Android 系统内存管理研究及优化

杜吉志，徐明昆

（北京邮电大学网络技术研究院，北京 100876）

摘 要：随着基于 android 平台开发的应用越来越多，内存利用率问题成为应用开发者不可忽视的问题，如何让应用运行的更流畅，避免频繁的因内存不足而导致的系统崩溃问题的发生，成为应用开发者必须面对的问题。本论文试图解决这个问题。首先对 android 系统内部的内存管理机制进行分析，其次分析 android 内存管理机制的优缺点，最后提出内存管理优化的建议。

关键词：计算机软件；内存管理优化；综述；安卓；Java 虚拟机

中图分类号：TP316.4 **文献标识码**：A DOI：10.3969/j.issn.1003-6970.2012.12.022

Research and Optimization of Memory Management in Android System

DU Ji-zhi, XU Ming-kun

(Institute of Network Technology, Beijing University of Posts & Telecommunications, Beijing 100876,China)

【Abstract】 The application running on the android platform is becoming more and more. The critical problem is how to improve the memory usage percentage to avoid the system collapse. It's the question all developers must be faced with. This paper is trying to find the solution. In this paper, we talk about the memory management mechanism in android system. And then we analyze the advantage and disadvantage of the mechanism. At last, we propose some suggestion to improve the efficiency of memory management.

【Keywords】 Computer software; Optimization of memory management; Overview; Android; JVM

0 引言

android 操作系统是 Google 公司发行的基于 Linux 的开放源代码操作系统。当前针对 android 的应用都是基于 Android 的 Dalvik 虚拟机[1]开发的。Dalvik 虚拟机本身为应用程序提供默认的内存管理支持。但由于 Dalvik 虚拟机功能的局限性，以及系统本身针对每个 Dalvik 虚拟机最多只分配 24M 的堆空间。因此，如果工程师对 Android 的内存管理机制不了解，容易造成系统的 OOM[2](out of memory) 错误。

内存管理，是指程序运行时对计算机内存资源的分配和使用的技术。其最主要的目的是如何为进程高效快速的分配，并且在适当的时候释放和回收内存资源。一个执行中的程序在内存中主要包括代码区和数据区，以及操作系统为该程序分配的系统各种资源。 操作系统对内存管理的机制直接影响程序读取数据的速度，从而间接影响程序执行的速度。如果程序执行过程中出现 OOM 等内存错误，用户的个人配置数据会处于不确定状态。Android 操作系统多用于内存较小的手机客户端，并且提供对多个进程提供内存管理功能。

综上所述，了解 Android 内存管理对工程师是至关重要的。

1 Android 系统对内存管理分析

1.1 Android 对内存管理的整体架构综述

Android 采取了一种有别于 Linux 的进程管理策略[3]。Linux 的在进程活动停止后就结束该进程。而 Android 会把这些进程都保留在内存中，直到系统需要更多内存为止。这些保留在内存中的进程通常情况下不会影响整体系统的运行速度，并且当用户再次激活这些进程时，提升了进程的启动速度。

首先介绍 android 程序分类：

(1) foreground：正在屏幕上显示的进程和一些系统进程。

(2) visible：可见进程是一些不再前台，但用户依然可见的进程，例如 widget、输入法等，都属于 visible。

(3) secondary server：目前正在运行的一些服务（主要服务，如拨号等，是不可能被终止的）。

(4) hidden：启动后被切换到后台 的进程，如浏览器。后台进程的管理策略有多种：有较为积极的方式，一旦程序到达后台 立即终止，这种方式会提高程序的运行速度，但无法加速程序的再次启动；也有较消极的方式，尽可能多的保留后台程序，虽然可能会影响到单个程序的运行速度， 但在再次启动已启动的程序时，速度会有所提升。

(5) content provider：没有程序实体，提供内容供其它程序去用的，比如日历供应节点，邮件供应节点等。在终止进程时，这类程序应该有较高的优先权。

(6) empty：没有任何东西在内运行的进程，有些程序，比如 BTE，在程序退出后，依然会在进程中驻留一个空进程，这个进程里没有任何数据在运行，作用往往是提高该程序下次的启动速度或者记录程序的一些历史信息。这部分进程因该是最先终止的。

作者简介：杜吉志（1987-），男，硕士，主要研究方向：面向对象和组件的软件技术

通信联系人：徐明昆，高工，主要研究方向：面向对象和组件的软件技术 .

Android 系统决定结束进程的依据：

(1) 系统会对 进程的优先级进行评估，将优先级以“oom_adj”这个数值表示出来．一般来说，“oom_adj”的值越大，该进程被系统选中终止的可能就越高。

(2) 前台程序的“oom_adj”值为 0，这意味着它不会被系统终止，一旦它不可访问后，会获得个更高的“oom_adj”，一般“oom_adj”的值是根据软件在 LRU 列表中的位置所决定的。

(3) Android 有一套自己独特的进程管理模块，这个模块有更强的可定制性，可根据“oom_adj”值的范围来决定进程管理策略。

综上所述，Android 系统在选择退出程序时，并不是完全退出程序，该程序仍然会在后台驻留一个进程，以便下次更快的打开。系统会给每个类型的程序一个内存值阈[4]（阀门），当运行内存低于某个值时，系统会自动按照打开的先后顺序来关闭该 类型的程序。例如，当内存小于 24MB 时，系统才会自动关闭空进程这一类型的程序，释放出更多的内存来供新程序使用，已保证新开程序的正常运行。

1.2 Dalvik 虚拟机内存结构

从图 1 中可以看出，运行时区主要下面由 5 个部分组成：

Method Area：被装载的 class 的元信息存储在 Method Area 中，它是线程共享的。

Heap(堆)：一个虚拟机实例中只存在一个堆空间，存放一些对象信息，它是线程共享的。

Java 栈：虚拟机直接对 java 栈进行两种操作，以帧为单位的压栈和出栈（非线程共享）

程序计数器（非线程共享）。

本地方法栈（非线程共享）。

Dalvik 虚拟机把对象分为 Young、Tenured、Perm，对不同生命周期的对象使用不同的垃圾回收算法。

Young：分为三个区，一个 eden 区，两个 Survivor 区。程序中大部分新的对象都在 Eden 区中，当 Eden 区满时，还存活的对象将被复制到其中一个 Survivor 区，当此 Survivor 区的对象占用空间满了时，此区存活的对象又被复制到另外一个 Survivor 区，当这个 Survivor 区也满了的时候，从第一个 Survivor 区复制过来的并且此时还存活的对象，将被复制到 Tenured 中。

Tenured：存放的是 Young 复制过来的对象，也就是在 Young 中还存活的对象，并且区满了复制过来的。Tenured 中的对象生命周期都比较长。

Perm：用于存放静态的类和方法，对垃圾回收没有显著的影响。(如图 1)

2 Android 内存管理的缺点

1) Android 针对每个应用程序启动一个独立的虚拟机，因此造成较大的内存消耗。

2) Android 对已经关闭的进程并不立即回收内存，容易造成内存不足。

3 Android 内存管理的优点

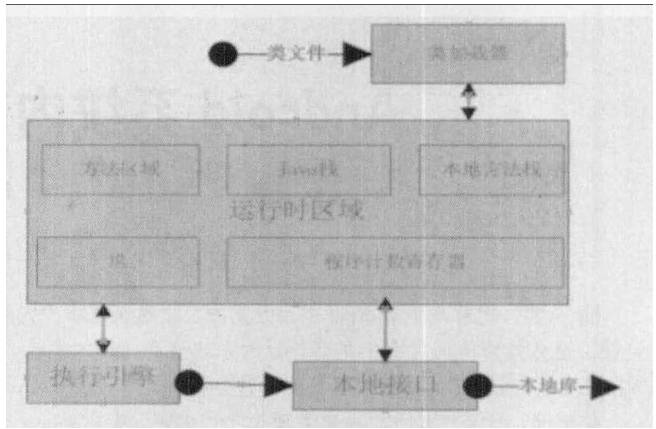

图 1　Dalvik 虚拟机内存结构

Fig.1　Memory Structure of Dalvik Virtual Machine

1) Android 的应用在被切换到后台时，它其实已经被暂停了，并不会消耗 CPU 资源，只保留了运行状态， 相对于 Linux 操作系统对内存的消耗更小。

2) 允许设定内存调度的阀值，只有低于这个值系统才会按一个列表来关闭用户不需要的东西，增加系统灵活性。

3) 为每个应用打开一个独立的虚拟机，避免虚拟机崩溃导致整个系统崩溃。

4 针对 Android 应用程序的优化策略

1) 应该尽量避免 static 成员变量引用资源耗费过多的实例，比如 Context。

2) Context 尽量使用 Application Context，因为 Application 的 Context 生命周期比较长，引用它不会出现内存泄露的问题。

3) 用弱引用代替强引用。

4) 将线程的内部类，改为静态内部类。

5) 在线程内部采用弱引用保存 Context 引用。

6) 对于比较耗内存的 Bitmap 对象，需要及时销毁。

参考文献：

[1] Jin T Y. Research of Android Architecture [M]. Beijing: Posts & Telecom Press, 2012

[2] Deng F P.Research on Android[M]. Beijing: China Machine Press

[3] Li N.Director of Android Developer[M]. Beijing: Posts & Telecom Press

[4] Yu Z L. Google Android SDK Application [M]. Beijing: Posts & Telecom Press, 2012

[5] Li M H. Introduction to Memory Management[OL].[2012]. http://android.mybdqn.com/kt/5414/

[6] Liu B L. Memory Leak in Android[OL]. [2012].http://www.apkbus.com/android-71944-1-1.html

[7] Marko Gargenta. Learning Android[M]. Publishing House of Electronics Industry, 2012

[8] Margaret Butler. Android: Changing Mobile Landscape[J]. IEEE Pervasive Computing, 2011, 10(1):4-7