OpenHarmony

开放原子开源基金会 OpenHarmony开发者大会 2023

OpenHarmony分布式数据管理能力构建与技术演进

李有福

OpenHarmony数据管理架构师 / UCLA计算机博士

OpenHarmony

目录 Contents

分布式数据OpenHarmony3.2能力概览

- **目标**：构建轻富设备归一、便捷、高效、安全的全局数据管理能力
- **总结**："深淘滩，低作堰"，夯实基础能力，同步提升2C&2D体验。
  - 分布式对象：自由流转、自由播控，代码量**减70%**，服务流转时延从**1.8秒->1秒内**；
  - 分布式数据库：自由共享(剪切板、图库等)，**代码量减50%**；基于关系表的数据直通，图库元数据**同步性能提升35%**；
  - 数据共享：设置应用数据托管，单应用**常驻内存减18MB**，服务端(Data ShareExtension)开发**代码量减45%**；

目标值	用户体验点			
	内存	高效	便捷	安全
1、提供数据库**存储/同步分离能力**，减少数据管理服务内存消耗，提升存储访问效率。	√	√		
2、提供优化数据**有序上线同步，优选通道，以及低功耗同步**的竞争力机制，提升分布式数据性能和体验。		√		
3、**数据库备份恢复、重建**等DFX可靠性措施，加固数据库健壮性。				√
4、**分布式RDB远程查询**，支撑分布式媒体库构筑跨设备增、删、改、查的媒体元数据的能力。			√	
5、提供分布式**数据对象缓存能力**，支撑跨设备迁移/恢复应用；			√	
6、数据共享**支持跨设备访问**，支持**对接非关系型数据库**，降低分布式媒体库及未来跨设备应用间分享的开发难度。			√	
7、通过统一数据跨应用访问代理的方式，实现**静默数据共享**访问，避免频繁拉起数据源应用，以获得整备功耗收益。	√		√	

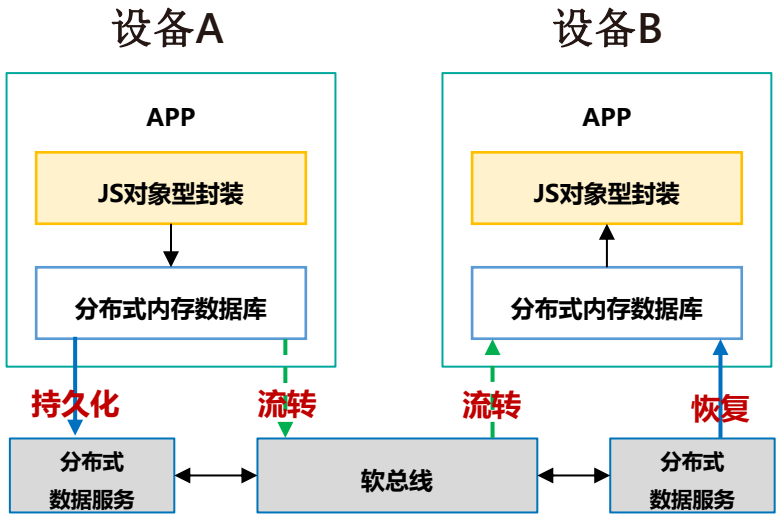
分布式数据OpenHarmony3.2能力概览

- 数据管理接口**API9合计开放接口330+**，使能数据管理基础能力。
- 涵盖本地存取、数据共享、数据同步等能力接口，支撑系统应用（APP）、系统服务（SA）、三方生态等数据管理需求。

目录 Contents

分布式对象 -- 高效的跨设备数据协同

- 分布式数据对象提供跨端内存数据同步，提升开发效率，降低开发门槛
  - 支持应用状态使用分布式数据库进行持久化。
  - 持久化数据支持同步跨设备恢复应用状态。
  - 基于软总线，支持数据对象跨设备协同，无需等待同步。
  - 支持订阅/取消订阅状态变化接口。



俄罗斯方块小程序（流转功能）

自由流转场景，**代码量下降70%，流转时延从1.8s降至<1s。**

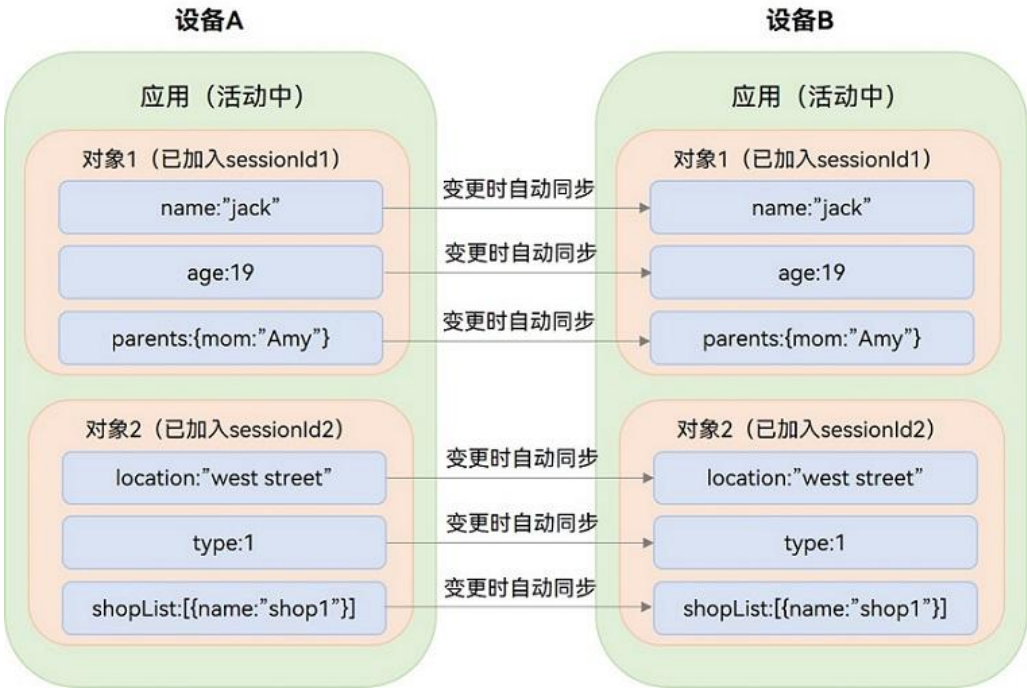
分布式对象 -- 高效的跨设备数据协同

设备A

```
import distributedObject from '@ohos.data.distributedDataObject'
var g_object = distributedObject.createDistributedObject({
name:“Jack", age:19, parents:“mom:'Amy'"}); //1. 创建对象
g_object.setSessionId("123456"); //2. 设置sessionID为“123456”
changeCallback : function (sessionId, changeData) {//3. 设置监听对象变更的
回调
   if (changeData != null && changeData != undefined) {
      changeData.forEach(element => {
         console.info("changed !" + element + " " + g_object[element]);
      });
   }
}
```

设备B

```
var g_object = distributedObject.createDistributedObject({
name:"Amy", age:undefined, parents:undefined}); //1. 创建对象
g_object.setSessionId(“123456”); //2. sessionId为“123456”，已有设备A的对象
statusCallback : function (sessionId, networkid, status) {//3. 监听状态变更
   if (status == "online" && networkid == networkid_A) {
      console.info ("age = {g_object.age}");
      g_object.name = “jack";
   }
}
g_object.on("status", this.statusCallback);
// 保存数据对象，如果应用退出后组网内设备需要恢复对象数据时调用
g_object.save(”local”).then((result) => {
  console.info(”Save object successfully!”);
});
```



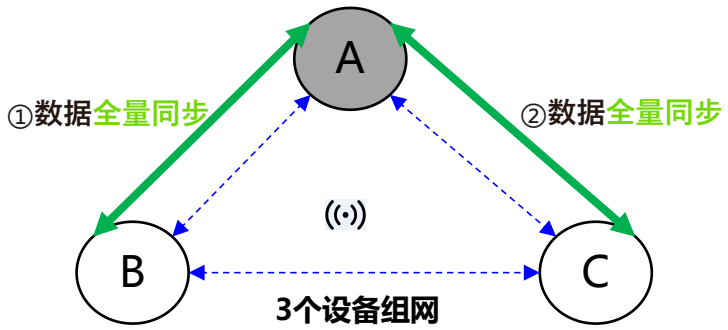
设备B输出：age = 19

设备A输出：changed !name jack

分布式数据同步优化 -- 兼顾性能与功耗

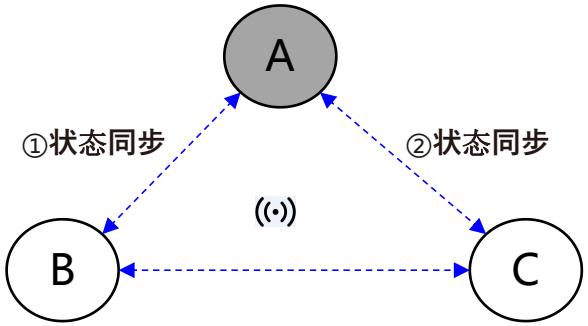
- 根据数据量选择传输通道、数据状态与同步分离、低功耗状态同步机制

设备首次上线



1. 设备首次上线，设备间进行全量数据同步；
2. 联合软总线，根据**数据量**选择合适**传输通道**；

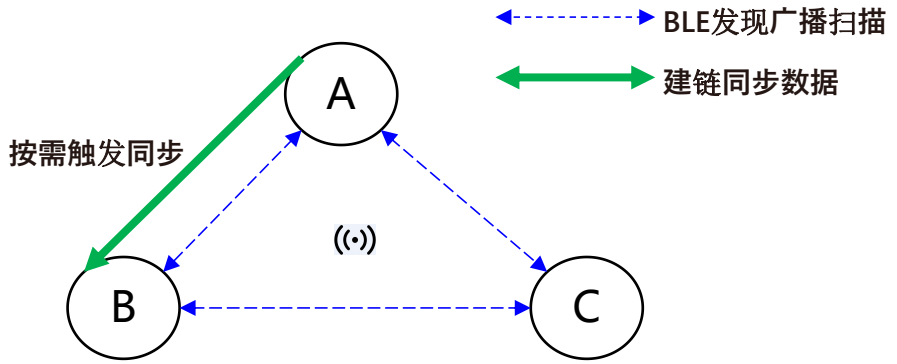
设备非首次上线、数据变更



Consistency : < 5分钟

1. 上线时、跟随**BLE广播**数据状态，但不同步数据；
2. 数据变更时，按需同步数据变更状态；

业务使用/需要时



Consistency : <1秒

1. 根据数据状态，按需触发数据同步；
2. 在条件合适的时机下，主动触发数据同步。

分布式数据同步优化 -- 兼顾性能与功耗

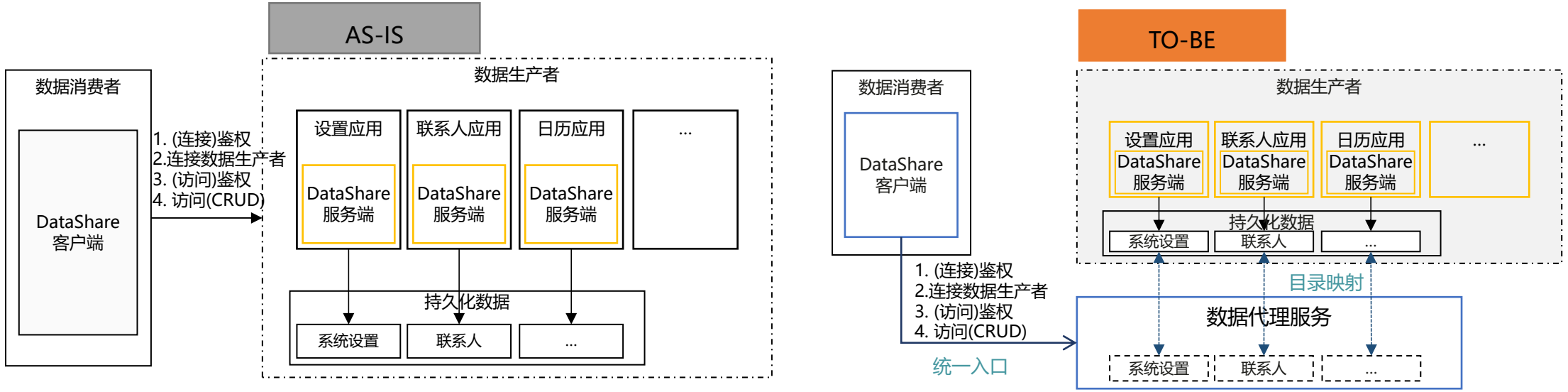
- 低功耗数据状态管理机制，数据状态矩阵和信息同步
- 基于BLE心跳广播的状态同步
  - 数据状态嵌入在心跳广播信息中；
  - 设备上线：数据状态在新设备和在线设备(包括唤醒和待机设备)之间同步；
  - 数据变更：在源设备和在线设备之间；
- 由于使用心跳广播，一致性最大可能达到(1个或多个)心跳周期（5分钟），对于一致性要求高的业务，需要主动建链发送。

同步的状态信息
（设备 & 业务维度）

	Bit0: DP	Bit1: BMS	Bit2: DHMS	Bit3: 元数据	Bit4: 缩率图	日历	联系人	备忘录	剪切板	通知	...
设备A	1	0	0	1	0						
设备B	1	1	0	1	0						
设备C											
...											

静默数据代理 -- 高效的跨应用数据分享

- 业务痛点：
  - 内存消耗：数据源应用在共享数据期间，若应用进程常驻，则持续消耗系统内存。
  - 频繁启动：用户访问相册或使用三方应用访问图库，则会频繁拉起媒体库服务。（大数据显示：跨应用访问数据频繁拉起进程，平均83次/人天）
- 关键措施：
  - 统一数据访问的入口：通过URI扩展标识访问模式，允许访问统一走数据托管服务;
  - 数据持久化文件映射：将应用的数据持久化文件映射到数据托管服务下，允许数据托管服务进行访问；
  - 原数据共享通道选用：保留使用原数据共享通道，以便处理通用CRUD外的其它数据操作。



静默数据代理 -- 高效的跨应用数据分享

- 静默数据访问借助数据管理服务通过目录映射方式直接读取数据提供方的配置，按规则进行预处理访问。
- 数据访问方如果使用静默数据访问方式，URI需严格按照如下格式：
  datashare:///{bundleName}/{moduleName}/{storeName}/{tableName}?Proxy=true
- Proxy=true表示通过静默方式访问数据不拉起数据提供方，如果没有此项则会拉起数据提供方。
- 数据管理服务根据bundleName判断数据提供方应用；读取配置，进行权限校验并访问对应数据。

```
import UIAbility from '@ohos.app.ability.UIAbility';
import dataShare from '@ohos.data.dataShare';
import dataSharePredicates from '@ohos.data.dataSharePredicates';
let dseUri = ('datashare:///com.samples.datesharetest/entry/dbo/table1?Proxy=true');
let dsHelper;
let abilityContext;
export default class EntryAbility extends UIAbility {
    onWindowStageCreate(windowStage) {
        abilityContext = this.context;
        dataShare.createDataShareHelper(abilityContext, dseUri, (err, data) => { dsHelper = data;
        });
    }
}
dsHelper.insert(...);
dsHelper.update(...);
dsHelper.query(...);
dsHelper.delete(...);
```

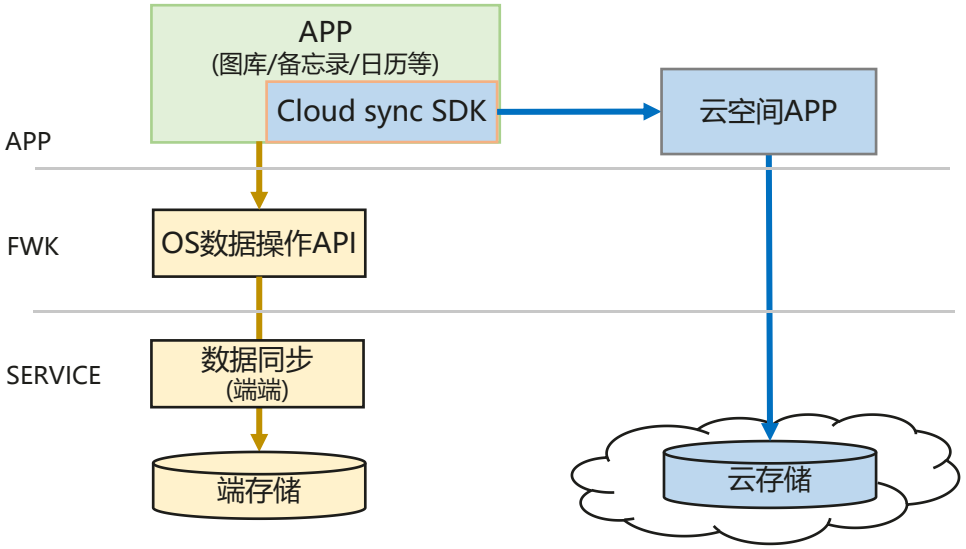
实践案例：设置应用内存减少18M，服务端(DataShareExtension)开发代码量减45%。

OpenHarmony

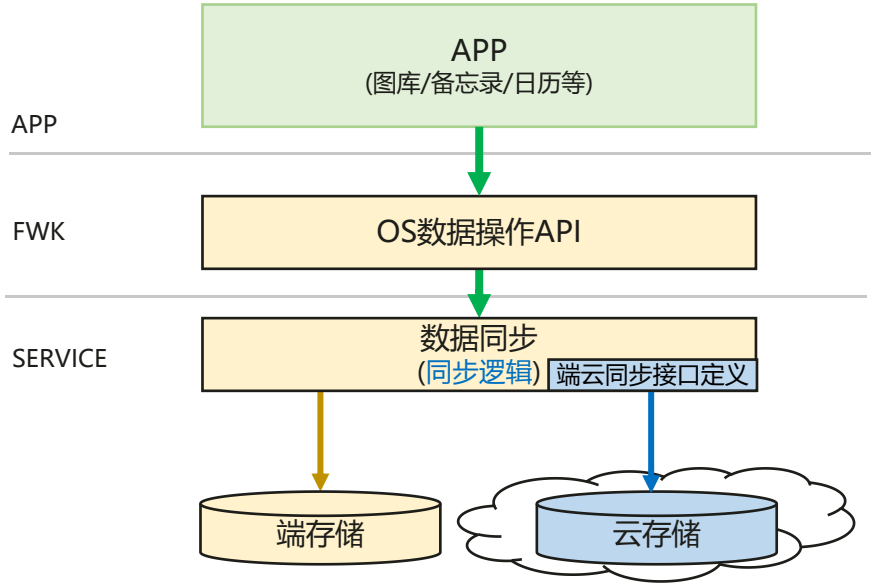
目录 Contents

端云协同：端云同步逻辑从APP下沉到OS

现状分析



OpenHarmony4.0目标



价值	AS IS	TO BE
2C	1. 端云、多端**同步不及时**（依赖App在前台主动触发）； 2. **分布式受限于近场**； 3. 图库应用自行接入云空间，**其他应用只能访问本地的图片**；	1. 端云同步系统托管，**数据上云应用免唤醒**，同步更及时、杜绝不一致； 2. **分布式能力延伸到远场**（如：多人协作）； 3. 系统（媒体库）接入云空间，**三方应用可以直接访问云空间图片**；
2D	1. 开发者**显式集成SDK**（云盘SDK、云同步SDK）； 2. 需额外**处理端云同步、多端同步**逻辑； 以备忘录APP为例，集成云空间SDK方式需要**3000+代码，5人月工作量**。	1. 云同步能力**下沉到系统数据管理框架**，APP不用再显式集成云空间/云同步SDK，也不用**显式处理云同步逻辑**； 2. **开发效率同比提升90%，以备忘录为例：5人月->0.5人月;**

OpenHarmony

THANK YOU

扫描二维码 关注官方公众号

【官网网址】www.openharmony.cn