类：
  - ✧ 基于时间的场景感知：工作日、节假日、首次亮屏、午休、睡前时光等。
  - ✧ 基于地点的场景感知：家、公司、公共交通、商城、旅游景点等。
  - ✧ 基于设备的场景感知：音频播放状态、网络连接能力、运动状态等。
  - ✧ 基于人物的场景感知：性别，年龄，应用使用偏好等。
  - ✧ 基于事件的场景感知；快递物流，异常天气，订单状态、未接来电等。
  - ✧ 特别说明，以上场景感知信息均在用户知情同意的前提下收集。
- 知识与理解层：知识与理解层是智能分发决策的重要依据，围绕核心场景，持续构建、学习、丰富知识，并基于全面感知与知识增强，精准理解用户意图。感知数据结合用户的行为习惯，辅以知识图谱提供的结构化数据作为输入，通过对用户、场景建模以及 NLU 等技术，实现了用户此刻的意图理解。同时在原子化服务上架的过程中，开发者可以选择服务分类和标签，通过服务分类、标签和智能理解，系统可以将原子化服务与统一全局意图进行关联。
- 推理&决策层：依托丰富的服务生态，完备的知识储备，学习型 AI 模型实现精准推理。通过基于规则的召回、热度召回、协同召回、深度学习模型召回等多路召回方

式，为每个用户召回与其意图、兴趣相关的原子化服务，同时通过端云融合排序模型将召回的服务进行排序，并将 top 服务展示给用户。

开发者可以按照服务分发接口规范接入数据，使用户意图和服务数据更精准匹配，从而显著提升原子化服务的使用。

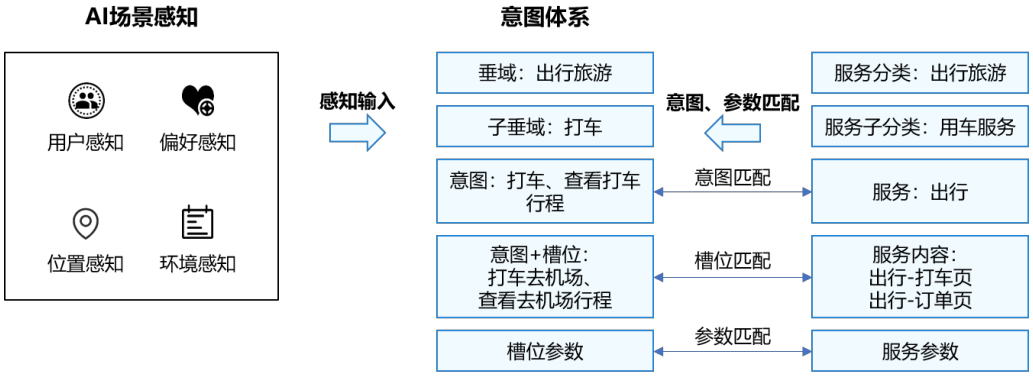

图 5-6：基于场景感知的 AI 分发原子化服务

数据接入接口按照垂域进行细分，例如附近优惠券，附近服务，热词排行，音乐内容等。开发者可以根据自己的服务具体使用场景，有选择的通过这些接口接入数据。

Chapter 6

自由流转与分布式运行环境

- 1）价值与架构定义
- 2）跨端迁移
- 3）多端协同

随着个人设备数量越来越多，跨多个设备间的交互将成为常态。基于传统 OS 开发跨设备交互的应用程序时，需要解决设备发现、设备认证、设备连接、数据同步等技术难题，不但开发成本高，还存在安全隐私、兼容性、性能等诸多问题。为了适应万物互联时代的环境变化，鸿蒙系统构建了基于分布式运行环境所需要的基础设施，为开发者提供了基础的分布式框架能力，使开发者可以更方便的实现跨设备的业务开发，向用户提供多设备的交互体验。

基于人机交互研究发现，多设备的交互场景按照时间的串、并行，分为以下两种类型：

串行交互：指用户相继使用多个设备。此类交互场景要求具备较高的连续性、一致性。

- 连续性：当用户从一个设备转向另外一个设备的时候，最新的操作状态应当是继续保留的、未被中断的。
- 一致性：当用户在使用手表、手机、大屏等不同设备时，交互方式与基础视觉元素应当是一致的，例如多指手势，控件样式等。这里的“一致”并不等于与“相同”，由于设备的屏幕尺寸、形态的不同，视觉元素可以进行适配调整。

并行交互：指用户同时使用多个设备。此类交互场景要求具备较高的协作性、互补性。

- 协作性：多个设备彼此交互协调，完成一项任务。
- 互补性：利用设备的本身形态差异，完成一项任务。例如，当用户在家里找不到电视遥控器的时候，手机可以变身为遥控器，这就是一种设备能力的互补。

对于并行交互与串行交互两者典型场景，鸿蒙系统分布式运行环境分别提供了与之对应的基础能力，即多端协同与跨端迁移，两者统称为“自由流转”。

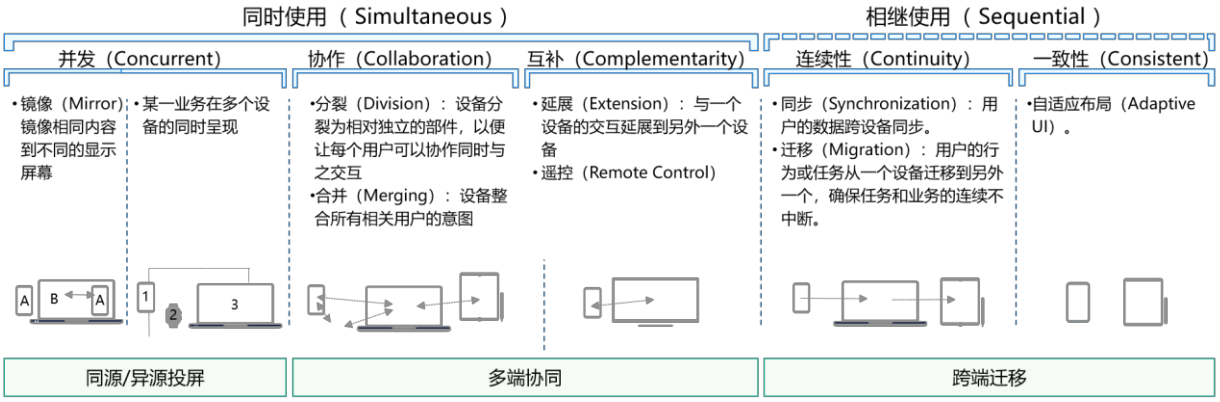

图 6-1：自由流转的两种形态

1）价值与架构定义

价值

自由流转的价值体现在以下几个方面：

- 自由流转提供了应用跨设备流转的能力。应用开发只需遵循框架并适配指定的 API，就能实现设备之间的跨端迁移和多端协同；

- 自由流转框架实现了流转过程，包括流转任务发布、应用免安装、数据序列化、兼容性判断等。应用开发只需关注在业务数据本身的同步与恢复，简化了应用的处理逻辑，降低了应用开发跨端特性的复杂度；

- 自由流转将彻底改变应用分发模式。鸿蒙生态应用不再与传统 OS 应用那样，只能局限在单一设备上。

架构定义

自由流转依赖于鸿蒙系统提供的分布式运行环境，该环境从下而上可以分为四层：

- 核心基础服务：为提供自由流转设备互联的核心基础服务，主要包括如下模块。

  ✧ 设备管理服务 ：提供设备管理相关的能力。设备管理服务在系统中的定位是提供应用开发接口的核心服务实现。

  ✧ 分布式软总线：主要提供基于近场通信技术的通信网络，实现分布式设备之间的有序通信，使得设备之间的传输变得安全可靠、通信 QoS（Quality of Service）可管理、业务质量可预期。

  ✧ 设备画像（Device Profile）：是设备硬件能力和系统软件特征的管理器。典型的设备 Profile 信息包括设备类型、设备名称、存储容量、是否折叠屏、有无屏幕、分辨率、设备安全等级、设备 OS 类型、OS 版本号等。设备画像可用于支持智能决策服务进行设备筛选、排序，也可用于支持设备信息的查询。

  ✧ 智能决策服务：提供智能化的设备筛选能力、设备排序能力。设备筛选能力主要的数据来源基于设备画像。

  ✧ HiChain：是设备互信认证部件，提供了设备间互信关系建立、管理、认证、解除的全生命周期管理能力，支撑设备间搭建安全的数据传输通道，是鸿蒙系统设备分布式可信互联的安全基础能力。

  ✧ 身份认证服务：提供端侧统一的用户身份管理、身份认证和访问控制判断能力。支持多用户操作系统，支持多种用户身份认证方式（包含 PIN、指纹、人脸等）。

- 分布式平台服务：负责拉通多个物理设备上的运行状态，同时提供跨设备间的资源访问和控制能力。
- 核心业务框架：对分布式平台服务层和核心基础服务层所提供能力进行调用，完成相应的业务功能，给应用提供更简单友好的使用接口。
- 开发接口层：所有的应用通过这一层来使用系统提供的能力。

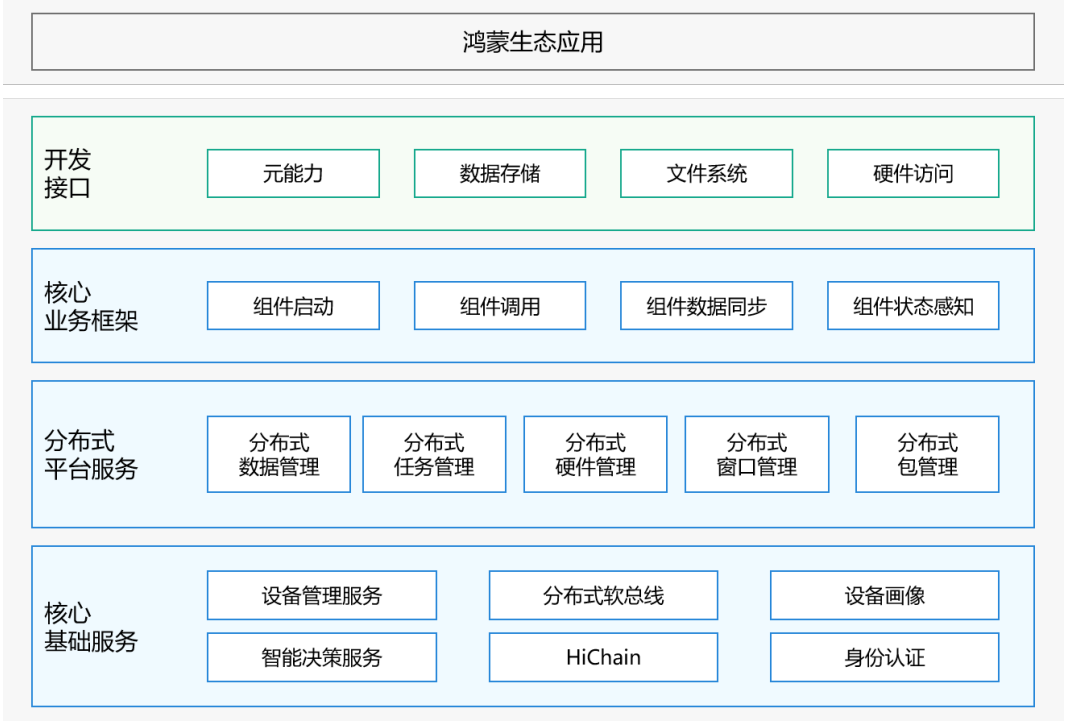

图 6-2：分布式运行环境

2）跨端迁移

跨端迁移是指将一个软件实体从一台设备转移到另外一台设备上运行。借助跨端迁移能力，鸿蒙生态应用可以自由地在多个设备之间流转，为用户带来无缝的用户体验，也会为开发者带来更多的入口和流量。

跨端迁移应用场景

用户使用应用的情境发生变化时（例如从室内走到户外、从办公室到车上等），之前使用的设备可能已经不适合继续当前的任务，或者周围有更合适的设备，此时，可以选择使用新的设备来继续当前的任务。

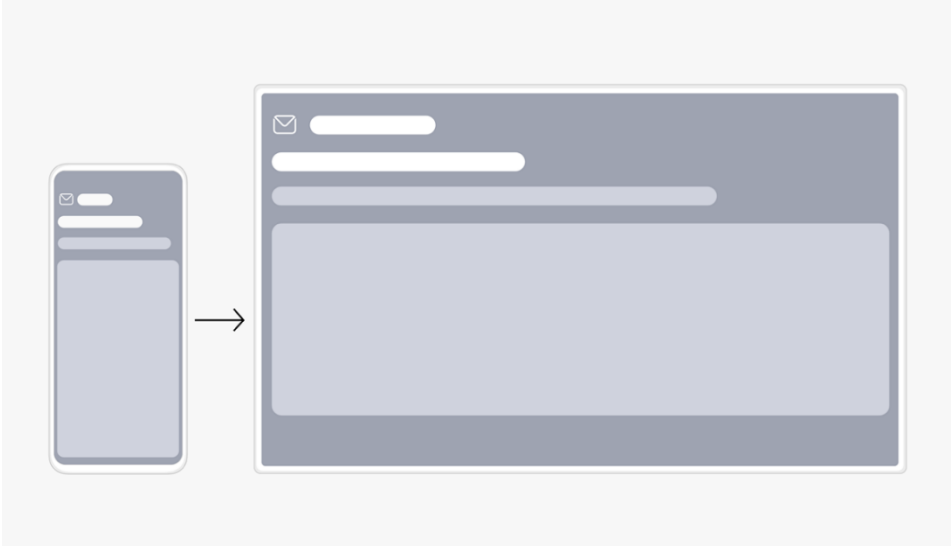

图 6-3：跨端迁移示意图

跨端迁移的应用场景举例：

- 在外时手机上编辑邮件，到家后迁移到平板上继续编辑。
- 在外时手机玩游戏，到家后迁移到平板上继续玩。
- 在家里智慧屏上看视频，出门时迁移到手机上继续观看。
- 手机视频通话迁移到智慧屏，更沉浸地视频聊天。

3）多端协同

多端协同是指运行在多个物理设备上的软件彼此协作完成一项任务。通过充分发挥每种设备的优势能力（例如智慧屏显示能力、手机输入输出能力等），为用户提供更好的体验。

根据协同能力的不同，例如显示能力、交互能力等，可以创造出丰富的协同模式。下面以显示协同、交互协同两种模式为例展开介绍。开发者可以根据应用的实际特点基于鸿蒙系统提供的分布式开放能力，选择合适的模式进行体验设计，或者展开更多探索、创造出新的协同模式。

显示协同

场景举例：文档编辑应用的文档内容和周边工具菜单可以分别显示在智慧屏和手机上，在手机上快速操作编辑菜单，在智慧屏上更清晰地查看编辑的效果。

图 6-4：显示协同示意图

交互协同

场景举例：在智慧屏上进行搜索时，在手机上进行文本输入。通过智慧屏上网课时，在手机上进行手写答题或绘画。

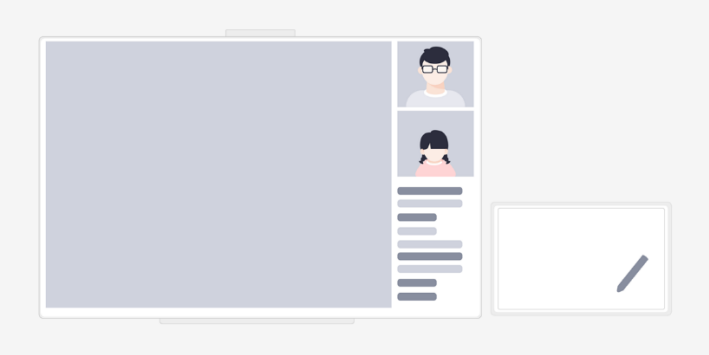

图 6-5：交互协同示意图

算力协同

场景举例：分布式游戏，在手机/大屏上玩游戏时，利用周边设备（手机、平板、笔记本等）协助完成游戏应用的计算任务(AI 计算、图像渲染)，提升游戏帧率、画质。

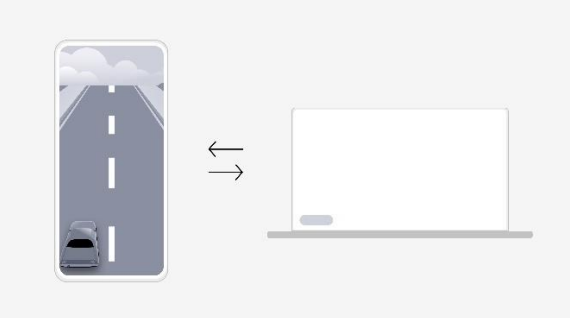

图 6-6：算力协同示意图

Chapter 7

全方位运维分析

HUAWEI AppGallery Connect 提供低门槛、高效率、多场景的大数据能力，包括质量分析、性能调优、故障定位、行业风向等。同时支持多维度数据分析，智能诊断问题并给出解决方案，为开发者明确质量优化方向，提升用户体验。

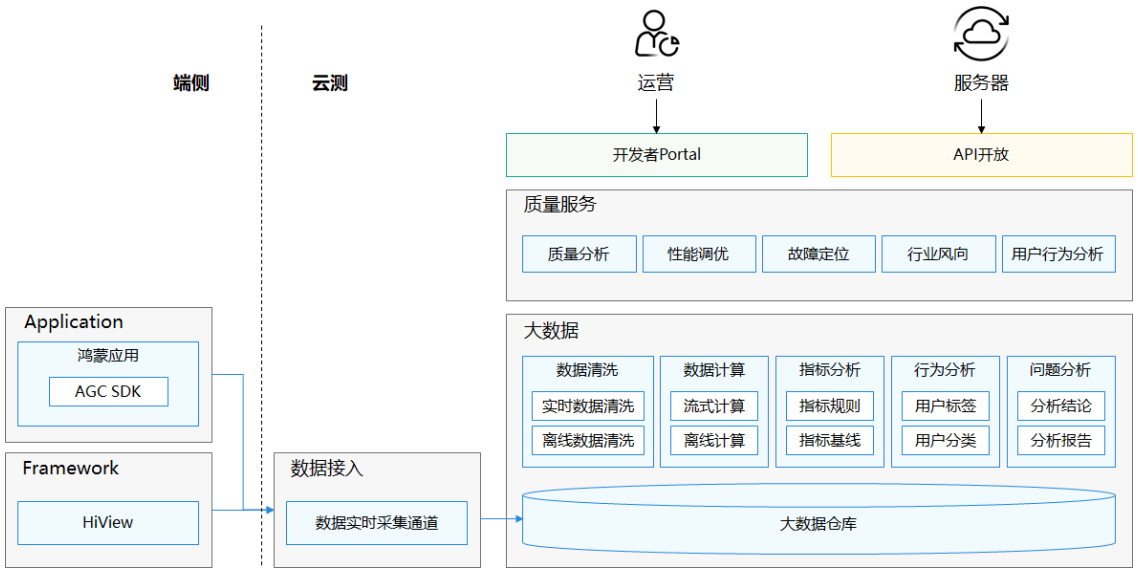

图 7-1：运维监控架构图

崩溃服务

崩溃服务帮助开发者快速发现、定位、解决应用崩溃（又称闪退）问题，使用时无需开发任何代码即可实现可视化数据报告的实时查看。也能检测到在每个设备上的运行状态，并提供实时数据报告，让开发者不错过任何一个崩溃问题。同时，崩溃服务智能汇聚崩溃问题，提供每个问题发生时的环境信息、堆栈等分析数据，让开发者轻松识别问题优先级，快速解决问题。

表 7-1：崩溃服务功能

主要功能	功能描述
可视化实时报告	零代码集成，自动生成包含堆栈及其他相关信息的可视化数据报告，实时跟踪应用稳定性。
可自定义报告	提供了用户标识符、键值对和日志三种机制增强报告。
智能分类	大量崩溃会按照异常类型、代码位置自动分类，开发者可以根据对用户的影响程度对崩溃进行排序，确定优先级别。
实时监测和提醒	分钟级实时报告让您可以实时跟踪应用稳定性，当发生重大崩溃时，系统能及时提醒开发者。

性能管理

性能管理（APM，App Performance Management）服务提供分钟级应用性能监控能力，检测应用在每个设备上的运行性能数据，帮助开发者快速发现、定位、解决应用性能问题（包括：应用启动慢、页面加载慢、应用无响应、网络请求慢等），确保应用运行平滑流畅，持续提升应用的用户体验。

表 7-2：性能管理功能

主要功能	功能描述
可视化实时报告	零代码集成，自动生成多维度（国家/地区、运营商、网络等）可视化数据报告。

主要功能	功能描述
自动采集应用性能数据	自动采集应用关键性能数据、应用无响应数据、应用体验分析数据、应用启动性能数据、应用屏幕性能数据、HTTP/HTTPS 网络性能数据、前台和后台活动性能数据等。
自定义场景跟踪	创建自定义跟踪记录来监控应用在自定义场景（如用户登录场景）下的性能，为自定义跟踪记录添加指标（如登录耗时）和属性（如帐号类型）。

云服务监控

云服务监控是面向云函数、云数据库等云服务的质量监控解决方案，帮助开发者快速发现、定位、解决云服务的业务层性能问题。支持云服务日志分析、自定义日志、调用链路分析等，帮助开发者从调用全链路角度分析性能瓶颈，保障用户体验。

表 7-3：云服务监控功能

主要功能	功能描述
指标看板	呈现云服务相关关键指标。
日志服务	支持各云服务日志查询、统计、上下文查看等。
自定义场景跟踪	分析业务请求在云服务间的调用链路，节点耗时等。
告警服务	基于指标设置自定义告警，支持短信、邮件、互动中心、webhook 等多种告警方式。

故障监控和预防

如果应用程序一段时间内无法响应用户的交互，系统则会向用户提供一个可快速恢复的方法，在系统界面显示一个对话框，用户可以选择等待程序继续运行，也可以选择强制关闭程序，这种情况被定义为应用无响应。其主要的技术监控指标包括：发生次数、影响用户数、用户 ANR 率、现场信息、主线程堆栈、 其他线程堆栈、系统日志等。

Chapter 8

全场景案例参考

影音娱乐

1. 场景描述

用户通过手机拍摄短视频/直播时，边拍摄边控制手机非常不方便，且无法做到多视角拍摄。在一些多人场景下如果能支持多机位、多视角拍摄，短视频或者直播的效果会更加出色。例如：乐队直播音乐表演时，智慧屏拍摄整个乐队，多台手机单独拍摄主唱和其他伴奏成员，最后通过一个平板或者手机整体控制切换不同的拍摄视角，完成整个直播的拍摄工作。这样拍摄出来的视频，既可以看到乐队的整体，又可以看到每个乐队成员的细节，体验更佳。

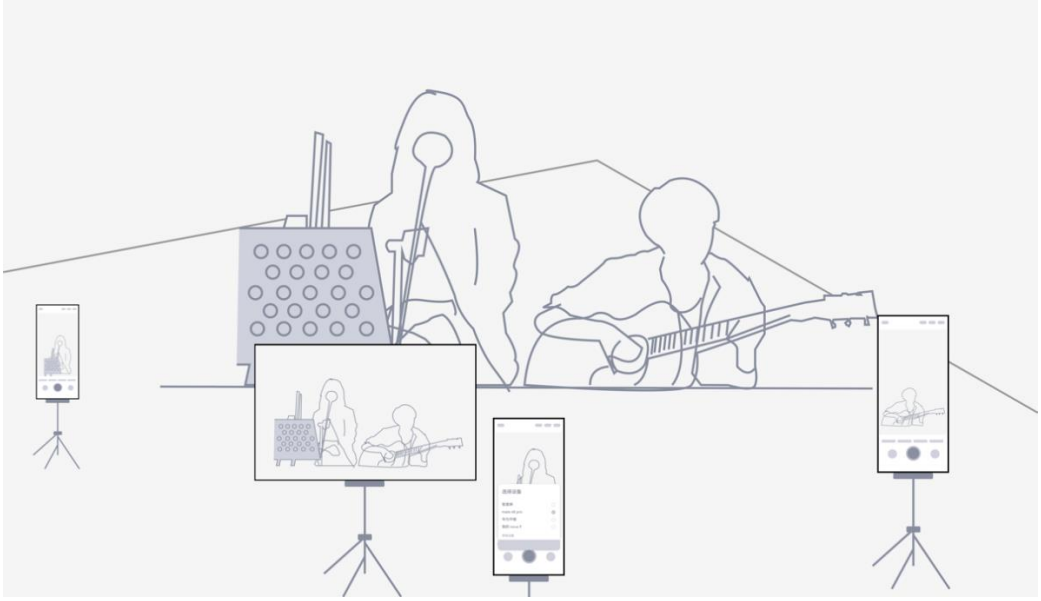

图 8-1：多机位直播场景示意图

2. 鸿蒙系统开放能力

“多端协同”、“分布式硬件（摄像头、麦克风）”等能力。

运动健康

1. 场景描述

用户通过智慧屏上的健身课程视频健身，单设备的智慧屏设备无法获取用户在健身过程中的健康数据（例如：心率、消耗热量等），进而无法根据运动数据给出更加科学的健身建议，可能会导致用户出现运动量过大或者不足等问题。基于鸿蒙系统提供的分布式能力，可以通过手表获取用户实时的心率数据，并把数据同步显示在智慧屏上，帮助用户了解健康状态。

图 8-2：分布式健身场景示意图

2. 鸿蒙系统开放能力

“多端协同”、“分布式硬件（sensor)”等能力。

智慧出行

1. 场景描述

用户外出旅行或者出差，到达机场之后需要用手机打车到目的地，当用户手上拿了行李时，用手机查看出租车的实时状态非常不方便。通过鸿蒙系统提供的分布式能力可以很轻松的解决这个问题，只需要把出租车信息从手机流转到手表，用户抬腕就可以轻松查看出租车的最新状态。

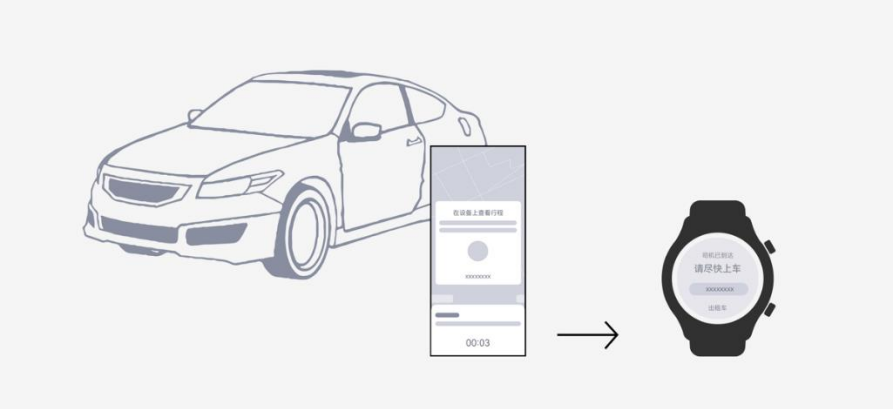

图 8-3：分布式打车场景示意图

2. 鸿蒙系统开放能力

“跨端迁移”等能力。

智慧办公

1. 场景描述

用户在上班路上通过手机编辑文档，到公司之后需要通过 PC 继续编辑时，传统的文档编辑应用需要借助云同步才能实现该需求。从用户体验看，借助“云”同步文件，如果文件很大，同步速度慢并且需要用户多次操作才能完成文档的接续编辑。从开发者投入成本看，“云”同步文件，需要服务器资源和云文件同步功能的开发，开发者在传统的单设备操作系统上开发“一多”应用也会有很多端侧重复适配工作，大大增加开发成本。

通过鸿蒙系统提供的分布式能力，开发者只需要花费很小的成本便可实现用户的跨设备文档同步并接续编辑的需求。用户只需要在手机编辑文档的页面点击流转按钮，选择需要流转的 PC，手机的文档就会同步到 PC，同步完之后自动打开文档进入编辑页面，用户就可以接着在 PC 上继续编辑这个文档了。

图 8-4：分布式办公场景示意图

2. 鸿蒙系统开放能力

“一次开发多端部署”、“跨端迁移”和“分布式文件”等能力。

智能家居

1. 场景描述

亲子游戏用于活跃家庭气氛，增进亲子关系。但是一般的多人游戏需要多个游戏手柄才能进入游戏，居家生活中，经常会出现找不到手柄的状况。通过鸿蒙提供的分布式能力，只需要把手机作为游戏手柄，随时可接入游戏。

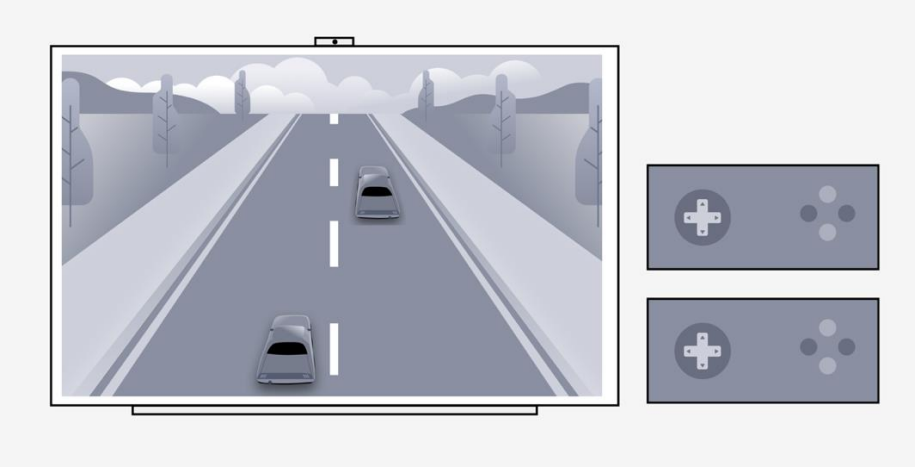

图 8-5：分布式游戏场景示意图

2. 鸿蒙系统开放能力

“多端协同”和“分布式硬件（sensor）”等能力。

Chapter 9

附录：术语

A

- Ability：应用的基本组成部分，是应用所具备能力的抽象。Ability 是系统调度应用的最小单元，是能够完成一个独立功能的组件，一个应用可以包含一个或多个 Ability。
- Application sandbox（应用沙箱）：通过自主访问控制（Discretionary Access Control, DAC）、强制访问控制（Mandatory Access Control, MAC）等访问控制机制，隔离系统资源，用于保护应用自身和系统免受恶意应用的攻击。
- App Pack：上架应用市场的应用包的组织方式，包含一个或者多个 hap 包，后缀名为.app。
- ArkCompiler：方舟编译器，是鸿蒙系统内置的组件化、可配置的多语言编译和运行平台。
- ArkCompiler Bytecode：即方舟字节码，ArkCompiler 编译工具链负责将 JS/TS/ArkTS 源码编译成字节码，作为运行时的输入，实现对应的语言的语义逻辑。该类字节码文件的后缀缩写为.abc。
- ArkUI：鸿蒙系统的原生 UI 开发框架，支撑开发者高效地构建跨设备应用 UI 界面。
- ArkTS：鸿蒙生态的应用开发语言。它在 TypeScript（简称 TS）的基础上，扩展了声明式 UI、状态管理等相应的能力，让开发者可以以更简洁、更自然的方式开发高性能应用。
- Atomic Service（原子化服务）：鸿蒙系统提供的一种全新的应用形态，具有独立入口，用户可通过点击、碰一碰、扫一扫等方式直接触发，无需显式安装，由程序框架后台静默安装后即可使用，可为用户提供便捷服务。

C

- C API：鸿蒙 SDK 提供的 native 开发接口。

H

- HAP：HarmonyOS Ability Package，一个 HAP 文件包含应用的所有内容，包括代码、资源、三方库及应用配置文件，其文件后缀名为.hap。

M

- Module：在开发态，专门指 IDE 中工程管理的一种由开发者决定的功能相对聚合的功能单元。一个 IDE 工程可以包含多个 Module。Module 可以被编译打包成一个 hap，用于在设备上安装运行。
- MSDP（Mobile Sensing Development Platform）：MSDP 子系统提供两类核心能力：分布式融合感知和分布式设备虚拟化两大部分。

  ✧ 分布式融合感知：借助鸿蒙分布式能力，将各设备感知源进行汇总融合，对用户的空间状态、移动状态、手势、健康状态等进行精准感知，构建全场景泛在基础感知能力，支撑智慧生活新体验。

  ✧ 分布式器件虚拟化：借助鸿蒙分布式能力，构筑器件虚拟化平台，将外部设备的各类器件（如 Camera、显示器、SPK/MIC 等）虚拟化为本地设备的器件延伸使用。同时具备将自身器件共享给其他设备使用的能力。

S

- Super virtual device（超级终端）：通过分布式技术将多个终端的能力进行整合，存放在一个虚拟的硬件资源池里，系统可以根据业务需要统一管理和调度终端能力，来对外提供服务。
- SystemCapability：简称 SysCap，即系统能力，指操作系统中每一个相对独立的特性，如蓝牙，WIFI，NFC，摄像头等，都是系统能力之一。每个系统能力对应多个 API，这些 API 绑定在一起，随着目标设备是否支持该系统能力共同存在或消失，也会随着 IDE 一起提供给开发者做联想。

T

- TSAOT：方舟编译运行时利用 TS 携带的类型信息，直接将 TS 语言编译成端侧机器码，使得 TS 运行阶段能获得更高的性能。

X

- XComponent：ArkUI 提供的组件接口，满足开发者自渲染的需求。

HUAWEI