**媒体扫描模块设计方案**

1. **系统U盘插入扫描媒体流程**



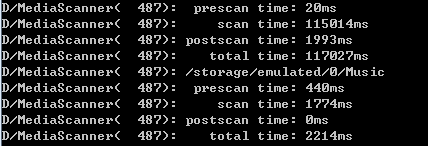
* 第一部分处理插入U盘Kernel检测425ms
* 第二部分处理挂载并发送挂载完成的广播952ms
* 第三部分U盘扫描

第一、第二部分我们很难再做优化，优化的效果也有限。我们主要进行第三部分的优化。

1. **U盘扫描详细流程**



**时间开销2000图片 500音乐：**

****

第一次扫描：117秒

拔插U盘第二次扫描：2.2秒

prescan：确认旧数据库文件是否还存在当前U盘中，不存在则删除对应数据。

scan：全盘扫描媒体文件，扫描到一个文件立即解析大小、ID3等信息，500条记录后入库。

postscan：最后记录入库，清理用不到的缩略图。

1. **优化分析**

从上面时间消耗可以看出时间主要花在了解析媒体上，减少重复解析媒体文件将能大大加快扫描速度。同时缩小扫描范围，加快数据库的读取都将能加快扫描速度。下面是分析扫描和数据库操作的时间：

1、数据库插入单条记录时间：21毫秒  
2、数据库插入1000条记录时间：14118毫秒，采用事务插入1000条记录时间：1019毫秒  
3、java扫描文件时间，30个文件夹，3层目录，共2000文件：1203毫秒  
4、jni c++扫描文件时间，30个文件夹，3层目录，共2000文件：347毫秒  
5、2000个文件路径，c++确认文件是否存在，文件最后是否修改时间： 253毫秒

结论：1、数据库插入采用事务插入将能省很多时间。

2、扫描文件采用C++来扫描比java扫描快很多。

3、确认文件是否存在耗时也是比较短。

优化策略：

1. 减小扫描范围。
2. 减少重复解析媒体文件。
3. 采用事务来处理数据库大量操作。
4. 扫描文件的过程在C++层处理。
5. 及时上报通知界面刷新。
6. **方案分析设计**
7. **减小扫描范围**
8. Internal数据库只扫描内存/media目录下的媒体文件，主要是存放铃声、闹钟、提醒声音。
9. External数据库只扫描U盘目录。实现方式：把U盘固定挂载在/udisk目录下，每次只扫描这个路径即可。
10. **减少重复解析媒体文件**
11. 增加多U盘支持。当插入不同的U盘时，把旧数据库保存。当再次插入之前插入过的U盘时，调取之前旧数据库。

实现方式：

1. MediaProvider封装数据库选择细节，内部选用哪个数据库由MediaProvider创建不同的DatabaseHelper，DatabaseHelper指定数据库文件。对外部来说调用方式是不变的。
2. 数据库以internal\_xxxx(USB ID)命名，当数据库插入不同的U盘时生成不同的数据库文件。
3. 当数据库数据超过3个时，扫描结束后把最久没有访问的数据删除。

相关类图：

对外：都是通过MediaProvider操作数据库。

对能：生成不同的DatabaseHelper将能操作不同数据库。

操作时序图：



当插入不同的U盘时，重复上面的操作，使得DatabaseHelper对象被替换，指向不同的数据库文件。外部使用MediaProvider操作数据库不受影响。

1. **采用事务来处理大量数据库操作**

文件扫描完成后先在内存中缓存，当扫描到文件的数量达到500个时，再采用事务处理，执行插入数据操作。

1. **扫面过程在C++层处理**

具体执行扫描过程通过JNI在C++层处理，扫描得到的数据再通过JNI回调给Java层进行数据库操作和通知上层应用。

1. **及时上报通知界面刷新**
2. MediaScannerService aidl增加注册监听接口addScanListener(Handler handler，int type)。
3. MediaScannerService 由Handler生成Messanger，插入监听者列表。
4. MediaScannerService 增加3个监听者列表ArrayList<Messanger>，对应音乐、视频、图片。
5. MediaScannerService 扫描到一个文件即可通过监听者列表Messanger上报给应用程序。
6. 应用退出时调removeScanListener(Handler handler，int type)取消监听。

时序图如下：



考虑到消息的实时性，扫描模块扫描到数据后实时通知应用。应用应自己控制好界面的刷新间隔。

**五、总体方案流程**

流程图如下：



时序图：



改进后从U盘到播放出音乐应该在2.5秒左右，待实际的验证。

从U盘到列表扫描完成（2000个文件）应该在2秒左右。