Android 内存优化指南

镁佳科技 - v1.0

变更记录

版本号 Rev.	日期 Effective Date	变更描述 Description	编制人 Editor
0.1	2022/1/24	初稿	马强
1.0	2022/10/10	正式发布	

1. 总则

提到内存优化，网络上经常有容易混淆的两个概念：

- RAM优化：程序在运行时，占用的内存的大小
- ROM优化：程序存储在磁盘上，占用的磁盘大小

这两个优化在概念上并不是彼此完全隔离开的，例如：减少程序在磁盘上占用空间的大小，一般也可以减少程序运行时在内存中的大小。本文档主要以 **Android** 系统为基础，尤其是 **Java** 层程序为优化重点，从程序运行时占用内存大小的角度来描述，部分优化做法同样减少程序存储在磁盘上的大小，在文档中不特别标注和区分。

1.1 问题概述

内存问题一般表现为三种

- 内存抖动：内存波动图形呈锯齿状、GC导致卡顿。此问题在Dalvik虚拟机比较明显，在现在ART虚拟机的情况下，出现此问题情况较少。
- 内存泄漏：程序的内存占用图形一直增长，一般是因为持有对象不释放或不能按照对象正常的生命周期进行释放。
- 内存溢出：即OOM，程序使用的内存超出了系统限定的运行内存，OOM时会导致程序异常。

1.2 优化的方法

内存优化从软件流程上可以分为三个阶段：

- 设计阶段优化：此阶段主要是在编码开始前，根据需求和经验划定系统中需要的各个程序，以及各个程序可以使用的内存大小，各个程序在编码开始前根据要求，设计程序架构，从架构角度减少不必要内存消耗，满足系统对于应用的内存要求。此阶段一般通过系统裁减和内存规划来实施和改进。
- 编码阶段优化：此阶段是在编码时，提高代码质量，减少内存消耗，主要通过资源优化和代码优化的方式来减少内存占用，此阶段一般通过同行代码评审来提高代码质量。
- 测试阶段优化：此阶段是在编码完成后，使用各种内存工具，对软件进行检测，找出并修复内存抖动和内存泄漏的问题，此阶段一般通过各种内存分析工具来实施和改进。

2. 设计阶段优化

2.1 系统裁减

系统裁减是确认只有系统使用到的程序和库才编译进ROM，不在需求范围内的程序和与此程序有关的资源和库等，都不能出现在系统的ROM中。系统裁减使用如下的方法：

1. 检查安装的APK信息，这些APK确实是系统需要的。

a. 查看编译 out 目录下 /system/app/，/system/priv-app/，/system/app/，/system/priv-app/，/data/app/ 等目录下的应用，列出的目录仅供参考，不同项目要检查的目录可能不同。
b. 运行系统，通过 adb shell pm list packages 显示所有安装的程序。

2. 检查 init.rc 及相关文件，确保所有的服务和程序都是系统需要的。

2.2 内存规划

内存规划是根据项目的硬件约束，结合需求和经验数据，确定各个子系统以及各子系统内每个程序可以使用的内存大小。各个系统在后续开发时，需要严格按照规定的内存要求进行开发，确保最终开发完成的程序符合内存规划时的要求。

3. 编码阶段优化

3.1 编译优化

Gradle 编译：镁佳的 App 现在已经全部实现了 gradle 编译，在 build.gradle 文件下，增加部分属性来对内存使用进行优化。

```
buildTypes {
    release {
        debuggable false
        zipAlignEnabled true
        shrinkResources true
    }
    debug {
        zipAlignEnabled true
        shrinkResources true
    }
}
```

shrinkResources：配置后自动清理未使用的资源
zipAlignEnabled：配置后可提高系统和应用的运行效率，更快的读取apk中的资源，降低内存的使用

3.2 资源优化

Android Java 程序如果占用内存过大，一般都是资源的不合理使用造成的，通过优化资源可以极大的减少内存的占用。

3.2.1 布局优化

越扁平化的视图布局，占用的内存就越少。分析布局的嵌套情况，减少布局嵌套，推荐使用 ConstraintLayout 布局来减少布局的嵌套。

3.2.2 BitMap使用

- 图片缩放：图片载入内存前，计算合适的缩放比例，通过BitmapFactory.Options.inSampleSize 来加载图片。
- 解码格式：图片载入内存前，选择合适的解码格式 (Bitmap.Config)，通过 BitmapFactory.Options.inPreferredConfig 来加载图片。
- 质量压缩：通过 Bitmap.compress() 压缩图片质量，载入压缩质量后的图片。
- 缓存图片：对于多处使用到的图片，对图片进行缓存避免多处加载同一个图片。使用时要确保不使用的缓存图片及时释放。

3.2.3 绘制替代

可以用绘制替代图片的地方，尽量使用绘制来实现，例如：渐变或者描边的UI效果，可以使用Shape来实现。

3.2.4 存储路径

hdpi/xhdpi/xxhdpi 等文件夹下的图片，当 hdpi 有此图片，但 xhdpi/xxhdpi 无对应图片时，在不同的设备会被拉神，提高内存占用。如不希望图片被拉神，可以选择直接将图片放到 xxhdpi 目录，或者放到 asset/nodpi 目录下。

3.3 代码优化

3.3.1 使用内存优化类

使用内存优化类一般是指，Android或Java提供了内存优化后的类，用这些内存优化类替代非内存优化类来减少内存占用。

- HashMap 替换：在数据量小于 1000 的情况下，请考虑将 HashMap 替换为 SparseArray 系列 (SparseArray, SparseBooleanArray, SparseIntArray, SparseLongArray, LongSparseArray) 以及 ArrayMap。

- HashSet 替换：在数据量小于 1000 的情况下，请考虑将 HashSe t替换为 ArraySet。
- Serializable 替换：Android 提供了 Parcelable 来进行序列化，推荐在内存间数据传输时推荐使用 Parcelable，在需要保存或网络传输数据时可选择 Serializable。
- ListView 替换：列表页展示界面，需要支持动画，或者频繁更新，局部刷新，建议使用 RecyclerView 替换 ListView。
- Service 替换：当服务执行执行完毕后可以退出时，建议使用 IntentService 替换 Service。
- LinearLayout 替换：使用 LinearLayout 进行布局嵌套时，替换成 RelativeLayout 或者 ConstraintLayout 来实现。

3.3.2 编码检查事项

编码检查事项是指：在编程时我们需要关注的编程规范，遵循这些规范会减少程序的内存占用和内存问题的产生。

- 枚举使用：枚举比起使用 static 常量会增加 2 倍以上字节的 APK 大小，5-10 倍的内存占用。如果想实现类似枚举的功能，可以使用 @IntDef 和 @StringDef 这两个注解来实现。
- 资源释放：使用到了BroadcastReceiver，ContentObserver，File，Cursor，Stream，Bitmap等资源需要及时进行释放。
- Message创建：不要通过 new Message() 创建 Message 对象，通过 Handler.obtainMessage() 来获取 Message 对象。
- 集合类使用
    - 集合类仅有元素的添加，没有元素的删除，而这些集合又是全局性的集合时，可能造成内存单向增加。
    - 关注集合类元素的初始容量，例如 ArrayList 初始容量为 10，如果知道使用数据小于 10，则可以直接指定容量进行 ArrayList 的创建。
- 帧动画使用：考虑使用补间动画、属性动画，或者surfaceView来实现“帧动画”
- 字符串拼接：使用 StringBuilder 拼接字符串，减少使用 “+” 进行字符串拼接。
- SharePreference 使用：对于同一个 SharePreference，会将整个 XML 导入到内存中，容易出现为了读取一个配置而将大量数据写入内存，造成内存浪费。尽量减少一个 SharePreference 包含的键值，频繁更改的配置项和不常修改配置项，分为不同的 SharePreference 存储。
- 多进程使用：应用的 activity、service、receiver 和 provider 均支持运行在单独的进程中，但多进程会显著增大内存，需要明确使用多进程的必要性。
- static 使用：Android应用中 static的生命周期和应用的进程一致,需要谨慎使用。
- onTrimMemory() 使用：通过继承 Application，Activity，Fragement，Service，ContentProvider 中 onTrimMemory() 方法，在不同的内存水平下，强制进行内存释放。
- onDraw() 使用：避免在类似 onDraw() 等被频繁调用的函数中创建对象。类似的有 ListView 的 getView() 函数。

- WebView 使用：在应用中只要使用一次 WebView，内存就不会被释放掉。我们可以为 WebView 开启一个独立的进程，使用 AIDL 与应用的主进程进行通信，WebView 所在的进程可以根据业务的需要选择合适的时机进行销毁，达到正常释放内存的目的。

3.4 多用户优化

Android 10 之后默认启动了多用户，启动多用户后，如果 App 不针对多用户进行相应的修改，则可能会在每个用户下都有一份应用的进程，造成内存占用的浪费。为解决这些问题，需要：

1. 确认 App 进程只需要在某一个User下，还是需要多个 User 下同时存在。
2. 多 User 下 App，确认拥有的每一个服务或组件，能否可以只运行在某个 User 下。
3. 创建可以感知多用户环境的 App。

3.4.1 基本使用

1）从传入的 Binder 调用中提取用户句柄：

```
int userHandle = UserHandle.getCallingUserId()
```

2）使用受保护的全新 API 启动特定用户的服务、Activity 和广播：

```
Context.startActivityAsUser(Intent, UserHandle)
Context.bindServiceAsUser(Intent, ..., UserHandle)
Context.sendBroadcastAsUser(Intent, ... , UserHandle)
Context.startServiceAsUser(Intent, ..., UserHandle)
```

UserHandle 可以是显式用户，也可以是以下特殊句柄之一：UserHandle.CURRENT 或 UserHandle.ALL。CURRENT 表示当前位于前台的用户。如果您想向所有用户发送广播，可以使用 ALL。

3）如需与您自己应用中的组件通信，请使用 (INTERACT_ACROSS_USERS) 权限；如需与其他应用中的组件通信，请使用 (INTERACT_ACROSS_USERS_FULL) 权限。

4）您可能需要创建代理组件，这些代理组件在用户进程中运行，之后会访问用户 0 中的 singleUser 组件。

5）使用新的 UserManager 系统服务查询用户及其句柄：

```
UserManager.getUsers()
UserManager.getUserInfo()
UserManager.supportsMultipleUsers()
UserManager.getUserSerialNumber(int userHandle)
```

```
UserManager.getUserHandle(int serialNumber)
UserManager.getUserProfiles()
```

6)注册可借助 ContentObserver和 BroadcastReceiver 上的新 API 监听特定或所有用户以及回调(可提供与回调发起用户相关的其他信息)。

```
MegaContentResolver.registerContentObserver(Uri uri, boolean
    notifyForDescendents,ContentObserver observer,
    @UserIdInt int userHandle)

MegaContext.registerReceiverAsUser(Context context,
    BroadcastReceiver receiver,
    UserHandle user, IntentFilter filter,String broadcastPermission,
    Handler scheduler)
```

7)为了解决后台用户占用资源导致前台用户功能无法使用的问题，可以监听 Intent.ACTION_USER_FOREGROUND 和 Intent.ACTION_USER_BACKGROUND 这两个系统广播

1. 当收到 ACTION_USER_BACKGROUND 的时候，说明用户进程已经在后台运行，需要释放所有有可能和其他 domain 有冲突的资源(如语音助手占用的Mic，Input通道)，并使自己所有的功能处于非活跃状态，所有处理应该交由前台进程处理。
2. 当收到 ACTION_USER_ FOREGROUND 的时候，说明用户进程恢复到前台，这个时候可以重新去获取需要的资源。

3.4.2 使用单例

android:singleUser="true" 加入到服务、接收器或程序的 AndroidManifest.xml 中，则系统只在 U0 下实例化一个服务，接收器或者程序。

3.4.3 停用指定组件

如果应用在 U0 和 U10 等用户下都有服务，接收器，提供者实例，但是只需要在 U0 下运行，可以停用其他用户下的组件，参考代码如下：

```
// Add on all entry points such as boot_completed or other manifest-listed
receivers and providers
if (!UserManager.isSystemUser()) {
    // Disable the service
    ComponentName targetServiceName = new ComponentName(this,
TargetService.class);
    context.getPackageManager().setComponentEnabledSetting(
        targetServiceName, COMPONENT_ENABLED_STATE_DISABLED, 0);
}
```

3.4.4 指定用户进程

首选通过HMI启动服务或者应用。如果系统应用的服务只需要在某个 U0 以外的用户下启动进程，可以通过如下的方法。

1. 应用的 AndroidManifest.xml 中不能有 android:persistent="true"，android:singleUser="true" 和对于系统 Intent 的静态 BroadCastReceiver。
2. 应用在 AndroidManifest.xml 文件中定义特定 BroadCastReceiver。
3. 由 CarController 利用特定的 Intent 和 StartServiceAsUser 方式，启动此服务到指定的 User 下。
4. 应用进程启动之后，通过动态方式注册基于用户的 BroadCastReceiver。

3.4.5 U0 用户显示界面的方法

根据 3.4.6 的方案，可以配置到 U10 下显示界面。如有特殊情况再具体处理。

3.4.6 配置系统应用安装到指定用户下

说明文档位置：frameworks/base/data/etc/preinstalled-packages-platform.xml

大体的修改如下：

1）在 device/mega/(monza/bigsur/jasper/…) 创建 preinstalled-packages-config 目录

2）以 device/mega/monza 为例，在device/mega/monza下放入配置文件 preinstalled-packages-monza.xml，文件内容类似如下：

```
<config>
      <!-- Mega Porting   -->
      <install-in-user-type package="com.mega.map">
      <install-in user-type="FULL" />
      </install-in-user-type>
      <install-in-user-type package="com.voyah.btphone">
      <install-in user-type="FULL" />
      </install-in-user-type>
      <install-in-user-type package="com.voyah.camera">
      <install-in user-type="FULL" />
      </install-in-user-type>
</config>
```

3）修改device/mega/monza/device.mk 增加如下三行：

```
# Default user install
PRODUCT_COPY_FILES += \

$(LOCAL_PATH)/preinstalled-packages-config/preinstalled-packages-monza.xml:sys
```

```
tem/etc/sysconfig/preinstalled-packages-monza.xml
```

各个产品的用户配置主要在步骤2中完成。

4. 测试阶段优化

4.1 常用命令

命令	说明
cat proc/meminfo	此命令查看详细的内存信息，包括总内存，可用内存，空闲内存等信息。
free	轻量级的内存查看工具，内容来源于proc/meminfo。
top	实时显示进程信息。
dumpsys meminfo	显示排序后的每个进程的内存信息，也显示总体内存信息。
pm list packages	显示系统中安装的APK信息。

4.2 测试工具介绍

4.2.1 Memory Profiler

此工具为 Android Studio 自带工具，使用方便，可以

- 实时图表展示应用内存使用量。
- 用于识别内存泄漏、抖动等。
- 提供捕获堆转储、强制GC以及根据内存分配的能力。

依次点击 Android Studio 的 View → Tool Windows → Android Profiler 则可以使用。

4.2.2 Memory Analyzer Tool

此工具为 Java 的内存分析工具，功能强大，可以用来分析 Android App 内存信息，此工具唯一不方便的地方是需要将 Android Studio 导出的内存 .hprof 文件，使用 Android SDK 的工具 (sdk\platform-tools\hprof-conv.exe) 进行转换之后，才能提供给 MAT 进行分析。

4.2.3 LeakCanary

LeakCanary 是 Android 上的一个自动检测内存泄漏的工具，通过集成 LeakCanary 的 jar 包到自己 App 中，添加简单的几行代码，LeakCanary 会自动检测内存泄漏，并将信息展示出来，极大地方便了 Android 应用程序内存泄漏检测。

4.3 多用户测试

4.3.1 测试方法

Android10 之后默认开启了多用户，一个预置在 ROM 中的 APK 有可能在多个用户下都创建了进程。可设置如下的场景，在这些场景下对存在多个用户的进程，逐个分析是否可去掉某个用户下的进程。

- 系统启动后；
- 系统运行一段时间后；
- 设定高负载场景，在高负载场景下运行一段时间后；
- 进行monkey测试，monkey测试一段时间后；

长安项目上的系统启动后场景的分析实例请参考：长安内存优化数据

4.3.2 测试信息记录

- 本测试为仅在 U0，U10 用户下测试，未考虑更多用户情况。
- 普通应用安装于/data目录下，即使有开机启动监听，系统签名，也不会运行于 U0 用户下。
- 监听系统启动的系统应用 system/app (已验证)，system/priv-app (未验证)，开机后会在 U0，U10 用户下启动两个进程。推测通过静态 AndroidManifest.xml 文件注册 BroadCastReceiver 监听系统信息的应用，如果未启动，在收到监听的系统信息后，会在 U0，U10 用户下启动两个进程。

5. 内存优化报告

请使用：Android 内存优化报告 - 模板 来形成内存优化报告

6. 参考文档

关于 Android 内存管理建议阅读三篇入门文章：

- 理论基础：「全面理解Android内存优化 1」-Android的内存机制与管理建议，主要讲解Android性能优化时涉及到的各种基础知识
- 工具使用：「全面理解Android内存优化 2」-内存优化工具的使用，主要讲解Android性能优化时各种常用工具的使用。
- 项目实践：「全面理解Android内存优化 3」-从理论到实践，以一个实际APP为例，总结在在开发中会被忽视的内存问题。

本文涉及的其他参考文档如下：

编号	标题	链接

1	最全的Android内存优化技巧	https://www.jianshu.com/p/51e28a2c609c
2	探索 Android 内存优化方法	https://juejin.cn/post/6844903897958449166
3	深入探索 Android 内存优化（炼狱级别-上）	https://juejin.cn/post/6844904099998089230
4	深入探索 Android 内存优化（炼狱级别-下）	https://juejin.cn/post/6872919545728729095
5	Android内存分析工具	https://www.cnblogs.com/tinylaker/p/10574899.html
6	构建可感知多用户的应用	https://source.android.google.cn/devices/tech/admin/multiuser-apps
7	Bitmap性能优化	https://www.jianshu.com/p/1f85cdb564ad
8	Android代码混淆使用手册	https://www.jianshu.com/p/dbe98916a21c
9	Gradle优化	https://blog.csdn.net/qq_29951983/article/details/112970361
10	【腾讯Bugly干货分享】Android ListView 与 RecyclerView 对比浅析—缓存机制	https://zhuanlan.zhihu.com/p/23339185
11	写给程序员的内存泄漏治理手册	https://www.jianshu.com/p/a4d99014e1a1
12	Android布局优化	https://www.jianshu.com/p/0d235a0cbb48
13	约束布局ConstraintLayout看这一篇就够了	https://www.jianshu.com/p/17ec9bd6ca8a
14	Android studio Memory Profiler简单使用	https://blog.csdn.net/qq_38998213/article/details/97915018
15	Memory Analyzer Tool 使用手记	https://www.iteye.com/blog/wensong-1986449
16	LeakCanary官方文档翻译	https://www.jianshu.com/p/bcaab8f0f280
17	Android LeakCanary使用详细教程	https://blog.csdn.net/chennai1101/article/details/103798870
18	常用注解@Intdef与@Stringdef	https://www.cnblogs.com/sjjg/p/4523457.html
19	Android多用户下数据隔离方案与常见问题解决思路	https://www.jianshu.com/p/1308c94b713e
20	Android多用户处理及相关广播发送	https://blog.csdn.net/u012758497/article/details/90084618
21	高效加载大型位图	https://developer.android.com/topic/performance/graphics/load-bitmap