

一种图片加载方法及装置

技术领域

本申请涉及图片处理技术领域，尤其涉及一种图片加载方法及装置。

背景技术

对于安卓设备来说，在进行图片加载时，由于解码图片需要占用应用进程大量的 Java 堆内存，且，安卓系统的应用进程都有一定大小限制，因而当应用进程的 Java 堆内存使用率较高时，极易出现堆内存不足的现象，从而触发 OOM（Out Of Memory，内存耗尽）。

为了解决这一问题，目前可采用图片缓存的方式来减少图片解码，以实现解码所得图片数据的复用，进而降低 Java 堆内存的使用率。例如，在从网络或本地资源解码出 Bitmap（位图）对象后，可通过使用 LinkedHashMap（基于链表、哈希实现的 Map 数据结构）实现 LRU（Least Recently Used，近期最少使用算法）缓存管理，即，将最近使用到的 Bitmap 对象用强引用保存起来（保存到 LinkedHashMap 中），当缓存数量达到预定值的时候，将不经常使用的 Bitmap 对象删除，从而实现对 Bitmap 对象的复用，提高用户的应用体验。

但是，由于在采用现有的图片缓存方式来实现解码所得图片数据的复用时，仍是在 Java 层解码出 Bitmap 对象，并将解码所得的 Bitmap 对象缓存在 Java 堆内存，即，缓存的图片数据所占用的内存仍为 Java 堆内存，从而导致 Java 堆内存的压力仍较大，在进行图片的解码时，仍极易出现堆内存不足的现象，从而触发 OOM。

发明内容

本申请实施例提供了一种图片加载方法及装置，用以解决采用现有的图片缓存方式来实现解码所得图片数据的复用时，由于 Java 堆内存的压力仍较大从

而极易触发 OOM 的问题。

一方面，本申请实施例提供了一种图片加载方法，包括：

在对图片进行加载时，判断 Native 堆内存中是否缓存有该图片的解码数据；

若是，则从 Native 堆内存中获取该图片的解码数据，并基于获取到的解码数据实现该图片的加载；

若否，则从存储有该图片的存储空间处获取并加载该图片，并将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中。

另一方面，本申请实施例提供了一种图片加载装置，包括：

判断单元，用于在对图片进行加载时，判断 Native 堆内存中是否缓存有该图片的解码数据；

执行单元，用于若确定 Native 堆内存中缓存有该图片的解码数据，则从 Native 堆内存中获取该图片的解码数据，并基于获取到的解码数据实现该图片的加载；否则，从存储有该图片的存储空间处获取并加载该图片，并将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中。

本申请有益效果如下：

本申请实施例提供了一种图片加载方法及装置，可将解码所得的图片数据缓存在本机堆内存，即 Native 堆内存中，以便图片加载时复用，这样，由于 Native 堆内存不计算在应用进程的 Java 堆内存内，因而可有效降低 Java 堆内存的压力，降低 OOM 的触发概率，同时，还可达到更好地利用堆内存实现图片缓存，以提高图片加载效率并减少系统垃圾回收，即系统 GC 的效果。

附图说明

为了更清楚地说明本申请实施例中的技术方案，下面将对实施例描述中所需要使用的附图作简要介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域的普通技术人员来讲，在不付出创造性劳动的前提下，

还可以根据这些附图获得其他的附图。

图 1 所示为本申请实施例一提供的图片加载方法的一种可能的流程示意图；

图 2 所示为本申请实施例一提供的图片加载方法的另一种可能的流程示意图；

图 3 所示为本申请实施例二提供的图片加载装置的一种可能的结构示意图。

具体实施方式

由于对于 Android 系统来说，堆内存可分为 Java 堆内存和 Native 堆内存，且，Native 堆内存是不计算在 Java 堆内存内的，因而，在本申请所述实施例中，可基于这一点将解码所得的图片数据缓存在 Native 堆内存中，以便图片加载时复用，这样，由于 Native 堆内存不计算在应用进程的 Java 堆内存内，因而可有效降低进程的堆内存压力，降低 OOM 的触发概率，同时，还可达到更好地利用堆内存实现图片缓存，即更好地实现图片解码数据的复用的效果，以提高图片加载效率并减少系统 GC，进而提升系统性能。

为了使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请作进一步地详细描述，显然，所描述的实施例仅仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本申请保护的范围。

实施例一：

本申请实施例一提供了一种图片加载方法，所述图片加载方法可适用于任何基于 Android 系统的客户端或终端设备，对此不作限定。具体地，如图 1 所示，所述图片加载方法可包括以下步骤：

步骤 101：在对图片进行加载时，判断 Native 堆内存中是否缓存有该图片的解码数据，若判断结果为是，则执行步骤 102，否则，执行步骤 103。

可选地，图片的解码数据通常为对该图片进行解码所得到的 Bitmap 对象的像素数组数据。另外，由于在本申请所述实施例中，针对任一图片，当将该图片的解码数据缓存到 Native 堆内存中时，通常会在 NativeCacheInfo（本机堆内存缓存记录信息）中生成一对应的记录数据，因而，判断 Native 堆内存中是否缓存有该图片的解码数据，可执行为：

判断是否能够从 NativeCacheInfo 中，获取到与该图片的解码数据相对应的记录数据，若判断结果为是，则确定 Native 堆内存中缓存有该图片的解码数据，否则，则确定 Native 堆内存中未缓存有该图片的解码数据。

其中，所述 NativeCacheInfo 中的每一记录数据可包括与该记录数据相对应的图片的宽、高、NativePointer（本机堆内存指针，用于指明该记录数据所对应的解码数据在 Native 堆内存中的存储地址）、Bitmap.Config（用于指明图片的每个像素所占用的内存大小）、解码数据的数据长度等属性信息。另外，为了便于查找与记录，每一记录数据通常可通过能够唯一标识该记录数据对应的图片的标识信息（或能够唯一标识该记录数据对应的图片的解码数据的标识信息）进行标记，对此不作赘述。

再有，由于 NativeCacheInfo 所占用的内存通常较小，因而，可将 NativeCacheInfo 存储于 Java 堆内存中（当然，为了进一步降低 Java 堆内存的存储压力，还可将 NativeCacheInfo 存储于 Native 堆内存中）；并且，可采用 LRU 对 NativeCacheInfo 中的各记录数据进行管理，以便达到对 NativeCacheInfo 中的各记录数据以及堆内存中的图片缓存数据进行实时维护，在对应淘汰相应的记录数据时将 Native 堆内存中的相应解码数据所占用的内存同步释放，以进一步降低堆内存压力，减少 OOM 概率的效果。

另外，需要说明的是，在本申请所述实施例中，还可通过其它方式来判断 Native 堆内存中是否缓存有该图片的解码数据，如，通过对 Native 堆内存中缓存的数据进行逐条扫描的方式，来判断 Native 堆内存中是否缓存有该图片的解码数据，对此不作限定。

步骤 102: 从 Native 堆内存中获取该图片的解码数据, 并基于获取到的解码数据实现该图片的加载。

可选地, 如图 2 所示, 步骤 102 所述的从 Native 堆内存中获取该图片的解码数据, 并基于获取到的解码数据实现该图片的加载, 具体可包括以下步骤:

步骤 102A: 判断 ImageView (画布) 当前显示的 Bitmap 对象是否满足该图片的解码数据所需的重用条件, 若满足, 则执行步骤 102B, 若不满足, 则执行步骤 102C。

可选地, 对于 Android 4.4 以下版本的系统来说, 当 ImageView 当前显示的 Bitmap 对象所对应的图片的宽、高、Bitmap.Config, 与该图片的解码数据所对应的图片的宽、高、Bitmap.Config 完全一致, 且, ImageView 当前显示的 Bitmap 对象的属性为 mutable (即, 可修改) 时, 可认为 ImageView 当前显示的 Bitmap 对象满足该图片的解码数据所需的重用条件, 否则, 则认为其不满足该图片的解码数据所需的重用条件;

而对于 Android4.4 及以上版本的系统来说, 当 ImageView 当前显示的 Bitmap 对象所对应的图片的宽、高、每个像素所占用的内存三者的乘积不小于该图片的解码数据的数据长度时, 可认为 ImageView 当前显示的 Bitmap 对象满足该图片的解码数据所需的重用条件, 否则, 则认为其不满足该图片的解码数据所需的重用条件。

步骤 102B: 将 Native 堆内存中缓存的、对该图片进行解码所得到的 Bitmap 对象的像素数组数据拷贝到 ImageView 当前显示的 Bitmap 对象中, 得到新的 Bitmap 对象, 并将新的 Bitmap 对象渲染到 ImageView 中, 以实现该图片的加载。

也就是说, 当 ImageView 上已经显示有 A 图 (Bitmap-A) 时, 若需要显示 B 图 (Bitmap-B), 则无需将 ImageView 上显示的 A 图移除, 再换上 B 图, 而是, 只要将 B 图和 ImageView 重新绑定, 把 B 图的像素数组数据拷贝到 A 图上即可, 以提高图片加载的效率。

可选地，为了防止 ImageView 中的老数据有可能干扰新数据的显示，在将该图片的 Bitmap 对象的像素数组数据拷贝到 ImageView 当前显示的 Bitmap 对象中之前，还可首先对 ImageView 当前显示的 Bitmap 对象中的像素数组数据进行清除，对此不作赘述。

步骤 102C：从 ReusableBitmapPool（位图文件池）中获取一符合该图片的解码数据所需的重用条件的 Bitmap 对象、或者创建一符合该图片的解码数据所需的重用条件的 Bitmap 对象，并将 Native 堆内存中缓存的、对该图片进行解码所得到的 Bitmap 对象的像素数组数据拷贝到获取到的或者创建的 Bitmap 对象中，得到新的 Bitmap 对象，并将新的 Bitmap 对象渲染到 ImageView 中，以实现该图片的加载。

可选地，如图 2 所示，在执行步骤 102C 之前，还可首先将 ImageView 当前显示的 Bitmap 对象存放至 ReusableBitmapPool 中（对应步骤 102C'），以便其它图片加载时复用（因为虽然该当前 Bitmap 对象不一定能满足当前待加载图片的复用条件，但其很有可能能满足其它图片的复用条件），以进一步提高图片加载的效率并减少系统的 GC。

其中，所述 ReusableBitmapPool 通常可位于 Java 堆内存中、且其引用方式为软引用，以便可以在内存不紧张时，有效地利用原有资源，减少 GC，也可以在系统需要内存时，及时释放，不会导致 OOM。另外，为了降低堆内存的存储压力，ReusableBitmapPool 中存储的各 Bitmap 对象还可为清除了像素数组数据的 Bitmap 对象，即为一个可被其它图片复用的 Bitmap 对象框架。

进一步地，需要说明的是，在执行步骤 102C 的过程中，可首先判断是否能够从 ReusableBitmapPool 中获取一符合该图片的解码数据所需的重用条件的 Bitmap 对象，若是，则无需再执行创建一符合该图片的解码数据所需的重用条件的 Bitmap 对象的操作（即，可在确定为否时，再执行创建一符合该图片的解码数据所需的重用条件的 Bitmap 对象的操作），以进一步提高图片加载效率并减少系统的 GC。

可选地，通常来说，通过调用系统 API 并传入要显示的图片的宽、高、Bitmap.Config 即可创建一符合该图片的解码数据所需的重用条件的 Bitmap 对象，对此不作赘述。

再有，需要说明的是，为了防止老数据有可能干扰新数据的显示，在将该图片的 Bitmap 对象的像素数组数据拷贝到获取到的 Bitmap 对象或创建的 Bitmap 对象中之前，若确定获取到的 Bitmap 对象或创建的 Bitmap 对象中的像素数组数据不为空，则也可首先对获取到的 Bitmap 对象或创建的 Bitmap 对象中的像素数组数据进行清除，对此也不作赘述。

进一步可选地，如图 2 可知，在从 Native 堆内存中获取该图片的解码数据之前，所述方法还可包括以下步骤：

步骤 104：判断 Native 堆内存中缓存的该图片的解码数据是否有效，并确定判断结果为是。

即，通常可在判定 Native 堆内存中缓存的该图片的解码数据有效时，再从 Native 堆内存中获取该图片的解码数据。

可选地，判断 Native 堆内存中缓存的该图片的解码数据是否有效，可执行为：

判断 NativeCacheInfo 中的、与该图片的解码数据相对应的记录数据中的 NativePointer 是否为有效指针，若判断结果为是，则确定 Native 堆内存中缓存的该图片的解码数据有效。

另外，由图 2 可知，若确定 Native 堆内存中缓存的该图片的解码数据无效，则可直接跳转至步骤 103，即执行从存储有该图片的存储空间处获取并加载该图片，并将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中的操作。

进一步地，在从 Native 堆内存中获取该图片的解码数据之前，所述方法还可包括以下步骤：

若确定 ImageView 当前显示的 Bitmap 对象所对应的图片非首次加载，则

将 ImageView 中当前显示的 Bitmap 对象的像素数组数据取出并缓存到 Native 堆内存中，以便下一次显示时复用。

另外，如前文所述，当将 ImageView 中当前显示的 Bitmap 对象的像素数组数据缓存到 Native 堆内存中时，可在 NativeCacheInfo 中生成一对应的记录数据，对此不作赘述。

步骤 103：从存储有该图片的存储空间处获取并加载该图片，并将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中。

可选地，存储有该图片的存储空间可为本地磁盘缓存、本地文件或者网络等；且，对获取到的图片进行加载可包括对获取到的图片进行解码得到 Bitmap 对象，并对 Bitmap 对象进行渲染等操作。

另外，如前文所述，当将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中时，也可在 NativeCacheInfo 中生成一对应的记录数据。

至此，即可完成当前待加载图片的加载操作。

由本申请实施例所述内容可知，在本申请所述实施例中，可将解码所得的图片数据缓存在本机堆内存，即 Native 堆内存中，以便图片加载时复用，这样，由于 Native 堆内存不计算在应用进程的 Java 堆内存内，因而可有效降低 Java 堆内存的压力，降低 OOM 的触发概率，同时，还可达到更好地利用堆内存实现图片缓存，以提高图片加载效率并减少系统 GC 的效果。

实施例二：

基于与本申请实施例一中的图片加载方法同样的发明构思，本申请实施例二提供了一种图片加载装置，所述图片加载装置的具体实施可参见上述方法实施例一中的相关描述，重复之处不再赘述，具体地，如图 3 所示，所述图片加载装置可包括：

判断单元 31，可用于在对图片进行加载时，判断 Native 堆内存中是否缓存有该图片的解码数据；

执行单元 32，可用于若确定 Native 堆内存中缓存有该图片的解码数据，

则从 Native 堆内存中获取该图片的解码数据,并基于获取到的解码数据实现该图片的加载;否则,从存储有该图片的存储空间处获取并加载该图片,并将加载该图片的过程中所得到的该图片的解码数据缓存到 Native 堆内存中。

可选地,所述判断单元 31 具体可用于判断是否能够从本机堆内存缓存记录信息中,获取到与该图片的解码数据相对应的记录数据,若判断结果为是,则确定 Native 堆内存中缓存有该图片的解码数据。

可选地,所述执行单元 32 具体可用于在判定 Native 堆内存中缓存的该图片的解码数据有效时,则从 Native 堆内存中获取该图片的解码数据。

可选地,所述执行单元 32 具体可用于判断本机堆内存缓存记录信息中的、与该图片的解码数据相对应的记录数据中的本机堆内存指针是否为有效指针,若判断结果为是,则确定 Native 堆内存中缓存的该图片的解码数据有效;

其中,所述本机堆内存缓存记录信息中的每一记录数据中的本机堆内存指针用于指明该记录数据所对应的解码数据在 Native 堆内存中的存储地址。

另外,需要说明的是,所述本机堆内存缓存记录信息可位于 Java 堆内存中,且,所述本机堆内存缓存记录信息中的各记录数据采用 LRU 进行管理。

可选地,该图片的解码数据通常可为对该图片进行解码所得到的 Bitmap 对象的像素数组数据;

所述执行单元 32 具体可用于判断 ImageView 当前显示的 Bitmap 对象是否满足该图片的解码数据所需的重用条件;若满足,则将 Native 堆内存中缓存的、对该图片进行解码所得到的 Bitmap 对象的像素数组数据拷贝到 ImageView 当前显示的 Bitmap 对象中,得到新的 Bitmap 对象,并将新的 Bitmap 对象渲染到 ImageView 中,以实现该图片的加载;

若不满足,则从位图文件池中获取一符合该图片的解码数据所需的重用条件的 Bitmap 对象、或者创建一符合该图片的解码数据所需的重用条件的 Bitmap 对象,并将 Native 堆内存中缓存的、对该图片进行解码所得到的 Bitmap 对象的像素数组数据拷贝到获取到的或者创建的 Bitmap 对象中,得到新的 Bitmap

对象，并将新的 Bitmap 对象渲染到 ImageView 中，以实现该图片的加载；

其中，所述位图文件池位于 Java 堆内存中、且其引用方式为软引用。

进一步可选地，所述执行单元 32 还可用于若确定 ImageView 当前显示的 Bitmap 对象不满足该图片的解码数据所需的重用条件，则将 ImageView 当前显示的 Bitmap 对象存放至所述位图文件池中。

本领域技术人员应明白，本申请的实施例可提供为方法、装置（设备）、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器、CD-ROM、光学存储器等）上实施的计算机程序产品的形式。

本申请是参照根据本申请实施例的方法、装置（设备）和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

尽管已描述了本申请的优选实施例，但本领域内的技术人员一旦得知了基

本创造性概念，则可对这些实施例作出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本申请范围的所有变更和修改。

显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。