HUAWEI

< HDC.Together >

华 为 开 发 者 大 会 2 0 2 0

分布式数据管理

管理跨设备共享数据

HUAWEI

介绍

案例

API

全场景多设备数据管理

Matebook

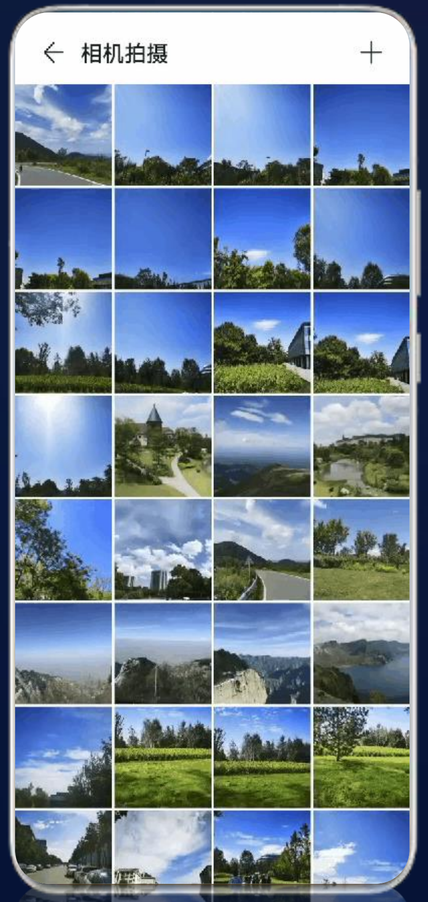
PAD

P40 pro

Mate30 Pro

智慧屏

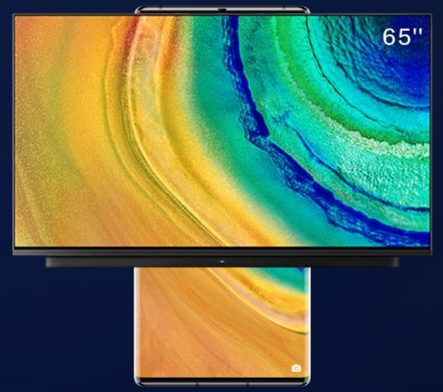
家庭出游场景下照片数据流转



❶ 本机浏览

家庭出游场景下照片数据流转

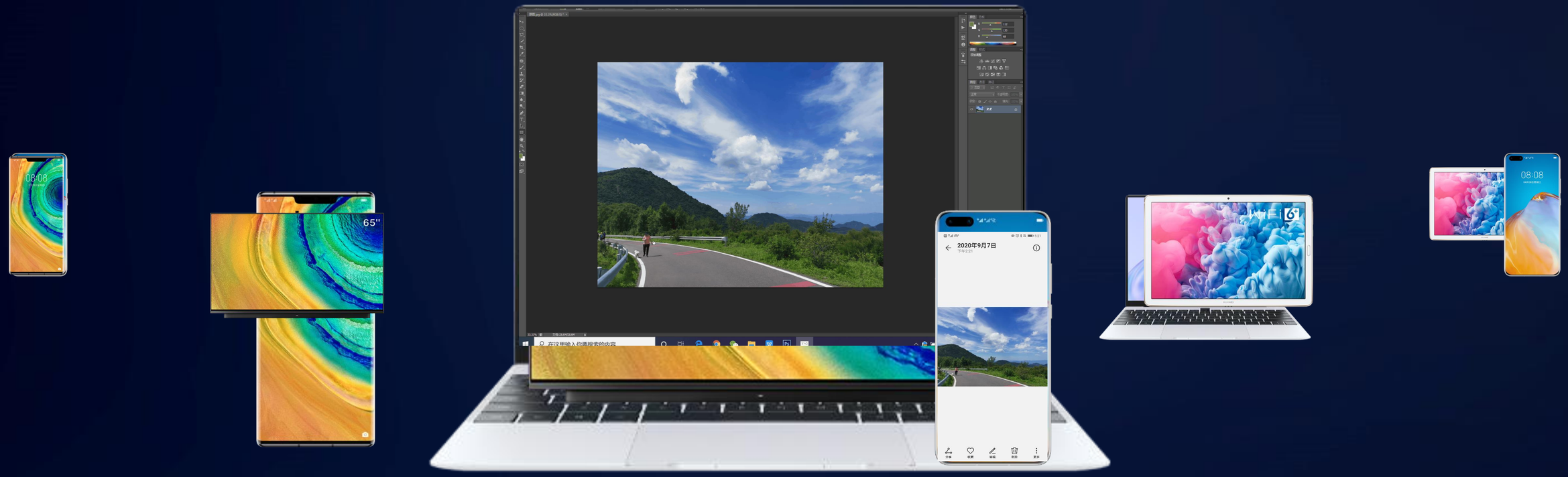
❷ 爸爸近场浏览



家庭出游场景下照片数据流转

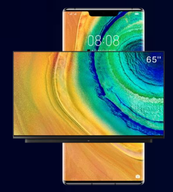
❸家中爷爷奶奶浏览并保存了照片

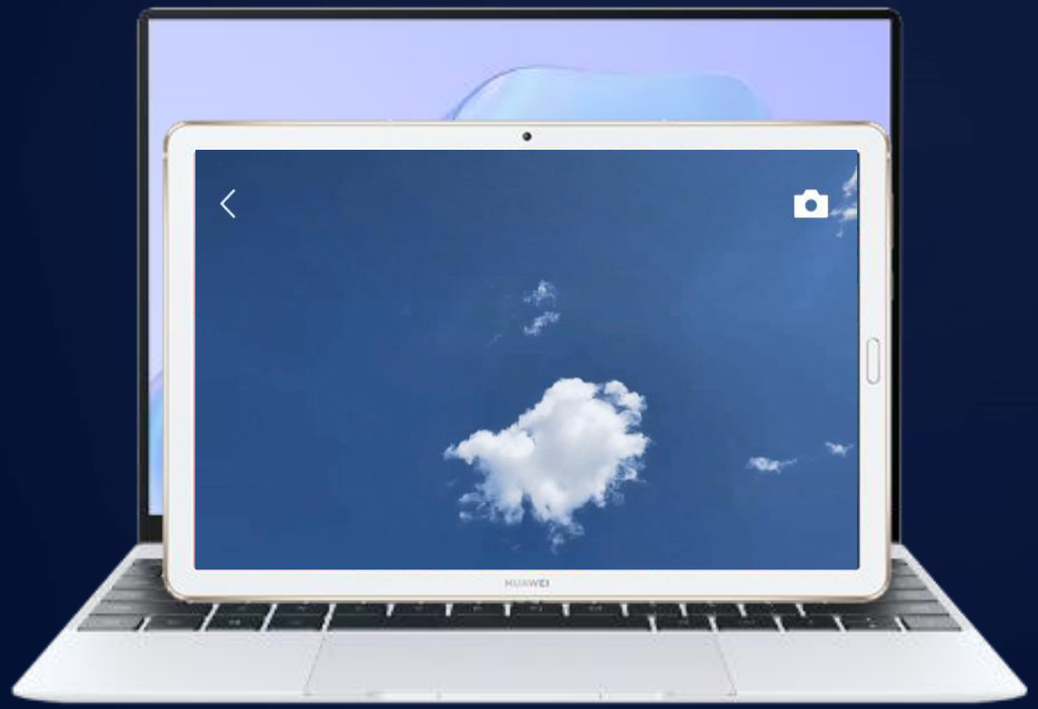
家庭出游场景下照片数据流转



④ 在PC端编辑手机照片

家庭出游场景下照片数据流转





❺ 发了条朋友圈

HUAWEI

Matebook

PAD

P40 pro

Mate30 Pro

智慧屏

如何快速实现用户全场景多设备数据流转和管理的需求?

如何快速实现用户
全场景多设备数据流转和管理的需求?

怎么存储?

- 存储位置
- 多副本
- 存储安全
- 存储空间
- 组织存储
- ...

怎么共享?

- 设备认证
- 近场共享
- 远程共享
- 数据安全
- 用户隐私
- ...

怎么访问?

- 数据一致性
- 性能体验
- 快速查找
- 鉴权控制
- 设备连接
- ...

开

HUAWEI

如何快速实现用户
全场景多设备数据流转和管理的需求?

多设备数据安全

多设备间数据同步

跨设备数据查找和访问

以云为中心的多设备数据安全：难度大、成本高

HUAWEI

设备认证：

```
// 构建认证请求
RequestBody requestBody = new
RequestBody(account，password);
// 发送认证请求
Request request = new Request(loginURL，requestBody);
// 获取请求结果并处理
httpClient.newCall(request).enqueue( new Callback() {
    public void onFailure(Call call, IOException e) {…}
    public void onSuccess(Call call, Response response) {…}
});
```

传输安全：

```
// 生成公钥
RSAPublicKey pubKey = RsaUtil.getPublicKey(modulus,
publicExponent);
// 传输前加密
String cipherText = RsaUtil.encryptByPublicKey(content, pubKey);
// 传输
sendData(cipherText);
// 传输后解密
RSAPrivateKey privateKey = (RSAPrivateKey)kp.get("privateKey");
String content = RsaUtil.decryptByPrivateKey(cipherText,
privateKey);
```

存储安全：

```
// 存储加密
EncryptionRequest request = new EncryptionRequest();
EncryptionClient client = new EncryptionClient(key);
client.setEncryption(request);
// 对称加密
算法：DES、3DES、AES
// 非对称加密
算法：RSA、ECC
```

密钥管理：

```
// 生成密钥 & 更新密钥
Key key = client.createKey();
key.updateLifeCyclePolicy();
// 数字签名
算法：RSA、DSA、ECDSA
```

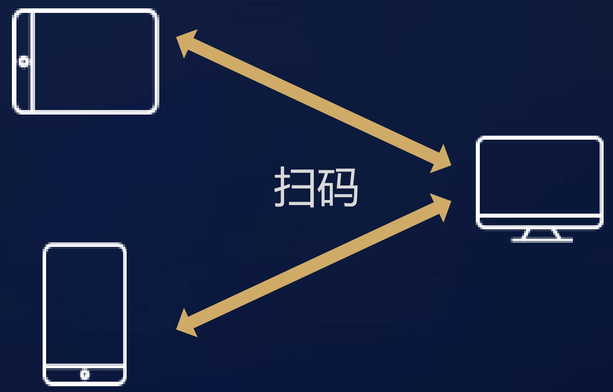
设备之间建立可信认证连接

同账号设备自动连接

照片/视频等用户数据

应用沙箱数据

账号无关设备连接



应用沙箱数据

设备认证

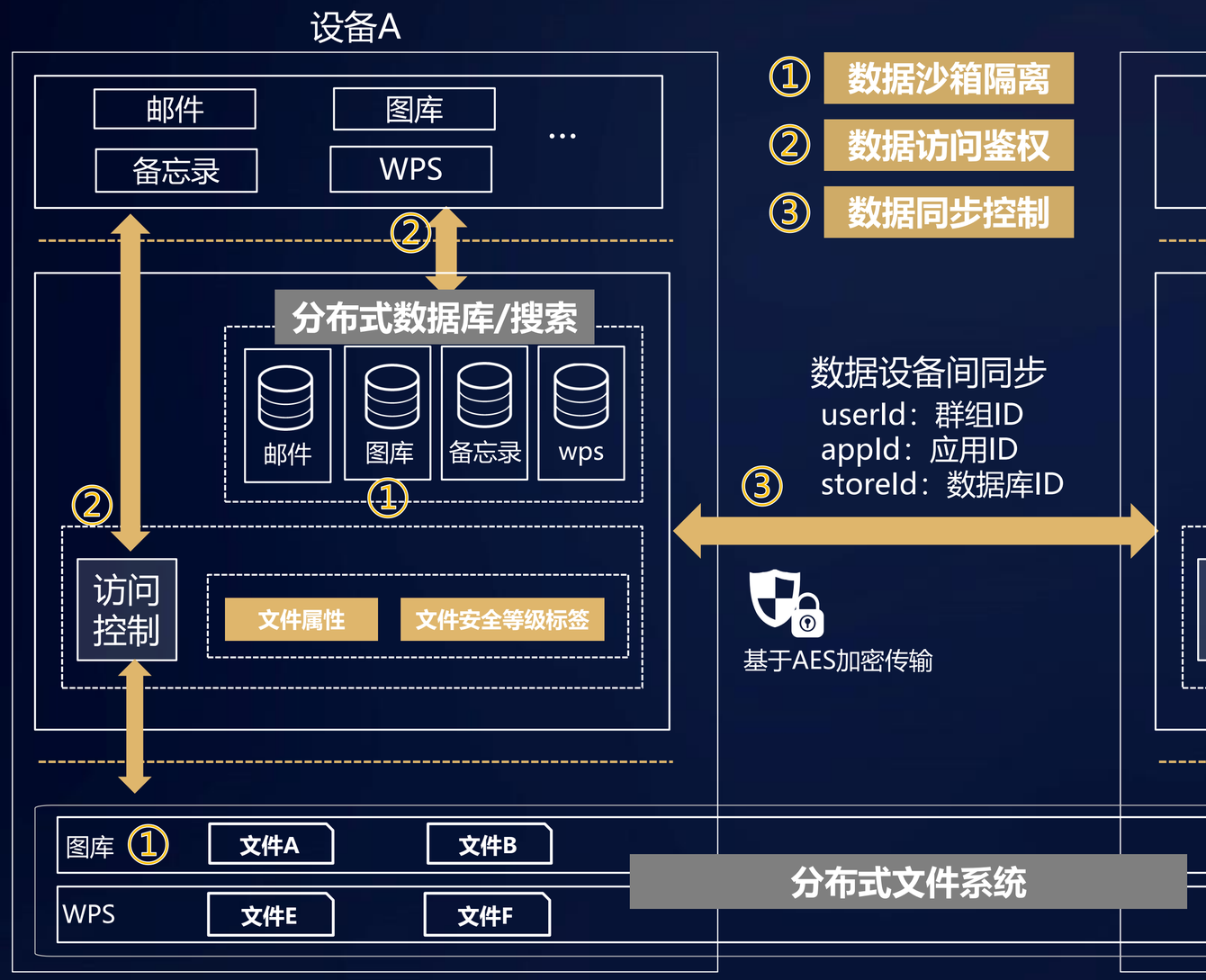
密钥管理

会话密钥协商

数据隔离与访问控制

- 正确的人安全访问正确设备的数据

数据的隔离与同步访问控制



① 数据沙箱隔离

② 数据访问鉴权

③ 数据同步控制

数据设备间同步

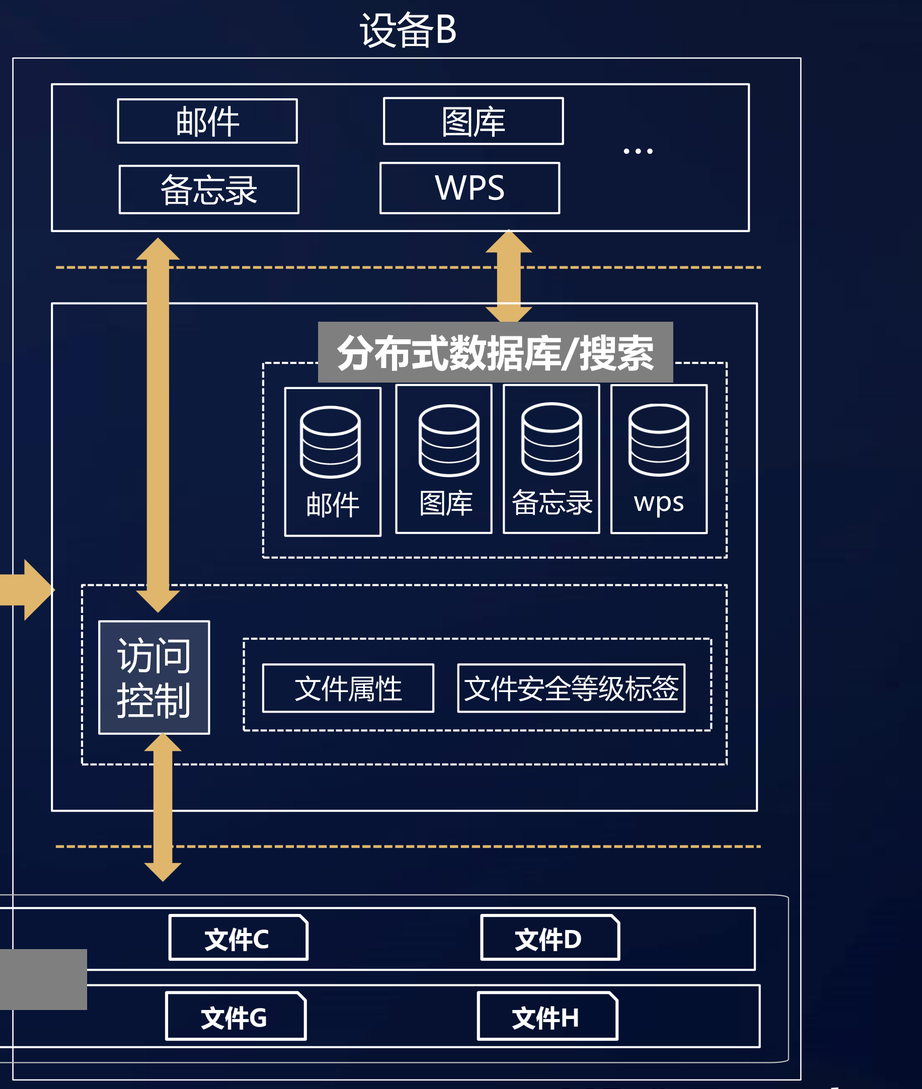
userId：群组ID

appId：应用ID

storeId：数据库ID

③

基于AES加密传输



提供系统级分布式数据安全能力

- 应用只需聚焦于业务逻辑开发，无需额外开发

HUAWEI

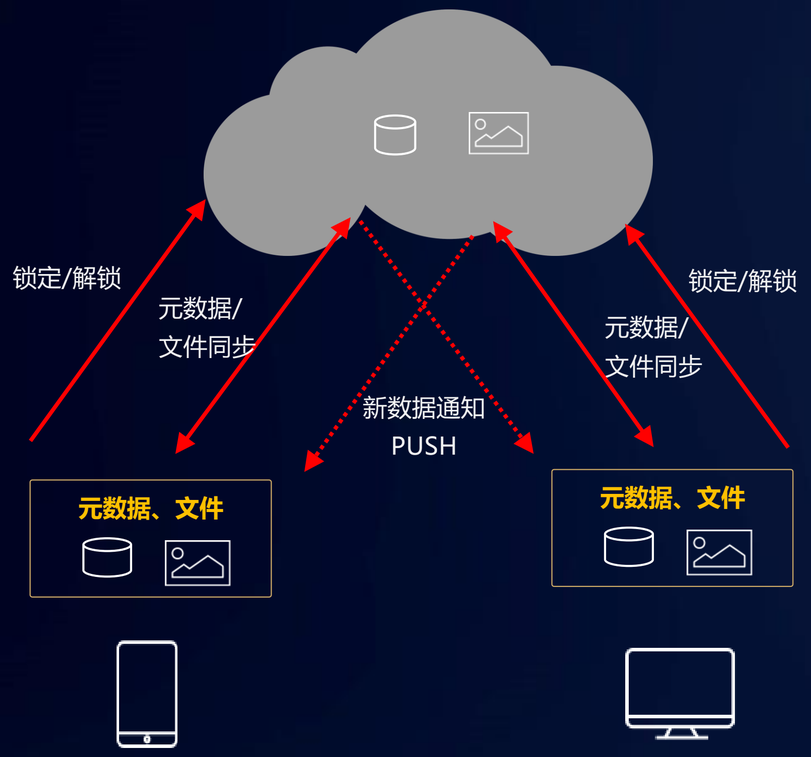
如何快速实现用户
全场景多设备数据流转和管理的需求?

多设备数据安全

多设备间数据同步

跨设备数据查找和访问

以云为中心的数据同步：技术门槛高，实现步骤多



Step1：获取同步锁，进行多设备互斥

```
response = httpClient.newCall(LOCK_URL, put("acquired")).execute();
```

Step2：下载云侧元数据

```
response = httpClient.newCall(META_URL).execute();
metadataCloud = gson.fromJson(response.body().string());
```

Step3：云侧元数据与本机合并

```
FOR fileName IN metadataLocal AND NOT IN metadataCloud： INSERT TO uploadList;
FOR fileName IN metadataCloud AND NOT IN metadataLocal： INSERT TO downloadList, metadataLocal;
FOR fileName IN metadataLocal AND MARKED DELETE： INSERT TO deleteList;
DELETE deleteList  FROM metadataLocal;
```

Step4：更新云侧元数据

```
httpClient.newCall(META_URL, post(gson.toJson(metadataLocal))).execute();
```

Step5：文件端云同步

```
FOR fileName IN downloadList： obsObject = obsClient.getObject(filename)； WRITE obsObject TO_DISK;
FOR fileName IN uploadList： obsClient.putObject(fileName； FROM_DISK);
FOR filename IN deleteList： obsClient.deleteObject(fileName); DELETE filename IN_DISK;
```

Step6：释放同步锁

```
response = httpClient.newCall(LOCK_URL, put("released")).execute();
```

分布式数据库技术

代码示例

```
// 创建分布式数据库，设置成自动同步
KvStore kvStore = kvManager.getKvStore(new Options().setAutoSync(true), FileMetaDB);
// 订阅数据库变更通知;
kvStore.subscribe(SubscribeType.SUBSCRIBE_TYPE_REMOTE, kvObserver);

// 写入一条数据;
kvStore.putString(fileKey, filePath);
// 感知对端数据变化，业务逻辑处理;
KvStoreObserver kvObserver = new KvStoreObserver() {
    @Override
    public void onChange(ChangeNotification changeNotification) {
        // 业务处理逻辑
    }
}
```

- 接口本地/远程访问透明，便捷高效；
- 同步冲突自动解决，保证数据最终一致性。

数据同步方案对比：6->2，简洁高效

HUAWEI

如何快速实现用户
全场景多设备数据流转和管理的需求?

多设备数据安全

多设备间数据同步

跨设备数据查找和访问

现有跨设备数据查找和访问：文件需要先整体下载，用户体验差

微信聊天记录迁移、备份

① 设备扫描、发现和配对

```
Set<BluetoothDevice> pairedDevices = bluetoothAdapter
                .startDiscovery()
                .getBondedDevices();
//MAC address
String deviceHardwareAddress = device.getAddress();
```

② 创建socket传输通道

```
//客户端发起连接
BluetoothSocket clientSocket = device
        .createRfcommSocketToServiceRecord(MY_UUID);
clientSocket.connect();
//服务端接收连接请求
BluetoothServerSocket serverSocket = bluetoothAdapter
.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);
BluetoothSocket socket = serverSocket.accept();
```

③ 收发二进制数据

```
InputStream inputStream = socket.getInputStream();
OutputStream outputStream = socket.getOutputStream();
byte[] buffer = new byte[1024];
int bytes = inputStream.read(buffer);
outputStream.write(buffer);
```

④ 数据转换

```
//定义数据传输协议，校验接收数据合法性
checkBufferInvalid(buffer);
//将接收到的二进制数据转化为数据对象
DataObject object = transformBufferToObject(buffer);
//将数据对象存储到数据库中
saveDataObjectToDatabase(object);
//将二进制数据存储到文件中
saveBufferToFile(buffer);
```

分布式文件系统技术


应用


应用

POSIX

File Ext API

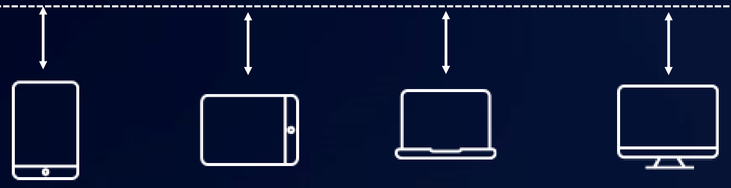
HarmonyOS分布式文件系统

Cache 引擎

元数据管理

通信组件

分布式文件融合统一视图



代码示例

```
//设备A写入:
File distDir = context.getDistributedDir();      // 获取应用分布式目录
File distFile = new File(distDir, "hello.jpg"); // 创建分布式文件对象
FileOutputStream stream = new FileOutputStream(distFile);
stream.write(...);                                // 写分布式文件
```

```
//设备B读取:
File distDir = context.getDistributedDir();      // 获取应用分布式目录
File distFile = new File(distDir, "hello.jpg"); // 创建分布式文件对象
FileInputStream stream = new FileInputStream(distFile);
stream.read(...);                                 // 读分布式文件
```

- 接口简单，兼容POSIX，应用可无缝迁移；
- 屏蔽储存位置差异，数据随人不随设备。

分布式搜索技术

代码示例

```
// 更新应用数据索引到统一索引库
searchAbility.update(groupId, pkgName, indexData);


// 发起搜索
searchAbility.beginSearch(groupId, pkgName);

// 联想搜索
searchSession.groupSearch(queryJsonStr, 10);

// 分页搜索
searchSession.search(queryJsonStr, 0, 8);

//...
```

- 统一索引管理，无需关注索引引擎实现；
- 搜索接口丰富，使用高效。

分布式数据库性能

指标项	性能
get	2967 OPS
put	1165 OPS
delete	1519 OPS
update	1242 OPS
E2E同步	25ms

来源于华为终端实验室测试数据

测试环境

- HUAWEI mate 30 pro
- 跨进程调用
- 10组，每组1000次测试

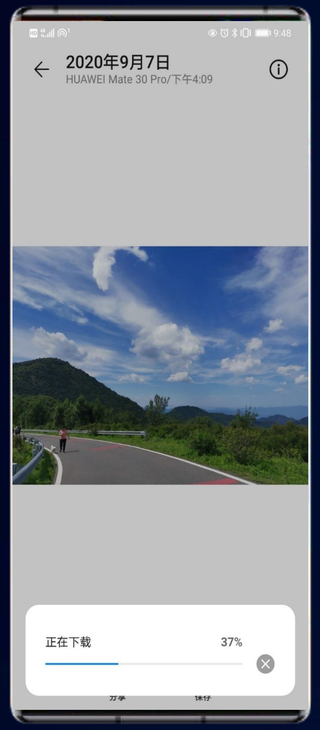
单次测试

- 测试数据key大小为100B，value大小为1KB

E2E同步只需25ms

备注：相关数据来源于华为实验室，不同环境与场景下可能存在差别，仅供参考

分布式文件系统性能



指标项	性能
随机读	1.4MB/s
随机写	14.63MB/s
顺序读	14.48MB/s
顺序写	16.09MB/s

来源于华为终端实验室测试数据

顺序读写速率可达网络带宽的
90%以上

测试环境

- HUAWEI P40 pro
- 网络带宽：模拟用户带宽场景，平均网速16.5MB/s
- 测试10组数据，取平均值

测试工具

- 标准fio工具

备注：相关数据来源于华为实验室，不同环境与场景下可能存在差别，仅供参考

分布式搜索性能

指标项	性能
索引	1000条1K数据共72个字段需1.65s
搜索	10000条检索平均时延21ms

来源于华为终端实验室测试数据

测试环境

- 10组，每组1000次测试
- HUAWEI mate 30 pro

单次测试

- 1000条索引、1k/条、72个字段
- 基于10000索引，返回50条目

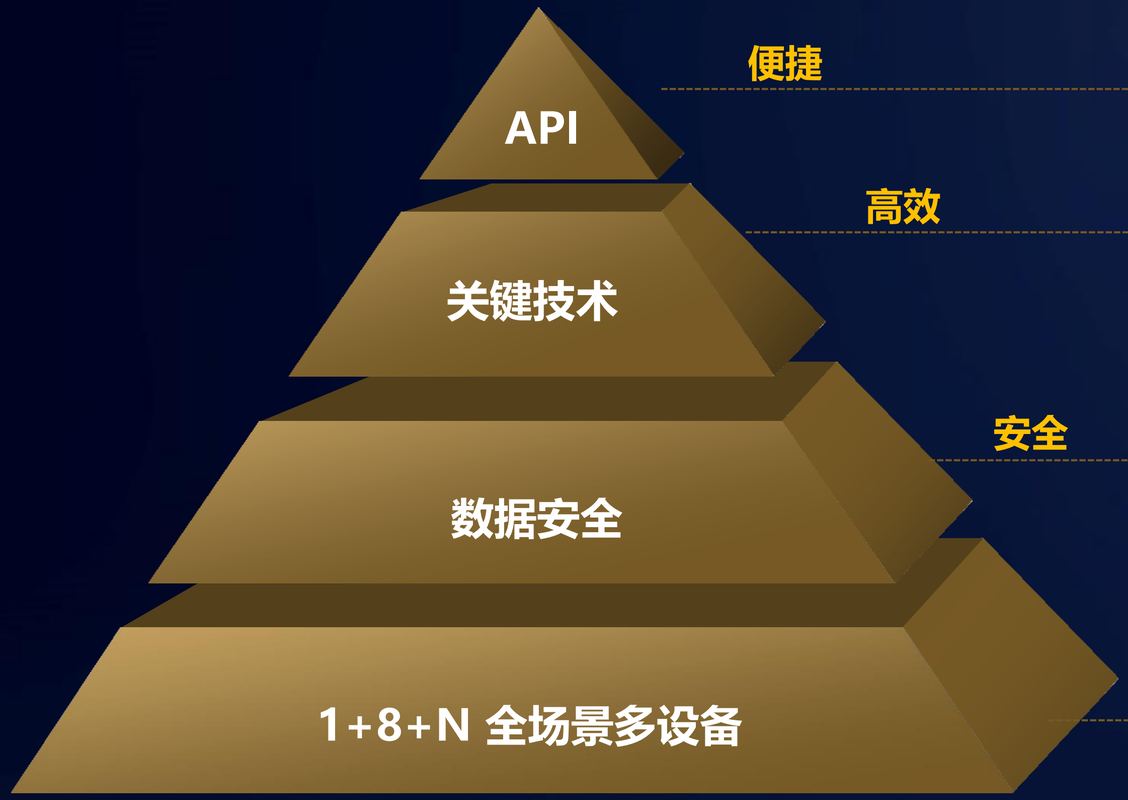
搜索平均时延只需21ms

备注：相关数据来源于华为实验室，不同环境与场景下可能存在差别，仅供参考

HarmonyOS 分布式数据管理平台

HUAWEI

分布式数据管理是HarmonyOS分布式技术栈基础功能之一，为应用程序和用户提供更加便捷、高效和安全的数据管理能力。



便捷

高效

安全

分布式数据库

分布式文件系统

分布式搜索

存储安全

传输安全

使用安全

密钥管理

HarmonyOS 分布式数据管理核心特征

HUAWEI

- 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

HUAWEI

介绍

案例

API

分布式数据管理面向开发者API

HUAWEI

分布式数据库	分布式文件系统	分布式搜索
数据库初始化	分布式沙箱目录获取	索引构建
数据插入、更新、删除	文件创建、删除、拷贝	索引变化订阅
关系型语义查询	遍历目录、属性查询	联想搜索
滑动窗口查询	文件读、写	分页、分组搜索
数据变化监听	文件锁	高频词搜索
批量操作	扩展属性查询	

API直通车 ——https://developer.harmonyos.com/cn/docs/documentation/doc-guides/database-relational-overview-0000000000030046

使用HarmonyOS分布式数据管理带来的收益

HUAWEI

便捷、高效、安全

降低开发成本
Reduced Programming Costs

不需要分布式系统中远程设备上的应用重复编写设备连接、安全认证和数据传输程序。

本地/远程透明
Local/Remote Transparency

应用不用关心数据存储在哪个物理设备上，通过一套API实现分布式网络中所有设备的数据访问和操作。

更好的资源管理
Better Resource Management

高效利用系统资源（网络IO，磁盘IO，CPU），实现数据的同步和传输。

减少数据冗余
Reduced Data Redundancy

数据可以只保存在一个物理位置，不需要每个设备都有一份数据。

数据完整性
Data Integrity

确保数据不会被意外销毁或者更改，并且不会因为并发用户之间的冲突而丢失更新。

HUAWEI

欢迎关注HarmonyOS开发者微信公众号