

硕 士 研 究 生 读 书 报 告



题目 Android热修复技术分析

作者姓名 刘康

作者学号 21651060

指导教师 李启雷

学科专业 移动互联网与游戏开发

所在学院 软件学院

提交日期 二○一七年一月

The Analysis of Android Hotfix technology

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Li Qilei

By

Liu Kang

Zhejiang University, P.R. China

2017

摘要

本文重点探讨了在Android应用发布后所使用的热修复技术，区别于普通的开发流程，在发现Bug后需要重新发布新版本的应用程序，Android热修复技术可以使开发的流程更加灵活。能够在用户无感知的情况下，修复bug,增加应用程序的用户体验。

文中主要分析了QQ空间补丁方案、微信Tinker和阿里百川HotFix技术，对比了它们之间的优缺点。

**关键词**：热修复、HotFix、Tinker

Abstract

This article focuses on the use of Android after the release of the hot repair technology. Different from common development process, after the discovery of Bug ,we need to release a new version of the application, Android hotfix technology can make the development process more flexible. In the absence of user awareness, we could repair bug and increase the user’s experience of the application.

This paper mainly analyzes the QQ space patching solutions, WeChat Tinker and Ali Baichuan HotFix technology, compares the advantages and disadvantages between them.

**Keywords：**patch、HotFix、Tinker

1引言

当一个App发布之后，突然发现了一个严重bug需要进行紧急修复，这时候公司各方就会忙得焦头烂额：重新打包App、测试、向各个应用市场和渠道换包、提示用户升级、用户下载、覆盖安装。有时候仅仅是为了修改了一行代码，也要付出巨大的成本进行换包和重新发布。不仅大大增加开发成本也会影响到产品的口碑，造成用户流失。这时候就提出一个问题：有没有办法以补丁的方式动态修复紧急Bug，不再需要重新发布App，不再需要用户重新下载，覆盖安装？于是涌现出来很多热修复方案。能够让应用能够在无需重新安装的情况实现更新，帮助应用快速建立动态修复能力[1]。

**2 技术背景**

****

图2.1 正常开发流程

在一般的开发流程上，修复Bug需要重新打包发布新版本的App，这样做有许多缺点：

1. 重新发布新版本的App代价太高，耗时也非常的长。
2. 用户需要下载新的完整版的App。
3. 从Bug的发现到处理可能需要很长的一段时间，若是Bug修复的不及时，影响用户的体验。

****

图2.2 热修复开发流程

若是采用热修复技术开发的App，这可以将Bug的修复变得更为灵活，相比较于正常开发就显得优势很大：

1. 在修复Bug后无需打包发布一个新版本的APP，更加轻量级。
2. 在Bug的修复过程中，用户无感知，提供了很好的用户体验。
3. 修复更加快捷，节省了人力和时间的成本。

**3 热修复技术**

热修复是这几年中比较新兴的技术，在技术上比较完善的有阿里巴巴的AndFix、Dexposed，腾讯QQ空间的超级补丁技术和微信Tinker技术。

**3.1 QQ空间超级补丁技术**

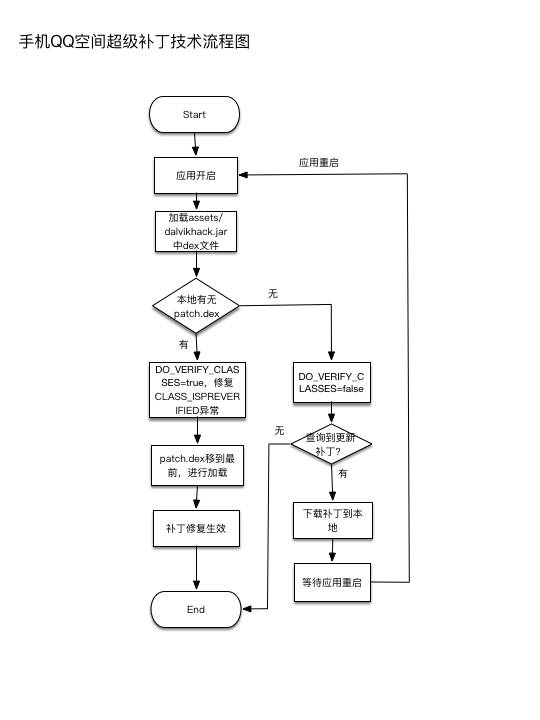


图3.1 手机QQ空间超级补丁技术流程图

超级补丁技术基于DEX分包方案，使用了多DEX加载的原理，大致的过程就是：把BUG方法修复以后，放到一个单独的DEX里，插入到dexElements数组的最前面，让虚拟机去加载修复完后的方法。

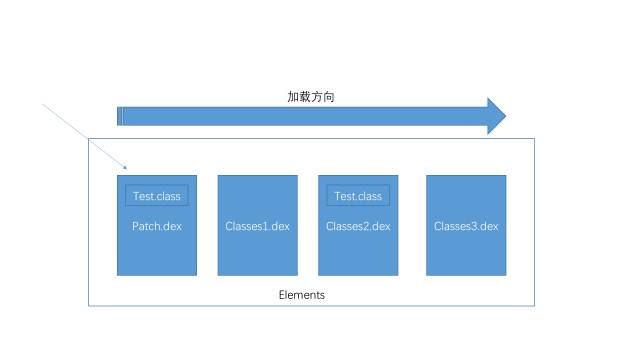


图3.2 patch.dex加载流程

当patch.dex中包含Test.class时就会优先加载，在后续的DEX中遇到Test.class就会直接返回而不去加载，这样就可以成功修复。但是有一个问题是，当两个调用关系的类不在同一个DEX时，就会产生异常报错。我们知道，在APK安装时，虚拟机需要将classes.dex优化成odex文件，然后才会执行。在这个过程中，会进行类的verify操作，如果调用关系的类都在同一个DEX中的话就会被打上CLASS\_ISPREVERIFIED的标志，然后才会写入odex文件。

所以，为了可以正常的进行打补丁修复，必须避免类被打上CLASS\_ISPREVERIFIED标志，具体的做法就是单独放一个类在另外DEX中，让其他类调用。

通过反编译QQ空间APK来查看具体实现：

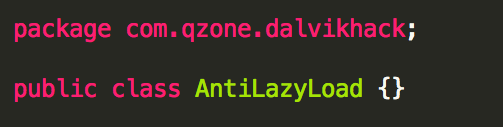
先进入程序入口QZoneRealApplication，在attachBaseContext中进行了两步操作：修复CLASS\_ISPREVERIFIED标志导致的unexpected DEX problem异常、加载修复的DEX。



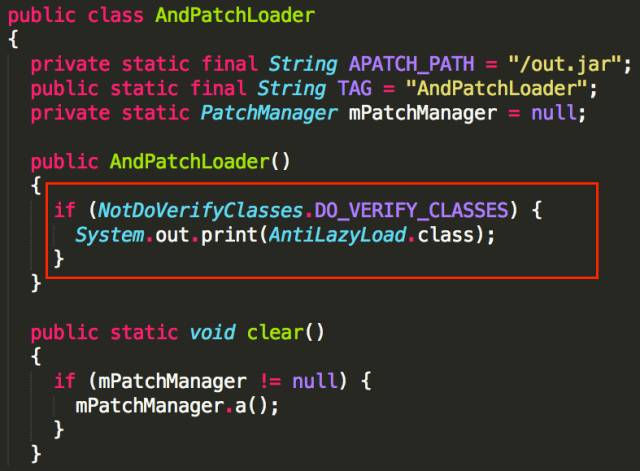
1. 修复unexpectedDEX problem异常



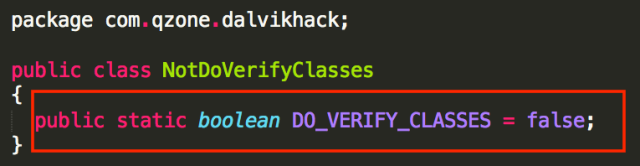
加载一个libs目录下的dalvikhack.jar。在项目的assets/lisbs下，解压得到classes.dex文件，逆向打开该dex文件。



通过不同的DEX加载进来，然后在每一个类的构造方法中引用其他DEX中的唯一类AnitLazyLoad，避免类被打上CLASS\_ISPREVERIFIED标志。

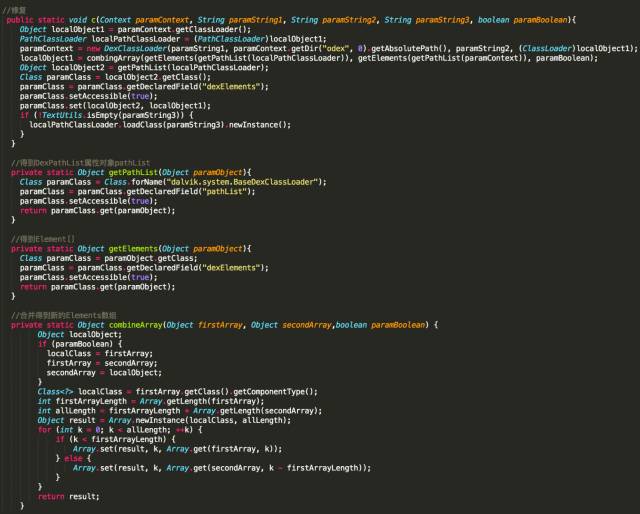


在无修复的情况下，将DO\_VERIFY\_CLASSES设置为false，提高性能。只有在需要修复的时候，才设置为true。



1. 加载修复的dex

从loadPatchDex()方法进入，经过几次跳转，到达核心的代码段，SystemClassLoaderInjector.c()。



1. 通过反射获取到DexPathList属性对象pathList;
2. 通过反射调用pathList的dexElements方法把patch.dex转化为Element[]
3. 将Element[]进行合并，并把patch.dex放在最前面，并加载Element[]，已达到修复的目的[2]。

**3.2 微信Tinker**

开源地址（https://github.com/Tencent/tinker）

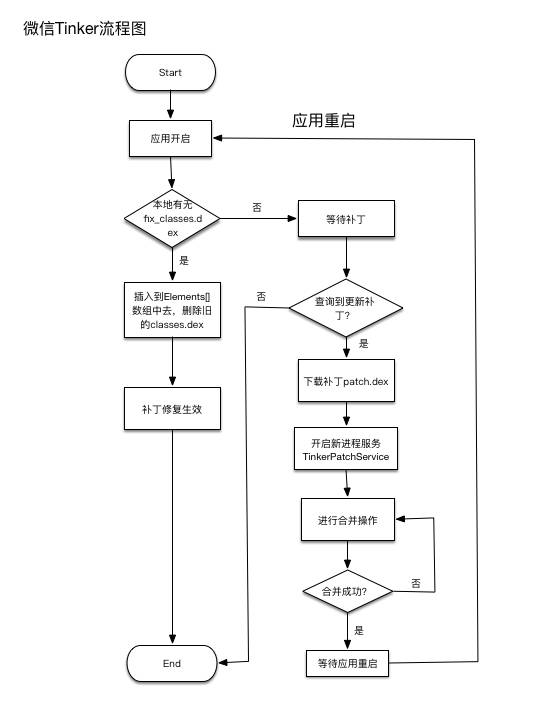


图3.3微信Tinker流程图

微信针对QQ空间超级补丁技术的不足提出了一个提供DEX差量包，整体替换DEX的方案。主要的原理是与QQ空间超级补丁技术基本相同，区别在于不再将patch.dex增加到elements数组中，而是差量的方式给出patch.dex，然后将patch.dex与应用的classes.dex合并，然后整体替换掉旧的DEX，达到修复的目的[3]。

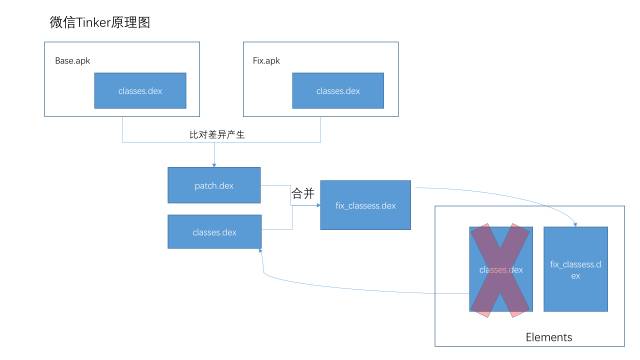
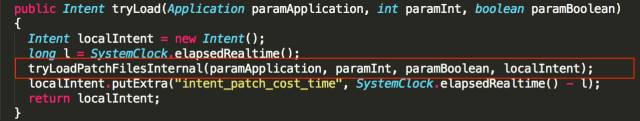


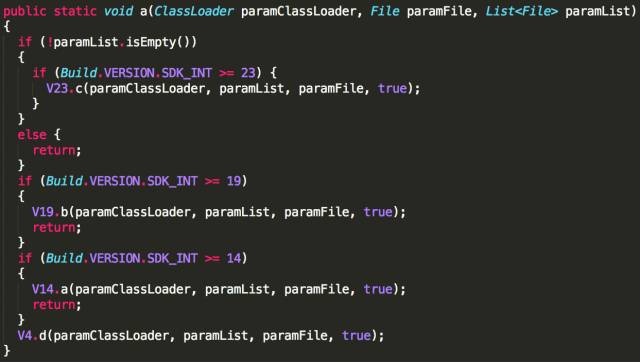
图3.4微信Tinker原理图

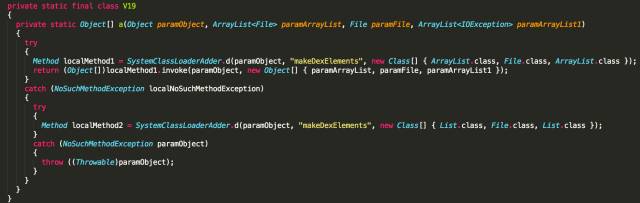
在onBaseContextAttached()调用了loadTinker()



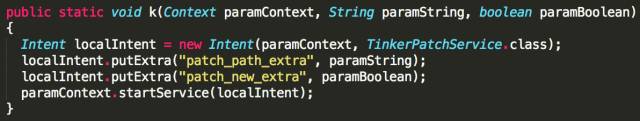


在tryLoadPatchFileInternal()中尝试加载本地的补丁，在经过跳转进入核心修复功能类SystemClassLoaderAdder.class中。





通过反射操作得到PathClassLoader的DexPatchList，反射调用patchlist的makeDexElements()方法把本地的Dex文件直接替换到Element[]数组中去，达到修复的目的。



开启新的进程服务TinkerPatchService对patch.dex和class.dex进行合并操作。

综上所述，微信Tinker热修复方法的优点在于：合成整包，不用在构造函数插入代码；性能提高，兼容性和稳定性比较高；开发者透明，不需要对包进行额外的处理。不足之处在于：和超级补丁技术一样，不支持即时生效，必须通过重启应用的方式才能生效；需要给应用开启新的进程才能进行合并，并且很容易因为内存消耗等原因合并失败；合并时占用额外磁盘空间，对于多DEX的应用来说，如果修改了多个DEX文件，就需要下发多个patch.dex与对应的classes.dex进行合并操作时这种情况会更严重，因此合并过程的失败率也会更高。

**3.3阿里百川HotFix**

阿里百川推出的热修复HotFix技术，相对于QQ空间超级补丁技术和微信Tinker来说，最大的优势在于能够在最短的时间内修复Bug，下拉补丁立即生效无需等待。



图3.5 HotFix关系图

* + 1. AndFix实现原理

AndFix不同于QQ空间超级补丁技术和微信Tinker通过增加或替换整个DEX的方案，提供了一种运行时在Native修改Filed指针的方式，实现方法的替换，达到即时生效无需重启，对应用无性能消耗的目的。

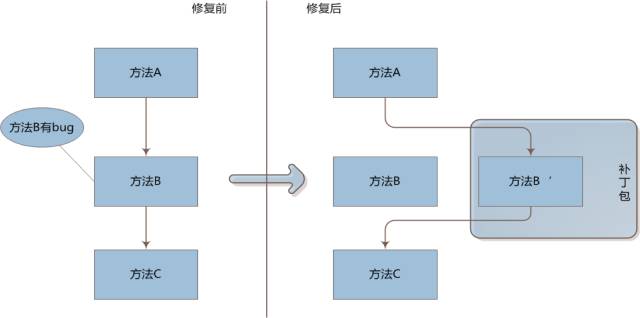


图3.6 AndFix原理图

* + 1. AndFix实现过程

对于实现方法的替换，需要在Native层操作，经过三个步骤：



图3.7 AndFix实现步骤

在Dalvik设备中的具体实现过程：

（1）setup()



对于Dalvik来说，遵循JIT即时编译机制，需要在运行时装载libdvm.so动态库，获取以下内部函数：

1）dvmThreadSelf( )：查询当前的线程；

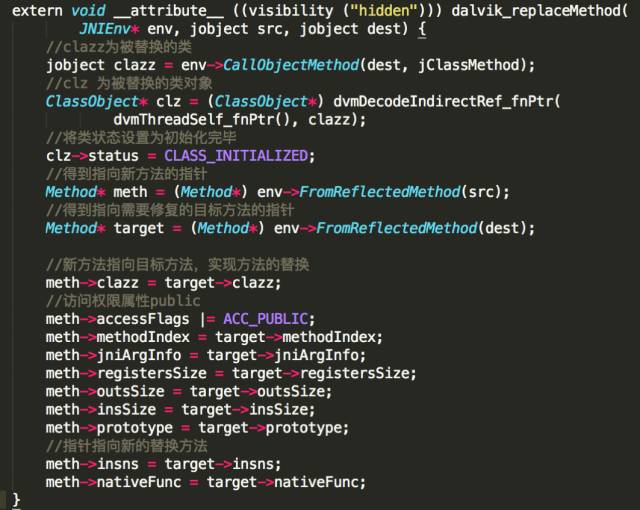
2）dvmDecodeIndirectRef()：根据当前线程获得ClassObject对象。

（2）setFieldFlag



该操作的目的：让private、protected的方法和字段可被动态库看见并识别。原因在于动态库会忽略非public属性的字段和方法。

（3）replaceMethod



该步骤是方法替换的核心，替换的流程如下：



图3.8替换流程图

AndFix对ART设备同样支持，具体的过程与Dalvik相似，这里不再赘述。

HotFix(AndFix)的优势在于：bug修复的即时性；补丁包同样采用差量技术，生成的patch体积小；对应用无侵入，几乎无性能损耗。不足之处在于不支持新增的字段，以及修改<init>方法，也不支持对资源的替换；对于厂商的自定义ROM，对少数的机型不支持。

**3.4 Demo分析**

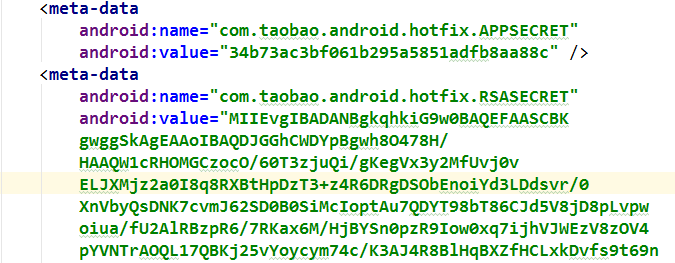
Demo采用阿里百川的HotFix进行热修复，阿里百川提供接口，让开发人员可以在控制台上直接发布补丁。

首先需要注册成为阿里百川的开发人员，开通热修复功能。

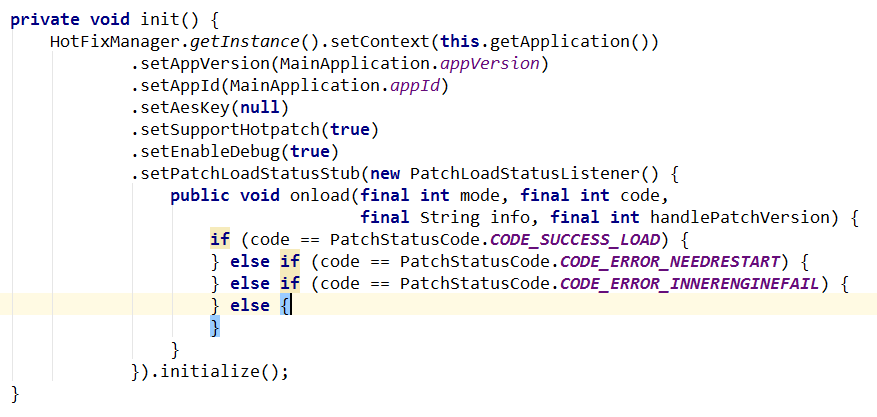


图3.9 开发者控制台

在应用中需要加入阿里百川的sdk，并修改Manifest文件



以及加入initialize方法并保证在应用的一开始就能运行



导出原始的有bug的apk文件，将bug修复，再导出修复后的apk文件，使用patch工具对比两个apk文件的不同，生成补丁包。

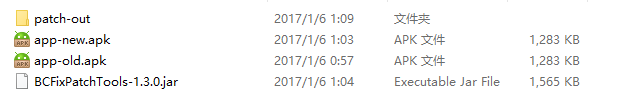


图3.10 补丁包文件夹

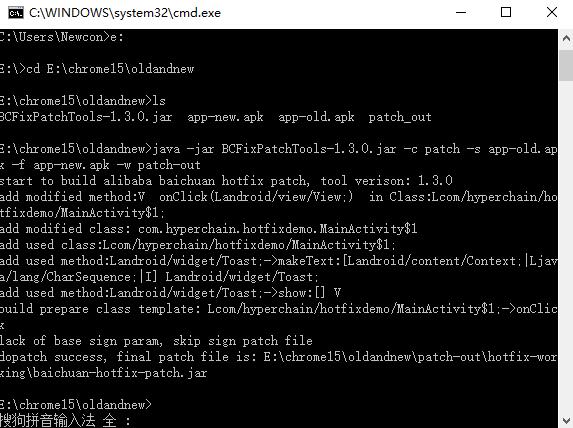


图3.11 补丁生成命令行

最后将生成的补丁包发布到阿里百川的开发者平台上就可以完成热修复。



图3.11 上传补丁

热修复完成



**4 小结**

对比上述三个热修复的方案，QQ空间超级补丁技术和微信Tinker修复原理都是基于类加载，而HotFix则是通过修改Filed指针的方式，实现方法的替换。

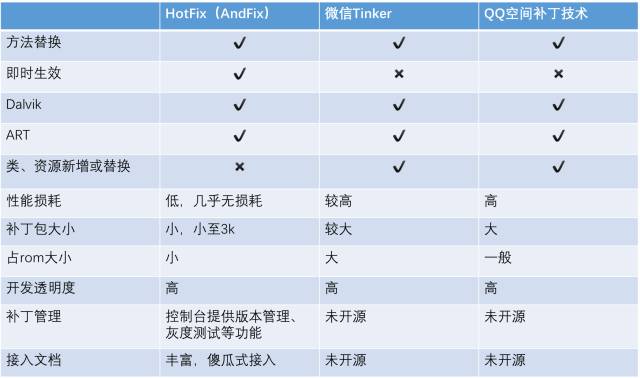


图4.1 综合分析图

在实际的运行中，我们发现超级补丁技术和Tinker热修复技术可能会对应用的性能造成影响。正是尤其多DEX加载导致了启动时间过长，很容易就会引发应用的ANR。我们知道当应用在主线程等待超过5s以后，就会直接导致长时间无响应而退出。超级补丁技术为保证ART不出现地址错乱问题，需要将所有关联的类全部加入到补丁中，而微信Tinker采取一种差量包合并加载的方式，都会使要加载的DEX体积变得很大。这也很大程度上容易导致ANR情况的出现。

针对应用的性能损耗，我们可以举例做一个对比。

某APP的启动载入时间为3s左右，本身就是基于多DEX模式的实现。

分别接入三种热修复服务，根据腾讯提供超级补丁技术和Tinker的数据，那么会变成以下的场景：

阿里百川HotFix:启动时间几乎无增加，不增加运行期额外的磁盘消耗。

QQ空间超级补丁技术：如果应用有700个类，启动耗时增加超过2.5s，达到5.5s以上。

微信Tinker：假设应用有5个DEX文件，分别修改了这5个DEX，产生5个patch.dex文件，就要进行5次的patch合并动作，假设每个补丁1M，那么就要多占用7.5M的磁盘空间。

显然对于修复紧急BUG这个场景，阿里百川HotFix的更为合适，它更加轻量，可以在不重启的情况下生效，且对性能几乎没有影响。微信Tinker、QQ空间超级补丁技术更多地把场景定位在发布小的新功能上，采用ClassLoader的模式，牺牲较高的性能代价去实现类、资源新增或替换的功能。阿里百川HotFix对应用本身做到无侵入，无性能损耗。

参考文献

[1]Lim Y K,Paramail S,Kim C G,et al.A Selective Ahead-Of-Time Complier on Android Device[C].In :Information Science and Applications(ICISA)，2012 International Conference on .IEEE，2012:1-6

[2]冯克环．基于Dalvik虛拟机自适应编译系统的分析与化化[D]．东南大学集成电路学院,2012

[3]李霞. Android虚拟机运行时技术的分析与测评[D]. 东南大学，2015.