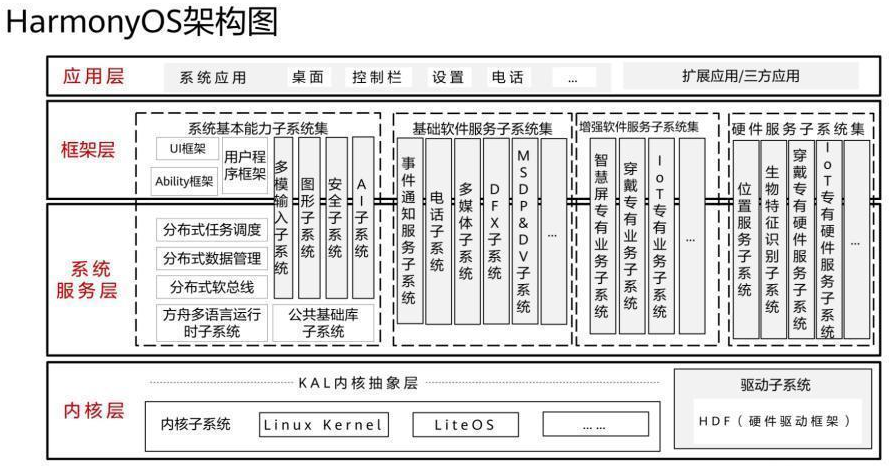
# HarmonyOS概述

* HarmonyOS是一款面向万物互联时代的、全新的全场景分布式操作系统
* HarmonyOS以手机为核心，构建了1+8+N的全场景应用
* 1+8+N：
  + 1：手机
  + 8：PC、Pad、智慧屏、车机、手表、VR/AR眼镜、耳机、AI音箱
* 在传统的单设备系统能力基础上，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，提供全场景（移动办公、运动健康、社交通信、媒体娱乐等）的业务能力

# HarmonyOS架构



* 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，支持根据实际需求裁剪某些非必要的子系统或功能/模块
* HarmonyOS整体遵从分层设计，从上往下依次为：应用层、框架层、系统服务层和内核层

## 应用层

* 应用层包括：系统应用（桌面、控制栏、设置和电话等）和拓展的第三方非系统应用
* HarmonyOS的应用分为一个或多个FA或PA组成：
  + FA（Feature Ability）有UI界面：负责提供与用户交互的能力
  + PA（Particle Ability）无UI界面：负责提供后台运行任务的能力和提供统一的数据访问抽象
* 基于FA/PA构建的新型应用生态，能够实现三方服务跨设备智能分发，提供一致、高效的用户体验
* FA在进行用户交互时所需的后台数据访问也需要由对应的PA提供支撑

## 框架层

* 用户程序框架：支持Java/C/C++/JS等多种语言
* Ability框架：应用所具备能力的抽象
* UI框架：JS UI和Java UI

|  |  |  |
| --- | --- | --- |
| Java UI和JS UI框架的对比 | | |
|  | Java UI框架 | JS UI框架 |
| 接口方式 | 命令式 | 声明式 |
| 执行方式 | 开发者处理，基于API驱动的UI变更 | 框架层处理，  基于数据驱动的UI自动变更 |
| 系统支持性 | 只有大型系统支持 | 覆盖平台更广，  轻量系统、小型系统、  标准系统、大型系统都支持 |
| 相对优势 | UI元素更丰富，开发更灵活 | 轻量化，开发更简便 |

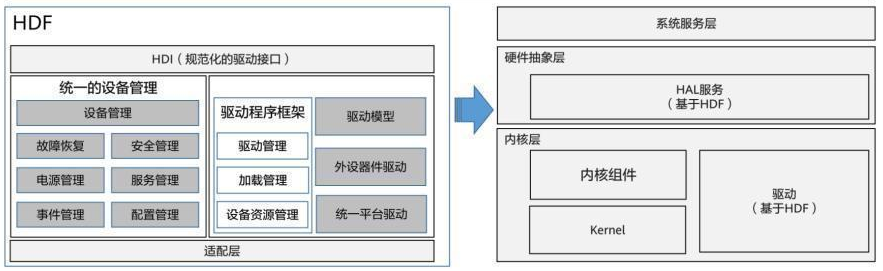
* HarmonyOS支持多种软硬件服务对外开放的语言框架，支持的开发语言包括：
  + Java
  + JS
  + CSS
  + XML（Java UI框架使用）
  + HML（HarmonyOS Markup Language）（相当于是Html的魔改版，JS UI框架使用）
  + C
  + C++
* 根据系统的组件化裁剪程度，HarmonyOS设备支持的API也会有所不同

## 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
* 系统服务层包含以下四个部分：
  + 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
    - 方舟多语言运行时子系统：提供了C/C++/JS多语言运行时和基础的系统类库，也为使用方舟编译器静态化的Java程序（即应用程序或框架层中使用Java语言开发的部分）提供运行时
  + 基础软件服务子系统集：为HarmonyOS提供公共的、通用的软件服务
  + 增强软件服务子系统集：为HarmonyOS提供针对不同设备的、差异化的能力增强型软件服务
  + 硬件服务子系统集：为HarmonyOS提供硬件服务
* 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安功能粒度裁剪

## 内核层

* 内核子系统：
  + HarmonyOS采用多内核设计，支持针对不同资源受限的设备选用适合的OS内核：支持的内核有Linux内核和LiteOS内核
  + KAL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：进程管理、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—HDF硬件驱动框架：
  + HDF是HarmonyOS硬件生态开放的基础，它提供了：统一外设访问能力，驱动开发和管理框架
  + 统一驱动框架：驱动与内核解耦，支持运行动态加载，让更多IOT设备接入超级终端：
    - 通过平台、系统接口解耦的构建统一的驱动平台底座兼容如Linux内核、LiteOS内核等不同的内核
    - 支撑百K级~G级容量的1+8+N设备部署
    - 根据不同设备形态，支持用户态部署和内核态部署



# HarmonyOS安全

* 在搭载HarmonyOS的分布式终端上，可以保证“正确的人，通过正确的设备，正确地使用数据”
* 通过“分布式多端协同身份认证”来保证“正确的人”
  + “正确的人”指通过身份认证的数据访问者和业务操作者
  + “正确的人”是确保用户数据不被非法访问、用户隐私不泄露的前提条件
  + HarmonyOS通过以下三个方面来实现协同身份认证：
    - 零信任模型：当用户需要跨设备访问数据资源或者发起高安全等级的业务操作（例如：对安防设备的操作）时，HarmonyOS会对用户进行身份认证，确保其身份的可靠性
    - 多因素融合认证： 通过用户身份管理，将不同设备上标识统一用户的认证凭据关联起来，用于标识一个用户，来提高认证的准确度
    - 协同互助认证： 通过将硬件和认证能力解耦（即信息采集和认证可以在不同的设备上完成），来实现不同设备的资源池化以及能力的互助与共享，让高安全等级的设备协助低安全等级的设备完成用户身份认证
* 通过“在分布式终端上构筑可信运行环境”来保证“正确的设备”
  + 在分布式终端场景下，只有保证用户使用的设备是安全可靠的，才能保护用户数据在虚拟终端上得到有效保护，避免用户隐私泄露
  + HarmonyOS通过以下三个方面来确保设备可靠：
    - 安全启动：
      * 确保源头每个虚拟设备运行的系统固件和应用程序是完整的、未经篡改的
      * 通过安全启动，各个设备厂商的镜像包就不易被非法替换成恶意程序，从而保护用户的数据和隐私安全
    - 可信执行环境：
      * 提供了基于硬件的可信执行环境（TEE：Trusted Execution Environment）来保护用户的个人敏感数据的存储和处理，确保数据不泄露
      * 由于分布式终端硬件的安全能力不同，对于用户的敏感个人数据，需要使用高安全等级的设备进行存储和处理
      * HarmonyOS使用基于数学可证明的形式化开发和验证的TEE微内核，获得了商用OS内核CC EAL5+的认证评级
    - 设备证书认证：
      * 支持为具备可信执行环境的设备预置设备证书，用于向其他虚拟终端证明自己的安全能力
      * 对于有TEE环境的设备，通过预置PKI（Public Key Infrastructure）设备证书给设备身份提供证明，确保设备是合法制造生产的
      * 设备证书在产线进行预置，设备证书的私钥写入并安全保存在设备的TEE环境中，且只在TEE内进行使用
      * 在必须传输用户的敏感数据（例如密钥、加密的生物特征等信息）时，会在使用设备证书进行安全环境验证后，建立从一个设备的TEE到另一设备的TEE之间的安全通道，实现安全传输
* 通过“分布式数据在跨终端流动的过程中，对数据进行分类分级管理”来保证“正确的使用数据”
  + 在分布式终端场景下，需要确保用户能够正确地使用数据
  + HarmonyOS围绕数据的生成、存储、使用、传输以及销毁过程进行全生命周期的保护，从而保证个人数据与隐私、以及系统的机密数据（如密钥）不泄露
    - 数据生成：
      * 对数据分类分级，根据分类设置相应的保护等级
      * 每个保护等级的数据从生成开始，在其存储、使用、传输的整个生命周期都需要根据对应的安全策略提供不同强度的安全防护
      * 虚拟超级终端的访问控制系统支持依据标签的访问控制策略，保证数据只能在可以提供足够安全防护的虚拟终端之间存储、使用和传输
    - 数据存储：HarmonyOS通过区分数据的安全等级，存储到不同安全防护能力的分区，对数据进行安全保护，并提供密钥全生命周期的跨设备无缝流动和跨设备密钥访问控制能力，支撑分布式身份认证协同、分布式数据共享等业务
    - 数据使用：
      * 通过硬件为设备提供可信执行环境
      * 用户的个人敏感数据仅在分布式虚拟终端的可信执行环境中进行使用，确保用户数据的安全和隐私不泄露
    - 数据传输：
      * 为了保证数据在虚拟超级终端之间安全流转，需要各设备是正确可信的，建立了信任关系（多个设备通过华为帐号建立配对关系），并能够在验证信任关系后，建立安全的连接通道，按照数据流动的规则，安全地传输数据
      * 当设备之间进行通信时，需要基于设备的身份凭据对设备进行身份认证，并在此基础上，建立安全的加密传输通道
    - 数据销毁：
      * 销毁密钥即销毁数据
      * 数据在虚拟终端的存储，都建立在密钥的基础上，当销毁数据时，只需要销毁对应的密钥即完成了数据的销毁

# HarmonyOS特性

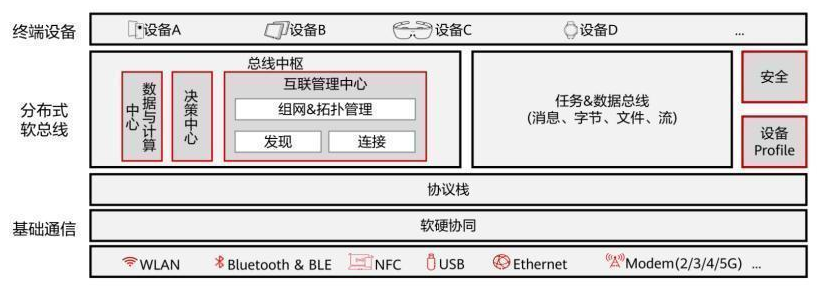
* HarmonyOS的特性有：
  + 硬件互助，资源共享
  + 一次开发，多端部署
  + 统一OS，弹性部署

## 硬件互助，资源共享

* 硬件互助，资源共享包含四个关键的分布式技术：
  + 分布式软总线
  + 分布式设备虚拟化
  + 分布式数据管理
  + 分布式任务调度

### 分布式软总线

* 概述：
  + 分布式软总线是手机、平板、智能穿戴、智慧屏、车机等分布式设备的通信基座，为设备之间的互联互通提供了统一的分布式通信能力，为设备之间的无感发现和零等待传输创造了条件
  + 开发者只需聚焦于业务逻辑的实现，无需关注组网方式与底层协议
  + HarmonyOS的分布式软总线的设计理念来源于计算机硬件总线
  + 分布式软总线技术是其他几项分布式特性的基础



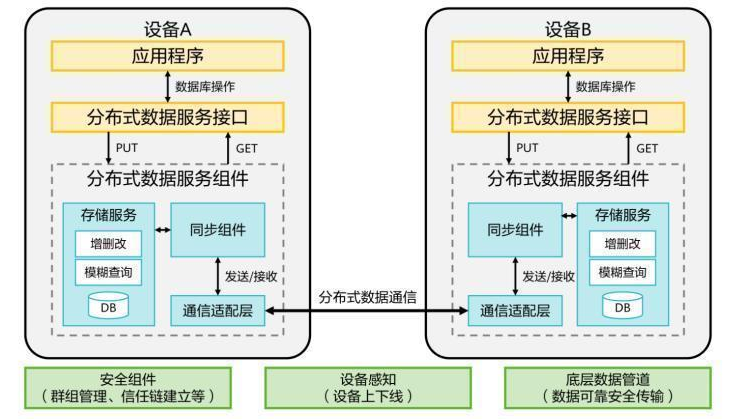
* 架构组成：
  + 数据总线
  + 任务总线
  + 总线中枢
  + 协议栈
* 特征：
  + 高可靠
  + 低时延
  + 高带宽（传输高带宽的关键技术-极简协议）
    - 传输极简API（任务&数据总线）
      * 消息
      * 字节
      * 文件
      * 流
    - 极简协议属于鸿蒙定制的通信协议，把传统通信协议的这些层级给精简了
      * 表示层
      * 会话层
      * 传输层
      * 网络层
* 自发现&连接能力：
  + 分布式软总线提出自动发现设备，实现用户零等待的自发现体验，附近同账号的设备自动发现无需等待，自动安全连接
  + 关键技术：
    - CoAP协议的广播与响应
    - HarmonyOS分布式软总线
    - Wifi、蓝牙、NFC
* 无感自组网：
  + 无感自组网是不同设备组成超级终端的核心基础能力
  + 软总线组网关键技术—异构网络组网（支持蓝牙/WIFI网络异构互通）
    - 分布式软总线提出了异构网络组网，自动构建一个逻辑全连接网络，以解决设备间不同协议交互的问题，业务开发者无需关心组网方式与物理协议
    - 面向一个逻辑网络，不感知协议
    - 业务开发与设备组网解耦：业务仅需监听上下线，业务开发成本大幅降低
    - 融合自组网：打破物理网络互通限制，异构网络自动组网，实现零等待发现组网

### 分布式设备虚拟化

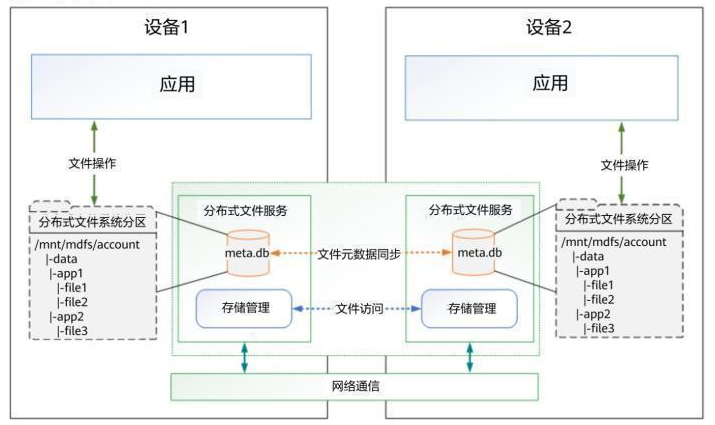
* 分布式设备虚拟化平台可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端
* 针对不同类型的任务，为用户匹配并选择能力合适的执行硬件，让业务连续地在不同设备间流转，充分发挥不同设备的能力优势，如显示能力、摄像能力、音频能力、交互能力以及传感器能力等
* 应用场景举例：在直播场景中，主播用运动相机采集视频作为直播镜头，这里相机的分布式使用主要体现了HarmonyOS的分布式设备虚拟化的技术

### 分布式数据管理

* 核心特征：
  + 更便捷的接口调用，跨设备数据库和文件访问接口和本地访问一致
  + 提供系统级数据同步能力，保证用户性能体验和数据一致性
  + 提供系统级数据安全能力，保护用户隐私
* 分布式数据管理：
  + 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，跨设备的数据处理如同本地数据处理一样方便快捷，让开发者能够轻松实现全场景、多设备下的数据存储、共享和访问
  + 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
  + 通过结合账号、应用唯一标识和数据库三元组，对属于不同应用的数据进行隔离



* 分布式文件服务：
  + 在多个终端设备间为单个设备上应用程序创建的文件提供多终端的分布式共享能力
  + 每台设备上都存储一份全量的文件元数据，应用程序通过文件元数据中的路径，可以实现同一应用文件的跨设备访问
  + 当需要在多个设备间查看、使用、编辑同一文件时，借助分布式文件系统，应用无需在设备间传递文件
* 设计原则：
  + 文件单副本
  + 离线不可见
  + Last one win冲突解决策略
* 开发约束：
  + 需要申请ohos.permission.DISTRIBUTED\_DATASYNC权限（分布式数据同步权限）
  + 文件所在设备离线时，文件无法读写，关键文件需在本地存储
  + 离线产生同名文件，组网解冲突后其中一个会被重命名
  + 并发写场景下，需要显式加锁
  + 弱网下访问性能慢或者超时，需要增加异常处理



### 分布式任务调度

* 分布式任务调度基于分布式软总线、分布式数据管理、分布式Profile等技术特性，构建统一的分布式服务管理机制，包括：服务发现，服务同步，服务注册和服务调用
* 分布式任务调度支持对跨设备的应用进行以下操作：远程启动、远程调用、远程连接和远程迁移等
* 分布式任务调度能够根据不同设备的能力、位置、业务运行状态、资源使用情况，以及用户的习惯和意图，选择合适的设备运行分布式任务
* 接口说明—分布式调度平台提供的以下能力是实现更多价值性场景的基础：
  + 连接远程PA
    - connectAbility(Intent intent, IAbilityConnection conn)接口提供连接指定设备上 PA 的能力，Intent 中指定待连接 PA 的设备的 deviceId、bundleName 和 abilityName
    - 当连接成功后，通过在 conn 定义的 onAbilityConnectDone 回调中获取对端 PA 的服务代理，两者的连接关系则由 conn 维护
  + 断开远程PA：disconnectAbility(IAbilityConnection conn)
  + 启动远程FA/PA：startAbility(Intent intent)接口提供启动指定设备上FA和PA的能力，Intent 中指定待启动FA/PA的设备的deviceId、bundleName和abilityName
  + 关闭远程PA：stopAbility(Intent intent)
  + 迁移FA：
    - continueAbility(String deviceId)接口提供将本地FA迁移到指定设备上的能力，需要开发者在调用时指定目标设备的 deviceId
    - Ability和AbilitySlice类均需要实现IAbilityContinuation及其方法，才可以实现FA迁移

## 一次开发，多端部署

* HarmonyOS提供了用户程序框架、Ability框架以及UI框架，支持应用开发过程中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
* UI框架支持Java和JS两种开发语言，并提供了丰富的多态控件，可以在手机、平板、智能穿戴、智慧屏、车机上显示不同的UI效果
* 采用业界主流设计方式，提供多种响应式布局方案，支持栅格化布局，满足不同屏幕的界面适配能力

## 统一OS，弹性部署

* 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
* HarmonyOS支持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，降低硬件设备的开发门槛
* 支持各组件的选择（组件可有可无）
  + 根据硬件的形态和需求，可以选择所需的组件
* 支持组件内功能集的配置（组件可大可小）
  + 根据硬件的资源情况和功能需求，可以选择配置组件中的功能集
  + 例如选择配置图形框架组件中的部分控件
* 支持组件间依赖的关系（平台可大可小）
  + 根据编译链关系，可以自动生成组件化的依赖关系
  + 例如选择图形框架组件，将会自动选择依赖的图形引擎组件等