Devik和Art虚拟机

**一、什么是Dalvik虚拟机**

Dalvik是Google公司自己设计用于Android平台的Java虚拟机，它是Android平台的重要组成部分，支持dex格式（Dalvik Executable）的Java应用程序的运行。dex格式是专门为Dalvik设计的一种压缩格式，适合内存和处理器速度有限的系统。Google对其进行了特定的优化，使得Dalvik具有高效、简洁、节省资源的特点。从Android系统架构图知，Dalvik虚拟机运行在Android的运行时库层。

Dalvik作为面向Linux、为嵌入式操作系统设计的虚拟机，主要负责完成对象生命周期管理、堆栈管理、线程管理、安全和异常管理，以及垃圾回收等。另外，Dalvik早期并没有JIT编译器，直到Android2.2才加入了对JIT的技术支持。

**二、Dalvik虚拟机的特点**

体积小，占用内存空间小；

专有的DEX可执行文件格式，体积更小，执行速度更快；

常量池采用32位索引值，寻址类方法名，字段名，常量更快；

基于寄存器架构，并拥有一套完整的指令系统；

提供了对象生命周期管理，堆栈管理，线程管理，安全和异常管理以及垃圾回收等重要功能；

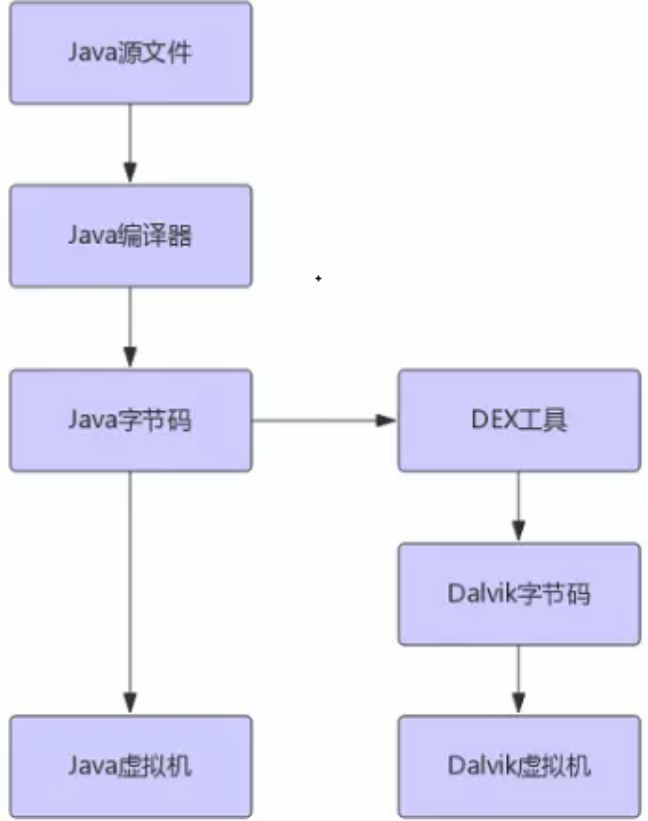
所有的Android程序都运行在Android系统进程里，每个进程对应着一个Dalvik虚拟机实例。

**三、Dalvik虚拟机和Java虚拟机的区别**

Dalvik虚拟机与传统的Java虚拟机有着许多不同点，两者并不兼容，它们显著的不同点主要表现在以下几个方面：

**Java虚拟机运行的是Java字节码，Dalvik虚拟机运行的是Dalvik字节码。**

传统的Java程序经过编译，生成Java字节码保存在class文件中，Java虚拟机通过解码class文件中的内容来运行程序。而Dalvik虚拟机运行的是Dalvik字节码，所有的Dalvik字节码由Java字节码转换而来，并被打包到一个DEX（Dalvik Executable）可执行文件中。Dalvik虚拟机通过解释DEX文件来执行这些字节码。



**Dalvik可执行文件体积小。Android SDK中有一个叫dx的工具负责将Java字节码转换为Dalvik字节码。**

dx工具对Java类文件重新排列，消除在类文件中出现的所有冗余信息，避免虚拟机在初始化时出现反复的文件加载与解析过程。一般情况下，Java类文件中包含多个不同的方法签名，如果其他的类文件引用该类文件中的方法，方法签名也会被复制到其类文件中，也就是说，多个不同的类会同时包含相同的方法签名，同样地，大量的字符串常量在多个类文件中也被重复使用。这些冗余信息会直接增加文件的体积，同时也会严重影响虚拟机解析文件的效率。**消除其中的冗余信息，重新组合形成一个常量池，所有的类文件共享同一个常量池。由于dx工具对常量池的压缩，使得相同的字符串，常量在DEX文件中只出现一次，从而减小了文件的体积。**

dex格式文件使用共享的、特定类型的常量池机制来节省内存。常量池存储类中的所有字面常量，它包括字符串常量、字段常量等值。  
简单来讲，dex格式文件就是将多个class文件中公有的部分统一存放，去除冗余信息。

**Java虚拟机与Dalvik虚拟机架构不同。**这也是Dalvik与JVM之间最大的区别。

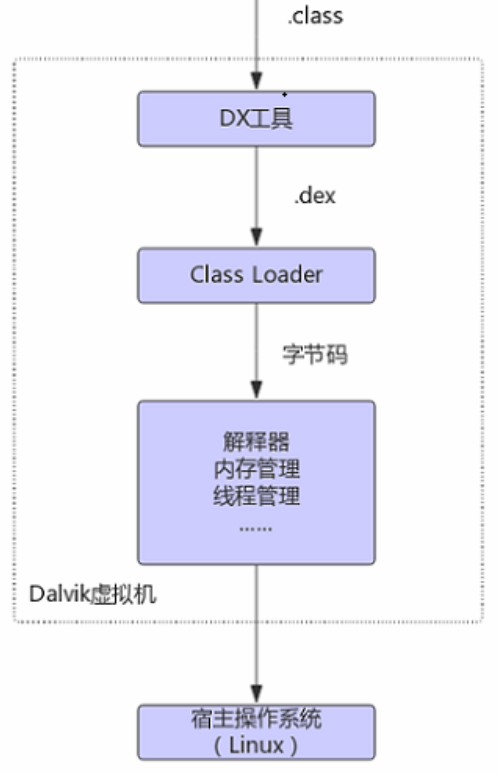
**Java虚拟机基于栈架构**，程序在运行时虚拟机需要频繁的从栈上读取或写入数据，这个过程需要更多的指令分派与内存访问次数，会耗费不少CPU时间，对于像手机设备资源有限的设备来说，这是相当大的一笔开销。**Dalvik虚拟机基于寄存器架构**。数据的访问通过寄存器间直接传递，这样的访问方式比基于栈方式要快很多。

总结:java虚拟机和Davik虚拟机的区别：

1. 去除冗余信息，dex格式文件就是将多个class文件中公有的部分统一存放（字符串常亮），从而减少dex文件大小
2. **Java虚拟机基于栈架构，Dalvik虚拟机基于寄存器架构**。数据的访问通过寄存器间直接传递，这样的访问方式比基于栈方式要快很多。

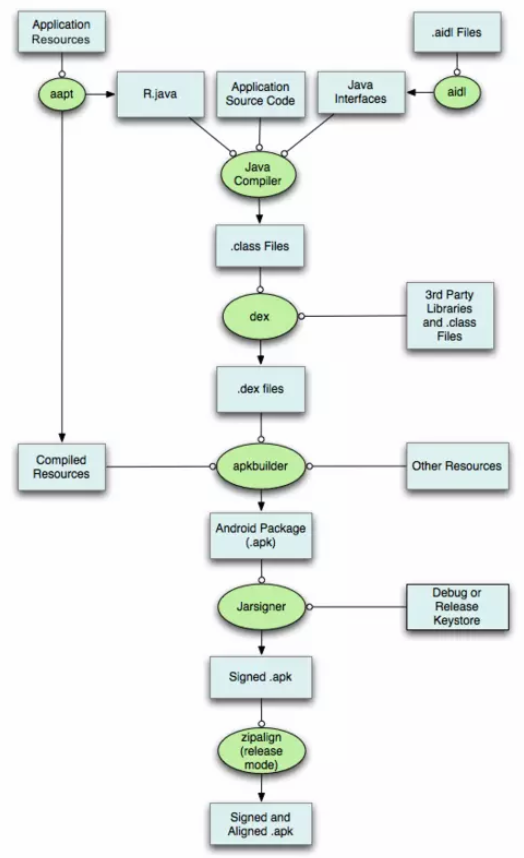
寄存器：是[中央处理器](https://baike.baidu.com/item/%E4%B8%AD%E5%A4%AE%E5%A4%84%E7%90%86%E5%99%A8/284033)内的组成部分。寄存器是有限存贮容量的高速存贮部件，它们可用来暂存[指令](https://baike.baidu.com/item/%E6%8C%87%E4%BB%A4/3225201)、[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/33305)和[地址](https://baike.baidu.com/item/%E5%9C%B0%E5%9D%80/80420)。在[中央处理器](https://baike.baidu.com/item/%E4%B8%AD%E5%A4%AE%E5%A4%84%E7%90%86%E5%99%A8/284033)的控制部件中，包含的寄存器有[指令寄存器](https://baike.baidu.com/item/%E6%8C%87%E4%BB%A4%E5%AF%84%E5%AD%98%E5%99%A8/3219483)(IR)和[程序计数器](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%A1%E6%95%B0%E5%99%A8/3219536" \t "_blank)(PC)。在[中央处理器](https://baike.baidu.com/item/%E4%B8%AD%E5%A4%AE%E5%A4%84%E7%90%86%E5%99%A8/284033" \t "_blank)的算术及逻辑部件中，寄存器有[累加器](https://baike.baidu.com/item/%E7%B4%AF%E5%8A%A0%E5%99%A8/8590163)(ACC)。

**四、Dalvik虚拟机的结构**



一个应用首先经过DX工具将class文件转换成Dalvik虚拟机可以执行的dex文件，然后由类加载器加载原生类和Java类，接着由解释器根据指令集对Dalvik字节码进行解释、执行。最后，根据dvm\_arch参数选择编译的目标机体系结构。

**五、Android APK 编译打包流程**



1.Java编译器对工程本身的java代码进行编译，这些java代码有三个来源：app的源代码，由资源文件生成的R文件(aapt工具)，以及有aidl文件生成的java接口文件(aidl工具)。产出为.class文件。

①.用AAPT编译R.java文件  
②编译AIDL的java文件  
③把java文件编译成class文件

2..class文件和依赖的三方库文件通过dex工具生成Delvik虚拟机可执行的.dex文件，包含了所有的class信息，包括项目自身的class和依赖的class。产出为.dex文件。

3.apkbuilder工具将.dex文件和编译后的资源文件生成未经签名对齐的apk文件。这里编译后的资源文件包括两部分，一是由aapt编译产生的编译后的资源文件，二是依赖的三方库里的资源文件。产出为未经签名的.apk文件。

4.分别由Jarsigner和zipalign对apk文件进行签名和对齐，生成最终的apk文件。

总结为：编译-->DEX-->打包-->签名和对齐

总结:Apk打包过程

1. 编译：app源代码，资源文件生成的R文件，以及aidl生成的java接口文件 经过aapt的编译生成.class文件；Class文件以及依赖的第三方库文件 通过 dex工具 生成dex文件
2. 打包：dex文件+ （aapt）编译后的资源文件 通过 apkbuilder 生成 未签名对齐的apk文件
3. 签名对齐：对apk进行签名对其

**六、ART虚拟机与Dalvik虚拟机的区别**

**什么是ART:**

ART代表Android Runtime，其处理应用程序执行的方式完全不同于Dalvik，Dalvik是依靠一个Just-In-Time (JIT)编译器去解释字节码。开发者编译后的应用代码需要通过一个解释器在用户的设备上运行，这一机制并不高效，但让应用能更容易在不同硬件和架构上运 行。ART则完全改变了这套做法，在应用安装时就预编译字节码到机器语言，这一机制叫Ahead-Of-Time (AOT）编译。在移除解释代码这一过程后，应用程序执行将更有效率，启动更快。

**ART优点：**

1、系统性能的显著提升。  
2、应用启动更快、运行更快、体验更流畅、触感反馈更及时。  
3、更长的电池续航能力。  
4、支持更低的硬件。

**ART缺点：**

1、更大的存储空间占用，可能会增加10%-20%。  
2、更长的应用安装时间。

**ART虚拟机相对于Dalvik虚拟机的提升**

参考：[art和dalvik的区别？](https://link.jianshu.com?t=https:/www.zhihu.com/question/29406156)

\*\*预编译 \*\*

在dalvik中,如同其他大多数JVM一样,都采用的是JIT来做及时翻译(动态翻译),将dex或odex中并排的dalvik code(或者叫smali指令集)**运行态**翻译成native code去执行.JIT的引入使得dalvik提升了3~6倍的性能。

而在ART中,完全抛弃了dalvik的JIT,使用了AOT直接在安装时将其完全翻译成native code.这一技术的引入,使得虚拟机执行指令的速度又一重大提升 ，在开机以及使用时更加流畅。

**①垃圾回收机制**

首先介绍下dalvik的GC的过程.主要有有四个过程:  
1、当gc被触发时候,其会去查找所有活动的对象,这个时候整个程序与虚拟机内部的所有线程就会挂起,这样目的是在较少的堆栈里找到所引用的对象.需要注意的是这个回收动作和应用程序**非并发**。

2、gc对符合条件的对象进行标记

3、gc对标记的对象进行回收

4、恢复所有线程的执行现场继续运行

**dalvik这么做的好处是,当pause了之后,GC势必是相当快速的.但是如果出现GC频繁并且内存吃紧势必会导致UI卡顿,掉帧.操作不流畅等。**

后来ART改善了这种GC方式 , **主要的改善点在将其非并发过程改变成了部分并发.还有就是对内存的重新分配管理**。

当ART GC发生时:

1、GC将会锁住Java堆,扫描并进行标记

2、标记完毕释放掉Java堆的锁,并且挂起所有线程

3、GC对标记的对象进行回收

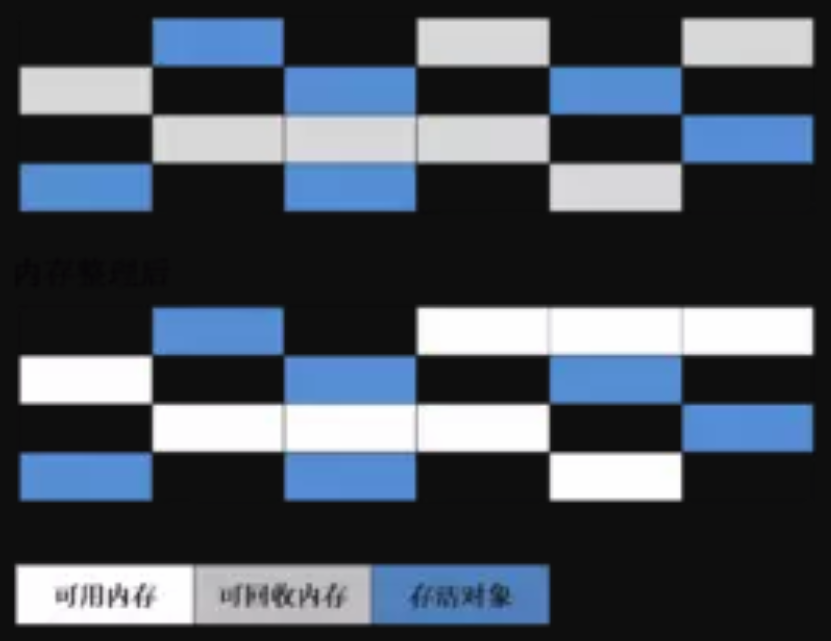
4、恢复所有线程的执行现场继续运行

5、重复2-4直到结束

可以看出整个过程做到了部分并发使得时间缩短.据官方测试数据说gc效率提高2倍

**提高内存使用，减少碎片化**

**Dalvik内存管理特点是:内存碎片化严重**，当然这也是Mark and Sweep算法带来的弊端



可以看出每次gc后内存千疮百孔，本来连续分配的内存块变得碎片化严重，之后再分配进入的对象再进行内存寻址变得困难。

**ART的解决:**在ART中,它将Java分了一块空间命名为**Large-Object-Space**,这块内存空间的引入用来专门存放large object。同时ART又引入了moving collector的技术,即将不连续的物理内存块进行对齐.对齐了后内存碎片化就得到了很好的解决.Large-Object-Space的引入一是因为moving collector对大块内存的位移时间成本太高,而且提高内存的利用率  
根官方统计，ART的内存利用率提高10倍了左右。

总结：

Java和Devik虚拟机

1.去除冗余信息，dex格式文件就是将多个class文件中公有的部分统一存放（字符串常亮），从而减少dex文件大小

**2.Java虚拟机基于栈架构，Dalvik虚拟机基于寄存器架构**。数据的访问通过寄存器间直接传递，这样的访问方式比基于栈方式要快很多。

ART替换Delvik虚拟机的好处：

1. 执行更快

Delvik虚拟机使用的是JIT(just-in-time)编译器，运行的使用，将smail翻译成native code再执行；而ART虚拟机的Android设备，在apk安装的时候，预编译字节码文件为native code，(这个过程叫做AOT{Ahead-of-Time})，没有了解释这一项，因此启动app更快了，确定，内存暂用率高了

1. GC部分并发，减少了卡顿，掉帧，操作不流畅

GC机制的回收(1)当前应用程序所有线程挂起，(2)查找所有活动的对象 (3)对符合回收的对象标记 (4)回收标记的对象 (5)挂起的线程恢复

Devik：(1)(2)(3)(4)（5）全部非并发

ART: (2)(3)(1)(4)(5)

1. 减少内存碎片化，提高内存利用率

Gc处理后，很多内存不在连续（内存碎片化），分配以及寻址很困难；

解决：(1) 它将Java分了一块空间命名为**Large-Object-Space**,这块内存空间的引入用来专门存放large object;(2) 使用moving collector的技术,即将不连续的物理内存块进行对齐.对齐了后内存碎片化就得到了很好的解决

补充内容：

Smail: apk文件通过apktool反编译出来的都有一个smali文件夹，里面都是以.smali结尾的文件。  
smali语言是Davlik的寄存器语言，语法上和汇编语言相似，Dalvik VM与JVM的最大的区别之一就是Dalvik VM是基于寄存器的。基于寄存器的意思是，在smali里的所有操作都必须经过寄存器来进行。

1. 机器语言  
   所谓机器语言，就是只用0和1两个二进制数书写的编程语言。它有的时候也叫作原生代码（Native Code）。这里需要注意，只要CPU种类不同，对机器语言的解释也就不同。
2. 编译器  
   由于计算机只能读懂机器指令，所以就需要一个能够将汇编指令转换成机器指令的翻译程序，这样的程序我们称其为编译器。程序员用汇编语言写出源程序，再用汇编编译器将其翻译为机器码。就像我们写的Java，OC，等等，都是由编译器最终转换成01代码，最后由计算机执行。
3. 时钟信号的频率  
   由时钟发生器发送给CPU的电信号的频率。单位是MHZ(兆赫兹 = 100万回/秒)。时钟信号是0和1两个数之间反复变换的电信号，类似于滴答滴答左右摆动的钟摆一样。通常我们把发出一次滴答的时间称作一个时钟周期。

**参考文章：**

[理解Android虚拟机体系结构](https://link.jianshu.com?t=http:/www.cnblogs.com/lao-liang/p/5111399.html)  
[深入理解Android（二）：Java虚拟机Dalvik](https://link.jianshu.com?t=http:/www.infoq.com/cn/articles/android-in-depth-dalvik)  
[Android编译流程和Gradle使用](https://www.jianshu.com/p/eaaddfe34d11)  
[art和dalvik的区别？](https://link.jianshu.com?t=https:/www.zhihu.com/question/29406156)  
[深入理解Android工程的编译过程](https://link.jianshu.com?t=https:/www.oschina.net/question/54100_33877)