手淘容器化框架Atlas实践

玄黎
淘宝-移动平台

倪生华(玄黎)，
淘宝移动平台技术部的资深专家

2012年加入手机淘宝技术部门，主要负责手机淘宝技术团队的快速交付、研发支撑等体系的建设，参与开发了手淘容器化的动态化体系，构建支撑以及相关支撑体系等的工作，从无到有实现了手淘的插件化构建，集成交付，监控运维等体系，目前支撑了阿里集团大部分的无线业务。

扫码观看大会视频

目 录
content

1. 技术背景&现状
2. 组件化实现
3. 动态性实现
4. 周边优化点

扫码观看大会视频

一、技术背景&现状

诉求点:

1. 并行开发
2. 快速迭代
3. 解耦&独立

扫码观看大会视频

应用现状

01 几十个团队几百人App的协作

高效团队协作

02 从月->周->天的发布周期变更

快速迭代能力

03 淘系大小型应用普遍采用

内部广泛应用

04 整体0.05%以下的Crash率

运行高稳定性

扫码观看大会视频

容器化支持

手淘的Atlas框架主要提供了组件化、动态性、解耦化的支持。支持工程师在工程编码期、Apk运行期以及后续运维修复期的问题。

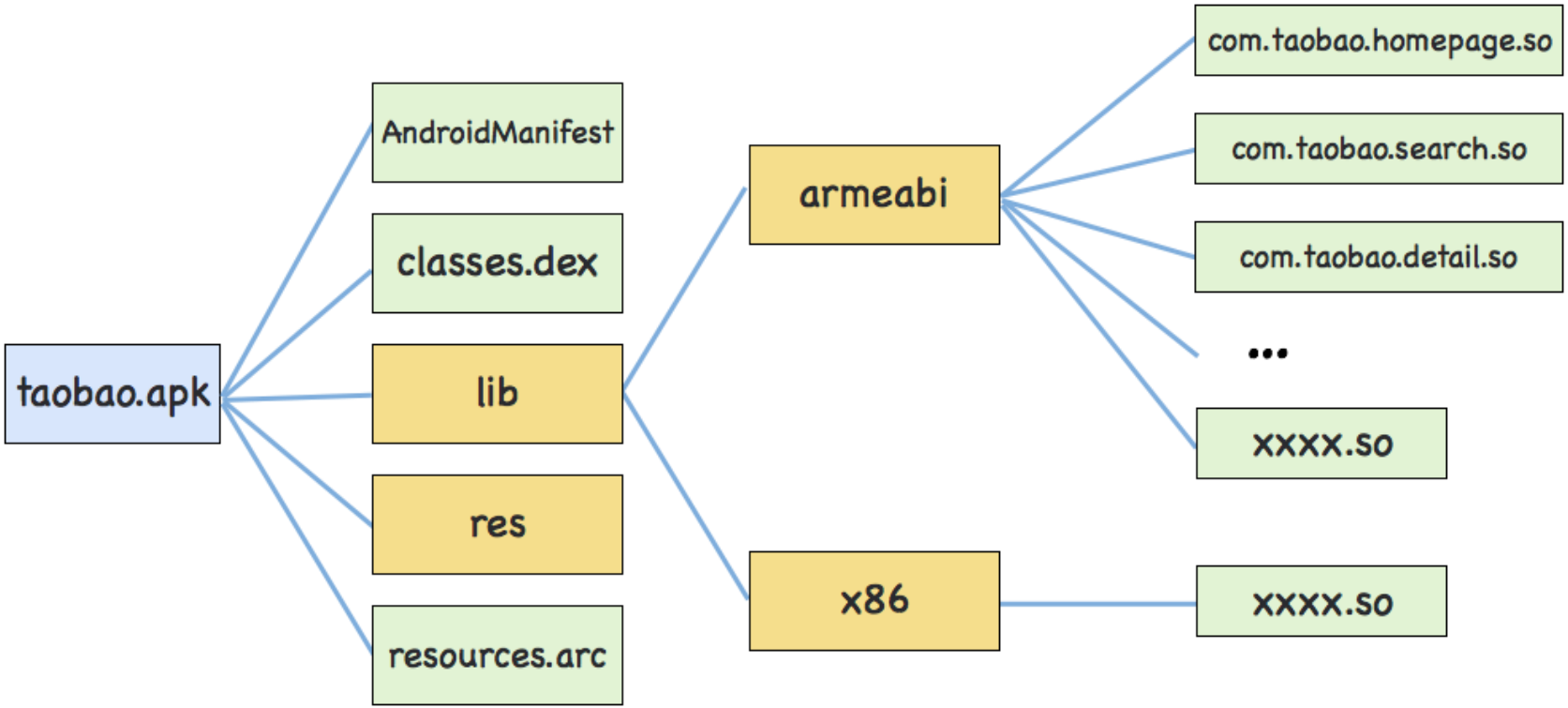
1. 实现完整的组件生命周期的映射，类隔离等机制
2. 实现工程独立开发，调试的功能，工程模块独立
3. 快速增量的更新修复能力，快速升级

透明，灵活，稳定，敏捷，高性能

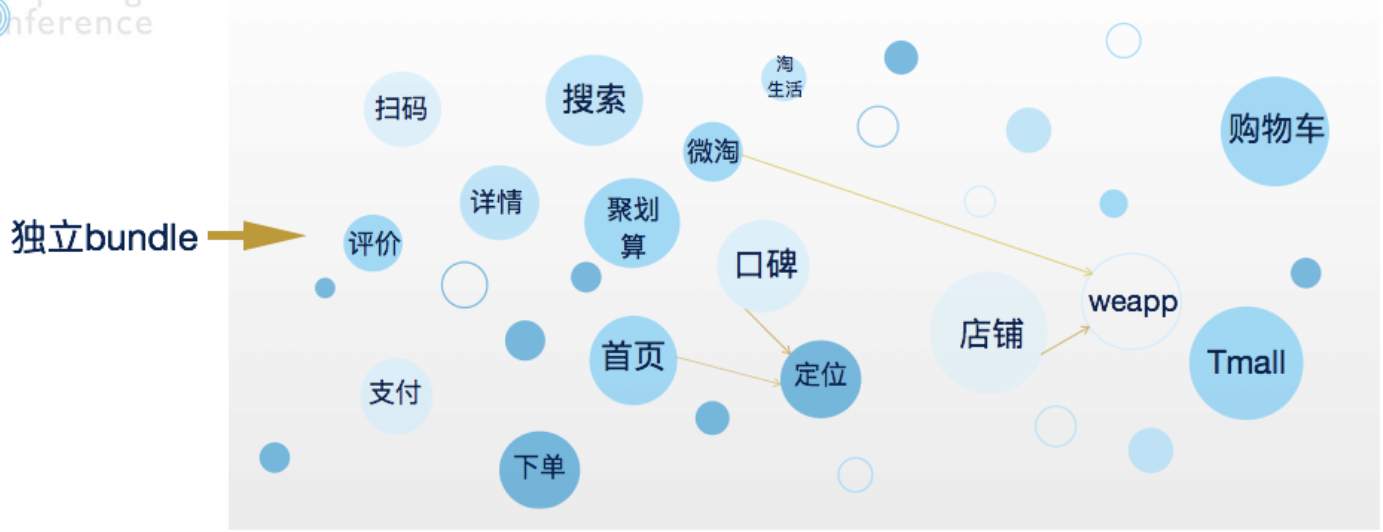
扫码观看大会视频

二、组件化实现

APK包结构

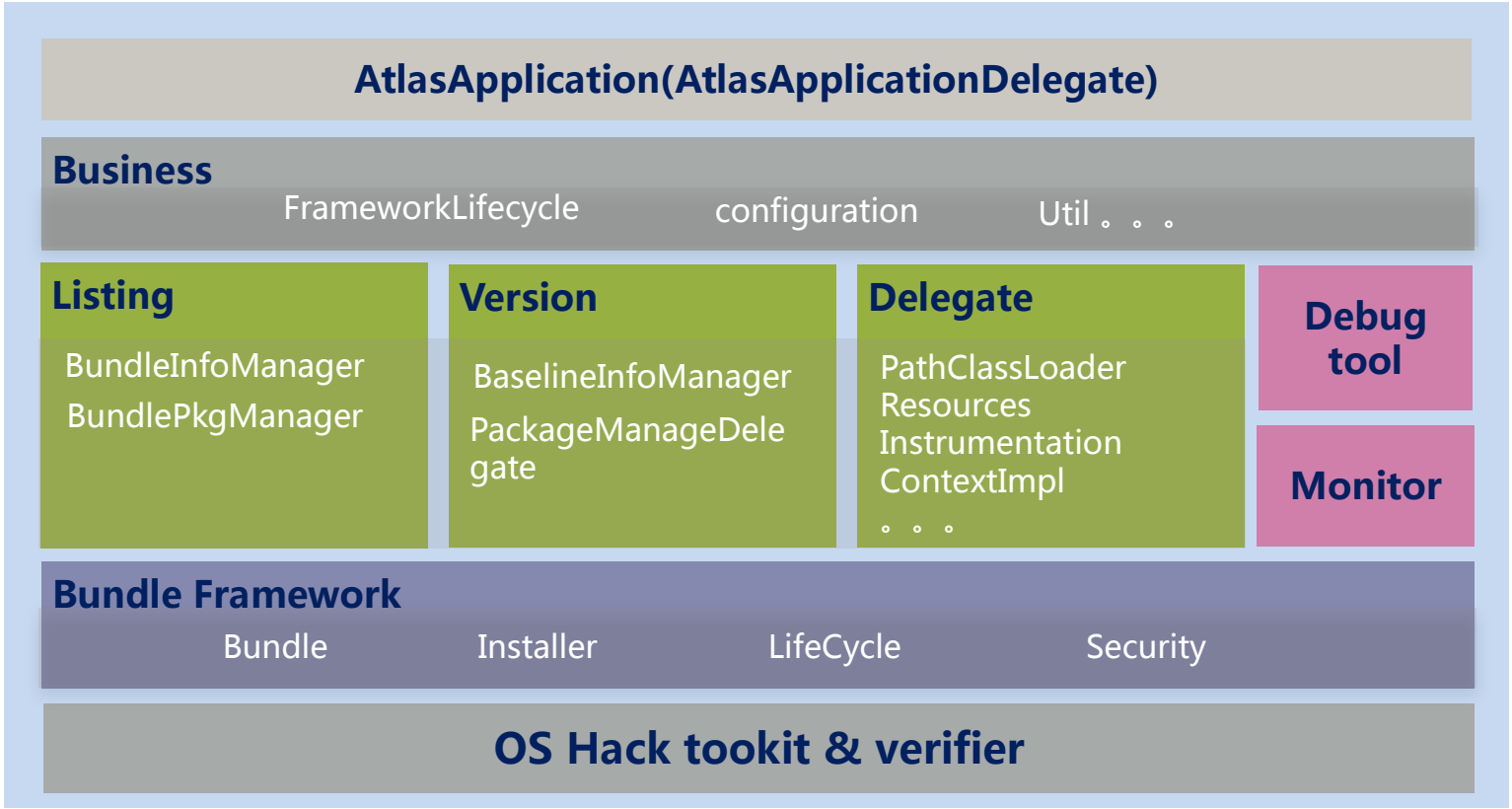


模块分层



扫码观看大会视频

容器设计



应用接入层

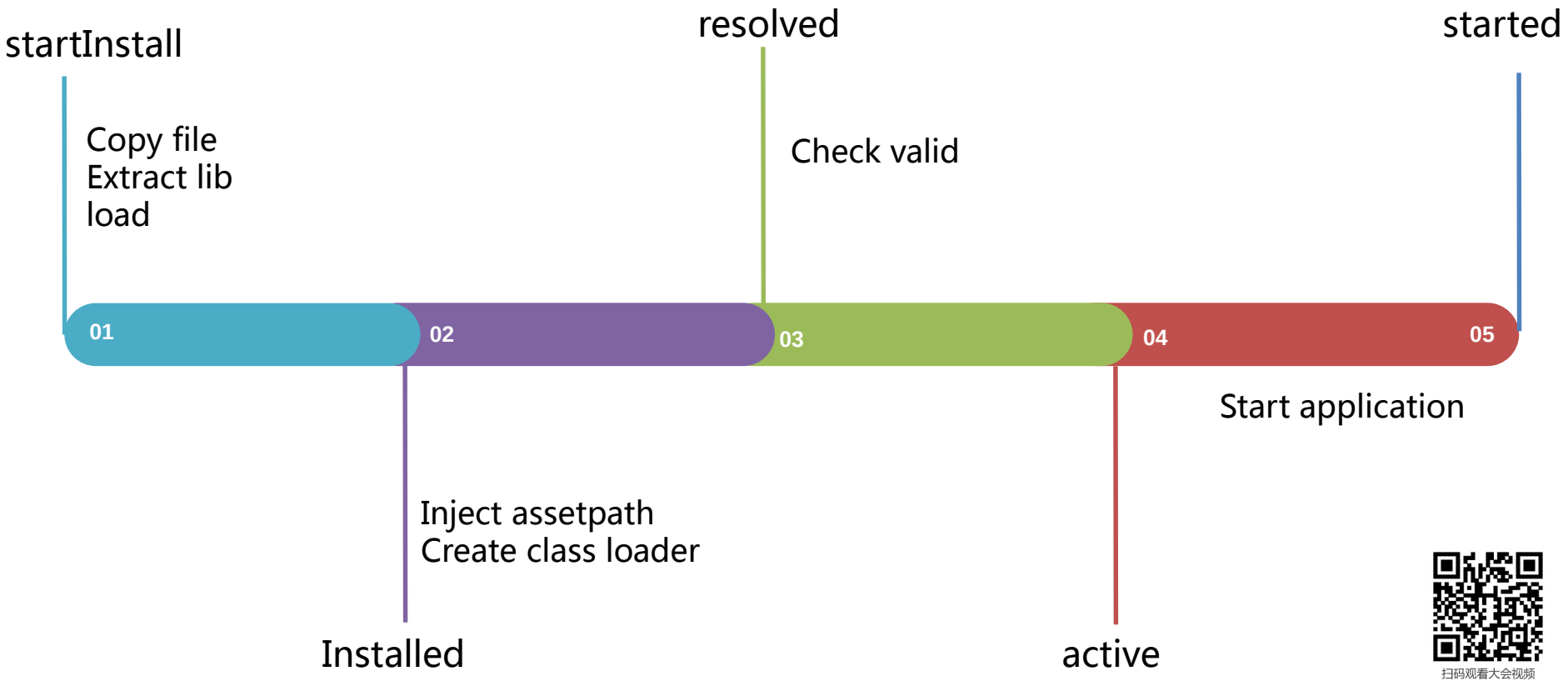
Business层

运行期管理层

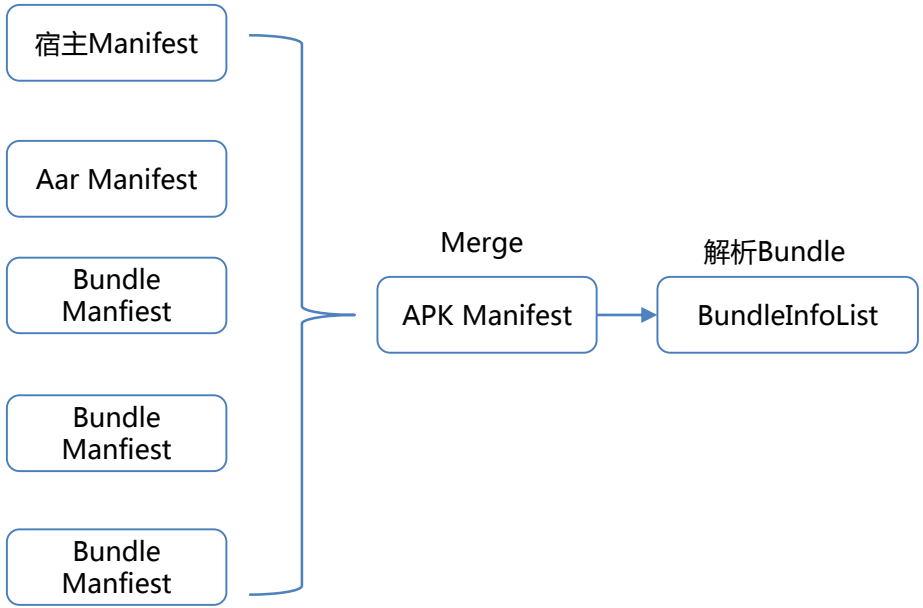
Bundle Framework层

Hack层

扫码观看大会视频


startInstall
Copy file
Extract lib
load
resolved
Check valid
started
01
02
03
04
05
Start application
Inject assetpath
Create class loader
Installed
active

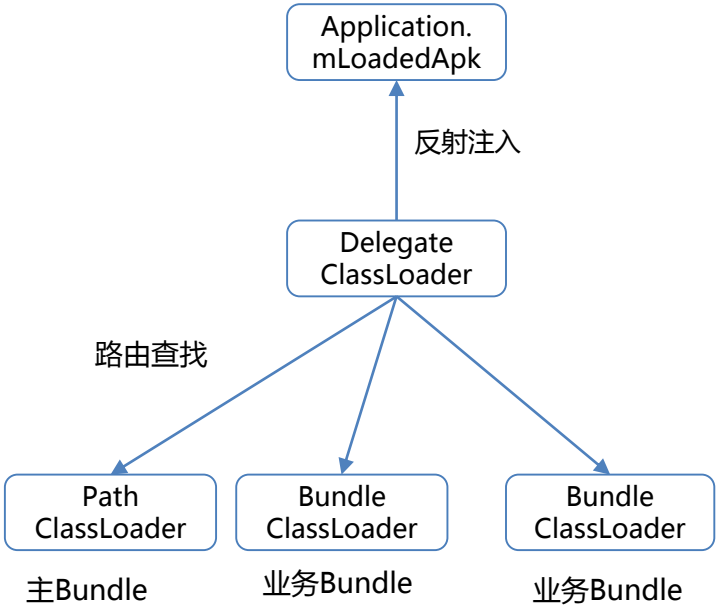
Manifest处理



1. 构建期进行全量Merge操作
2. Bundle的依赖单独Merge，生成Bundle的Merge manifest
3. 解析各个Bundle的Merge Manifest，得到整包的BundleInfoList

扫码观看大会视频

类加载

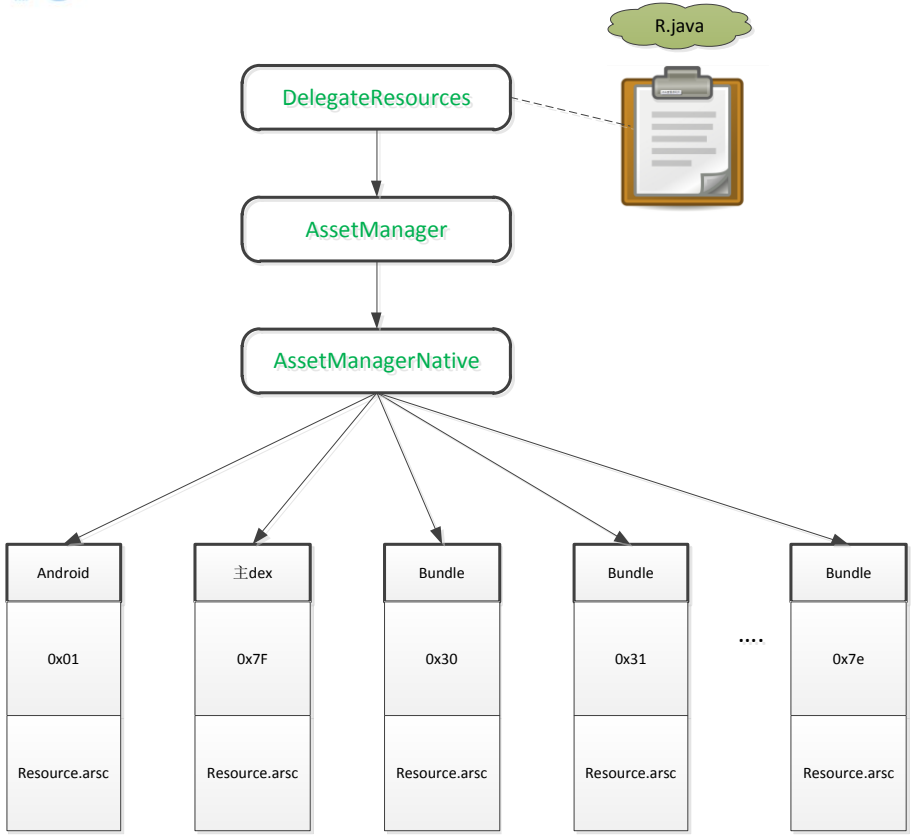


1. DelegateClassLoader以PatchClassLoader为父classLoader, 找不到的情况下根据BundleList找到对应的BundleClassLoader

2. BundleClassLoader的父对象为BootClassLoader, 包含PathClassLoader对象，先查找当前classLoader, 再查找当前PathClassLoader

扫码观看大会视频

资源加载



1. 所有的Bundle Res资源都是加入到一个DelegateResources
2. Bundle Install的时候将Res,So等目录通过反射加入(AssetManager.addAssetPath)
3. 通过不同的packageId来区分Bundle的资源ID
4. 覆盖getIdentifier方法，兼容5.0以上系统

扫码观看大会视频

按需加载


```
//ActivityThread.java
private Activity performLaunchActivity(ActivityClientRecord r,
                                       Intent customIntent) {
    ...
    Activity activity = null;
    try {
        java.lang.ClassLoader cl = r.packageInfo.getClassLoader();
        activity = mInstrumentation.newActivity(
                cl, component.getClassName(), r.intent);
        ...
    } catch (Exception e) {
        ...
    }
    ...
}

//Instrumentation.java
public Activity newActivity(ClassLoader cl, String className,
        Intent intent)
        throws InstantiationException, IllegalAccessException,
        ClassNotFoundException {
    return (Activity)cl.loadClass(className).newInstance();
}
```

1. Bundle之间隔离，通过Android四大原生Component进行交互
2. 在DelegateClassLoader，根据BundleInfoList信息，得到组件所在的bundle，如果没加载，进行install,dexopt等操作

扫码观看大会视频

组件新格式

- **Awb(业务Bundle)**
  - /AndroidManifest.xml (mandatory)
  - /classes.jar (mandatory)
  - /res/ (mandatory)
  - /R.txt (mandatory)
  - /assets/ (optional)
  - /jni/<abi>/*.so (optional)
  - /proguard.txt (optional)
  - /lint.jar (optional)

与AAR一致(不添加本地lib)，构建的时候做依赖仲裁区分

- **solib(so库)**
  - /armeabi
    - libweexv8.so
  - /x86
    - libweexv8.so

Native so库的依赖

依赖仲裁

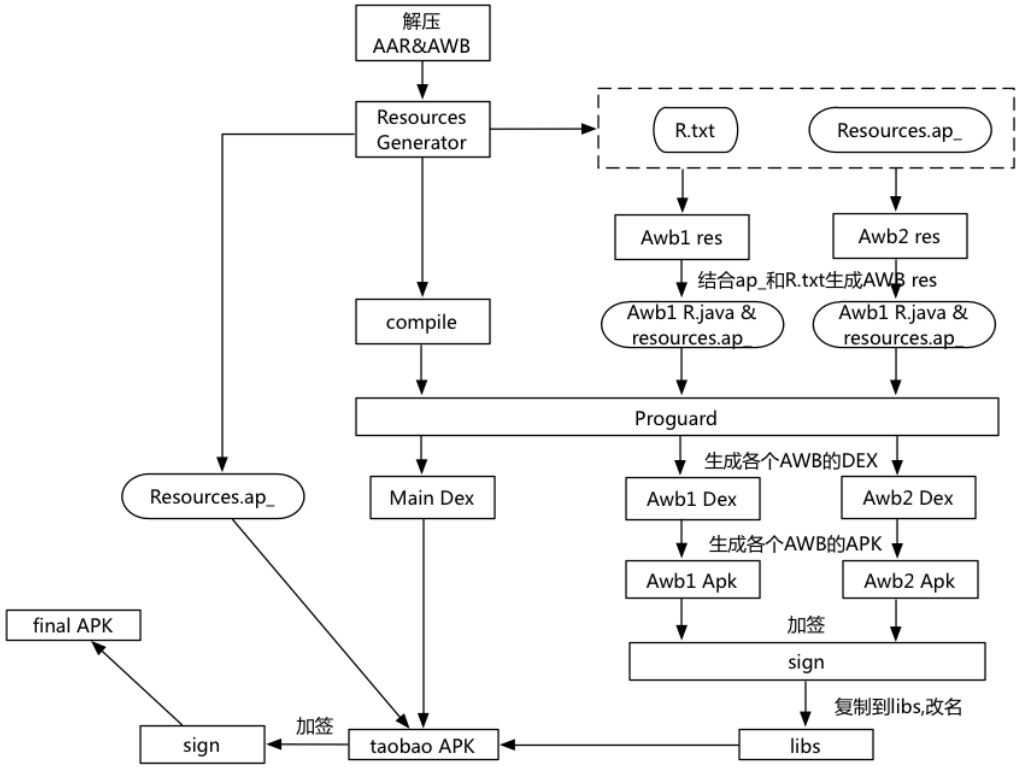
```
dependencies {
    compile('com.android.support:appcompat-v7:23.1.0')
    compile 'com.taobao.android:atlas:2.3.73'
    compile('com.taobao.android.atlas.demo:lib:2.0.2.1@jar')
    compile('com.taobao.android.atlas.demo:ShareLibrary:2.0.2.1@aar') {
        transitive = true
    }
    compile('com.taobao.android.atlas.demo:FirstBundle:3.0@awb') {
        transitive = true
    }
    compile('com.taobao.android.atlas.demo:RemoteBundle:2.0.2.1@awb') {
        transitive = true
    }
}
```

```
{
  "awbs": {
    "com.taobao.android.atlas.demo:RemoteBundle:awb:2.0.2.1": [

    ],
    "com.taobao.android.atlas.demo:FirstBundle:awb:3.0": [
      "com.taobao.android:proc_guard_so:solib:1.1.3",
      "org.jetbrains.kotlin:kotlin-runtime:jar:1.0.2",
      "org.jetbrains.kotlin:kotlin-stdlib:jar:1.0.2"
    ]
  },
  "mainDex": [
    "com.taobao.android.atlas.demo:lib:jar:2.0.2.1",
    "com.android.support:appcompat-v7:aar:23.4.0",
    "com.android.support:support-annotations:jar:23.4.0",
    "com.android.support:support-v4:aar:23.4.0",
    "com.android.support:support-vector-drawable:aar:23.4.0",
    "com.android.support:animated-vector-drawable:aar:23.4.0",
    "com.android.databinding:library:aar:1.1",
    "com.taobao.android.atlas.demo:ShareLibrary:aar:2.0.2.1",
    "com.taobao.android.atlas.demo:SecondShareLibrary:aar:2.0.2.1",
    "com.taobao.android:downloader:jar:1.5.0.4",
    "com.taobao.android:tnet-so:solib:3.0.14",
    "com.android.databinding:compiler:jar:2.1.0",
    "com.googlecode.juniversalchardet:juniversalchardet:jar:1.0.3",
    "com.android.databinding:adapters:aar:1.1",
    "com.android.databinding:baseLibrary:jar:2.1.0",
    "com.taobao.android:atlas:jar:2.3.73"
  ]
}
```

1. 将宿主Bundle和业务Bundle的依赖分别打包
2. 按照最短路径，第一声明原则进行树状仲裁

APK构建



1. Awb的res根据宿主的resource.ap_加包内资源构建
2. Awb的R文件由bundle的R资源+宿主R资源合并而来
3. 修改Aapt，每个awb有不同的packageId
4. Proguard为统一优化混淆，多个output产物

扫码观看大会视频

三、动态性实现

动态性的支持

包大小缩减

动态发版、修复

增量动态化方案

动态部署

业务发布、问题修复

基于自研差量算法，主Bundle基于classloader机制，业务Bundle基于差量merge,支持全业务类型

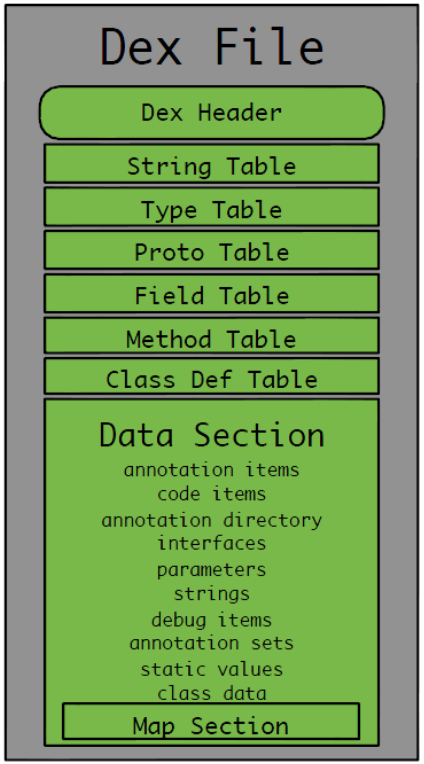
Andfix
(plugin)

快速故障修复

基于Native hook的方式来做，主要做方法的修改。

做法：工程构建期适配，研发人员开发一套代码二套方案通用

Dex Patch生成




```
struct ClassDef {
  uint16_t class_idx_;  // inde
  uint16_t pad1_;  // padding =
  uint32_t access_flags_;
  uint16_t superclass_idx_;  //
  uint16_t pad2_;  // padding =
  uint32_t interfaces_off_;  //
  uint32_t source_file_idx_;  /
  uint32_t annotations_off_;  /
  uint32_t class_data_off_;  //
  uint32_t static_values_off_;
```

```
struct ClassDataMethod {
  uint32_t method_idx_delta_;
  uint32_t access_flags_;
  uint32_t code_off_;
```

1. **Dex->Smali代码解析对比**
2. **Diff Dex处理**
   - 删除类：class_data_off_ => -1，删除所有的class_data字节
   - 新增类
   - 修改类：指定方法code_off_ => -1,删除方法字节和debugInfo
3. **Dex Merge处理**

扫码观看大会视频

资源Patch生成

- 业务Bundle: 文件Md5 diff / BSDiff
- 主Bundle:
  - 预留空资源 (Add Entry Count)
  - Keep已有资源
  - 资源Overlay

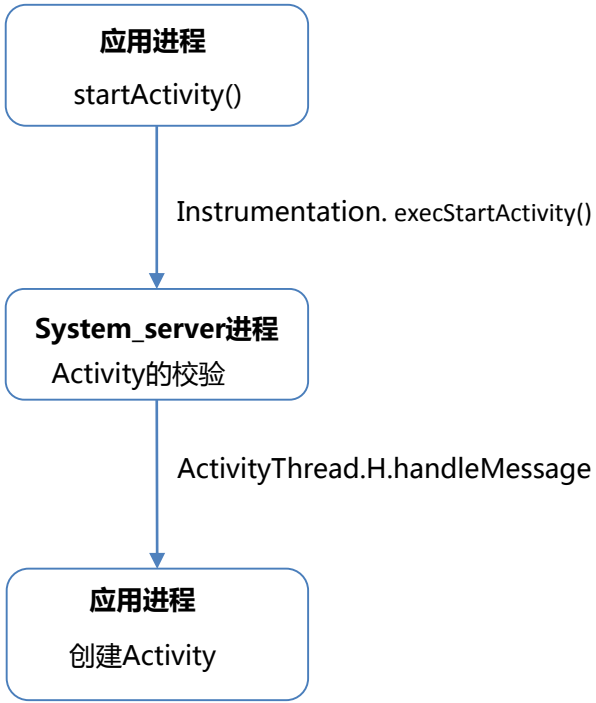
a	b	c	d

基线resources.arsc

NO_ENTRY	NO_ENTRY	NO_ENTRY	d1
e	f	g	h

补丁resources.arsc

Activity新增



1. 在Manifest预埋一个StubActivity
2. Instrumentation. execStartActivity()阶段替换为StubActivity
3. 注入ActivityThread.H, 替换为TargetActivity

扫码观看大会视频

工程实践

- **AP(基线包)**
  - Android.apk
  - Mapping.txt
  - AndroidManifest.xml
  - Usage.txt
  - Dependency.txt

```
patchConfigs{
    debug {
        createTPatch true //生成动态部署包
        createAPatch false //生成Andfix包
        basePatchDependency  "com.taobao.android:taobao-android-debug:6.0.0@ap"
        filterFiler file("filter-debug.txt")
    }
    release {
        createTPatch true //生成动态部署包
        createAPatch true //生成Andfix包
        basePatchDependency  "com.taobao.android:taobao-android-release:6.0.0@ap"
        filterFiler file("filter-release.txt")
    }
}
```

1. 每次发布APK包同步发布AP(基线包）到Maven仓库
2. 支持连续版本构建(容器版本不变的情况下)

扫码观看大会视频

版本升级

01

5.3.0

基线版本，应用市场发布

Taobao-5.3.0.apk

02

5.3.1

业务版本，动态部署

Patch-5.3.1@5.3.0.patch

03

5.3.2

业务版本，动态部署

Patch-5.3.2@5.3.0.patch

Patch-5.3.2@5.3.1.patch

04

5.3.3

业务版本，动态部署

Patch-5.3.3@5.3.0.patch

Patch-5.3.3@5.3.1.patch

Patch-5.3.3@5.3.2.patch

采用非Index的增量，支持一次产出多个版本的动态包

扫码观看大会视频

四、周边优化点

优化tips

1. Bundle重复资源合并
2. Bundle依赖校验
3. 类库瘦身，方法数裁剪
4. 依赖查询库
5. 混淆Mapping处理

开源准备中...

扫码观看大会视频

获取演讲资料和更多阿里移动技术动态

手机淘宝技术团队
微信公众号

阿里百川
微信公众号

扫码观看大会视频

2016 The Computing Conference

THANKS