### 阿里巴巴 Glide对Bitmap的缓存与解码复用如何做到的

本专栏专注分享大型Bat面试知识，后续会持续更新，喜欢的话麻烦点击一个star

Glide 使用简明的流式语法API，大多数情况下，可能完成图片的设置你只需要： Glide.with(activity) .load(url) .into(imageView);

默认情况下，Glide 会在开始一个新的图片请求之前检查以下多级的缓存：1. 活动资源 (Active Resources) 2. 内存缓存 (Memory Cache) 3. 资源类型（Resource Disk Cache）4. 原始数据 (Data Disk Cache)活动资源：如果当前对应的图片资源正在使用，则这个图片会被Glide放入活动缓存。内存缓存：如果图片最近被加载过，并且当前没有使用这个图片，则会被放入内存中资源类型: 被解码后的图片写入磁盘文件中，解码的过程可能修改了图片的参数(如:inSampleSize、inPreferredConfig)原始数据: 图片原始数据在磁盘中的缓存(从网络、文件中直接获得的原始数据)

在调用into之后，Glide会首先从Active Resources查找当前是否有对应的活跃图片，没有则查找内存缓存，没有则查找资源类型，没有则查找数据来源。

![](data:text/html;base64...)

相较于常见的内存+磁盘缓存，Glide将其缓存分成了4层。

#### 第一层 活动资源

当需要加载某张图片能够从内存缓存中获得的时候，在图片加载时主动将对应图片从内存缓存中移除，加入到活动资源中。 这样也可以避免因为达到内存缓存最大值或者系统内存压力导致的内存缓存清理，从而释放掉活动资源中的图片(recycle)。 活动资源中是一个”引用计数"的图片资源的弱引用集合。

因为同一张图片可能在多个地方被同时使用，每一次使用都会将引用计数+1,而当引用计数为0时候，则表示这个图片没有被使用也就是没有强引用了。这样则会将图片从活动资源中移除，并加入内存缓存。

![](data:text/html;base64...)

#### 第二层 内存缓存

内存缓存默认使用LRU(缓存淘汰算法/最近最少使用算法),当资源从活动资源移除的时候，会加入此缓存。使用图片的时候会主动从此缓存移除，加入活动资源。

LRU在Android support-v4中提供了LruCache工具类。

![LruCache.png](data:text/html;base64...)

LruCache.png

构造LinkedHashMap的accessOrder设置为true。在使用的此map的时候，自动进行排序(每次get/put,会将使用的value放入链表header头部)。LruCache会在每次get/put的时候判断数据如果达到了maxSize,则会优先删除tail尾端的数据。

![LruCache.png](data:text/html;base64...)

LruCache.png

磁盘缓存同样使用LRU算法。

#### 第三、四层 磁盘缓存

**Resource** 缓存的是经过解码后的图片，如果再使用就不需要再去进行解码配置(BitmapFactory.Options),加快获得图片速度。比如原图是一个100x100的ARGB\_8888图片，在首次使用的时候需要的是50x50的RGB\_565图片，那么Resource将50x50 RGB\_565缓存下来，再次使用此图片的时候就可以从 Resource 获得。不需要去计算inSampleSize(缩放因子)。 **Data** 缓存的则是图像原始数据。

# Bitmap复用

如果缓存都不存在，那么会从源地址获得图片(网络/文件)。而在解析图片的时候会需要可以获得BitmapPool(复用池)，达到复用的效果。

![复用前](data:text/html;base64...)

复用前

![复用之后](data:text/html;base64...)

复用之后

复用效果如上。在未使用复用的情况下，每张图片都需要一块内存。而使用复用的时候，如果存在能被复用的图片会重复使用该图片的内存。 所以复用并不能减少程序正在使用的内存大小。Bitmap复用，解决的是减少频繁申请内存带来的性能(抖动、碎片)问题。

<https://developer.android.google.cn/topic/performance/graphics/manage-memory.html>

![](data:text/html;base64...)

![](data:text/html;base64...)

![img](data:text/html;base64...)

img

Google给出的案例可以看出: 使用方式为在解析的时候设置Options的inBitmap属性。

1. Bitmap的inMutable需要为true。
2. Android 4.4及以上只需要被复用的Bitmap的内存必须大于等于需要新获得Bitmap的内存，则允许复用此Bitmap。
3. 4.4以下(3.0以上)则被复用的Bitmap与使用复用的Bitmap必须宽、高相等并且使用复用的Bitmap解码时设置的inSampleSize为1，才允许复用。

因此Glide中，在每次解析一张图片为Bitmap的时候(磁盘缓存、网络/文件)会从其BitmapPool中查找一个可被复用的Bitmap。

BitmapPool是Glide中的Bitmap复用池,同样适用LRU来进行管理。 当一个Bitmap从内存缓存 **被动** 的被移除(内存紧张、达到maxSize)的时候并不会被recycle。而是加入这个BitmapPool，只有从这个BitmapPool **被动** 被移除的时候,Bitmap的内存才会真正被recycle释放。