# 咕咚首页滑动卡顿分析

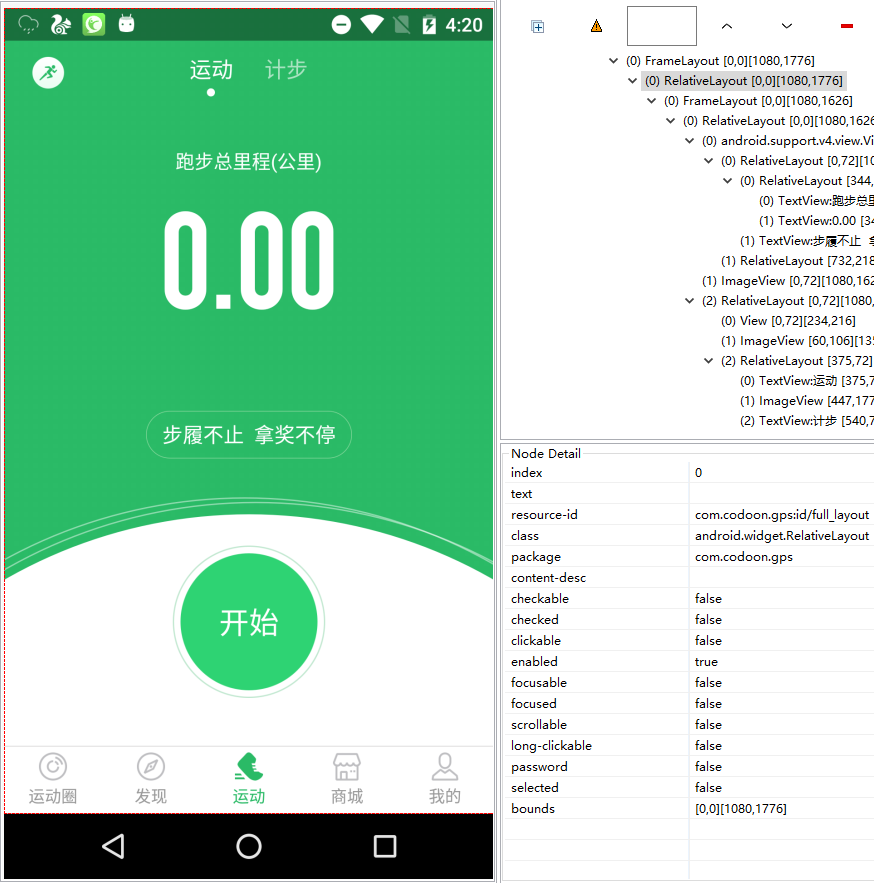
先说结论：由于手指抬起UP事件的时候，有做耗时操作，导致VSYNC消息阻塞，整体UI就丢帧。

接着一步一步讲一下我的分析过程。

## 一、反编译并添加基本的Log

第一步是从反编译编译的类文件和资源文件中找到首页的Activity及其使用的布局文件。

首先是Manifest文件，找到欢迎界面WelcomeActivity。里面有startActivity跳转到登录页面和首页。仔细查看了下，里面有好几个activity，如SlideActivity，ViewPage，SportingGDActivity。为了快速定位，利用了uiautomatorviewer查看首页UI元素的id。



content下的第一个View id为full\_layout，接着对资源文件进行全局搜索。

find . -name "\*.xml" | xargs grep -rn "full\_layout"



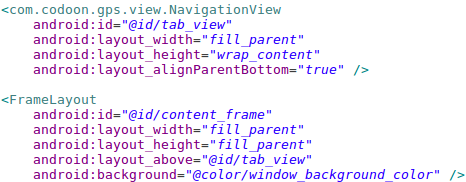
有两个布局在使用。从命名来看，首页应该就是slide.xml了。并且full\_layout的id定义在一个公共的xml里面。



通过这个ID可以搜索到，确实SlideActivity用了这个资源。



确定了首页，然后打开SlideActivity及其资源文件看一下。



一个是底部的导航栏，一个是中间的内容显示，采用Fragment实现。

接着找主页Fragment使用了那个布局。先从onCreate看起



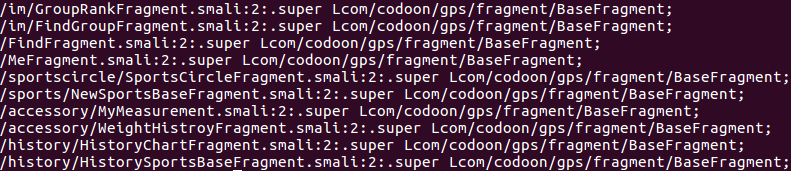
这里通过工厂模式创建了一个Fragment，然后再调用chaneToModule切换到新建的Fragment。从FragmentFactory的代码看，里面封装了创建Fragment以及切换Fragment的功能。

创建的Fragment是一个NewSportsBaseFragment。



没找到任何类继承这个类，那么这个就是运动板块所使用的Fragment了。

这个类继承自com/codoon/gps/fragment/BaseFragment。



这些类也继承了BaseFragment。那么来看看咕咚的BaseFragment做了些什么？有点好奇。

打开smali文件，此类继承自原生的Fragment并实现了View$OnTouchListener接口，用来辅助处理View的Touch事件。包括自动打开关闭VelocityTracker，监测速度，如果检测到向左或者向右滑动，则发出广播，并通知监听器mOnSlideListener。

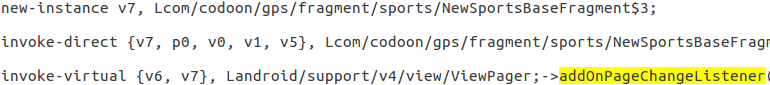
继续分析NewSportsBaseFragment，主要关注以下几点：

1. UI的布局、切换等
2. 左右滑动效果怎么实现的，分析为什么会出现近乎跳变式滑动

首先是onCreateView，这里面加载了资源文件0x7f0304a1(sports\_pre\_start\_layout.xml)。

创建了一个mMainBGDrawable: GradientDrawable，这个就是用来在滑动时显示渐变效果的。

创建了一个内部类实例，实现了OnPageChangeListener。里面做了页面滑动的相关动画操作。**后面会看到这个类就是问题的所在。**



NewSportsBaseFragment$3.smali里面实现了三个接口方法：

.method public onPageScrollStateChanged(I)V

.method public onPageScrolled(IFI)V

.method public onPageSelected(I)V

接着在代码中里可以看到：

new-instance v2, Lcom/codoon/gps/fragment/sports/RotatePageTransformer;

invoke-virtual {v0, v1, v2}, Landroid/support/v4/view/ViewPager;->setPageTransformer(ZLandroid/

通过设置PageTransFormer来实现ViewPager翻页效果。

可以ViewPager在回调transformPage的时候，应该是按照插值器渐进变化，不应该跳变啊。

但实际上，除了ViewPager滑动卡顿，“开始”按钮的动画一样会卡顿。会不是是插值器计算有问题呢？或者速度计算有问题？ViewPager有什么接口可以造成这种效果呢？

从布局来看，ViewPager之上没有有其他自定义View用来拦截Touch事件。而咕咚自己定义的ControlableViewPager只是限制了本身能不能滑动，所以按理说ViewPager本身的滑动效果是支持平滑翻页的。

到这一步就可以从一些关键的地方加日志来定位问题了。

从动画的关键环节.method public transformPage(Landroid/view/View;F)V入手比较好。先看ViewPager通知PageTransFormer的消息有没有跳变。

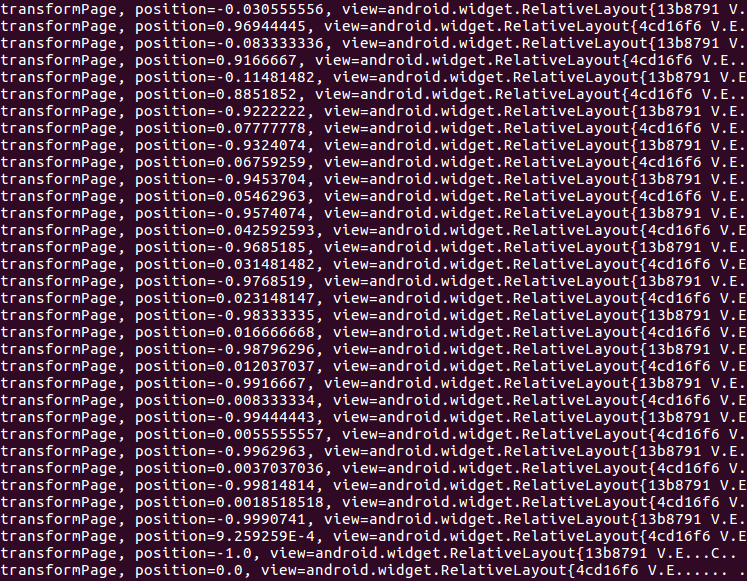
原来的代码是返回到goto\_0，代码里面有三处都是goto :goto\_0



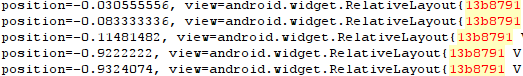
所以要加日志的话，那就将goto\_0改为我要加的日志，然后再goto: goto\_1。goto\_1就是原来的返回。



添加的日志为Log.d("Rotate", "transformPage, position=" + position + ", view=" + view)。然后回编译，签名，重新安装。测试一下，滑动的时候，日志有了。



从日志来看，确实在滑动的时候，position从-0.11跳变到了-0.92.



接下来，就要找出来，为什么ViewPager传出来的值会跳变了。

ViewPager通知mPageTransformer的位置在：

protected void onPageScrolled(int position, float offset, int offsetPixels)

if (mPageTransformer != null) {

final int scrollX = getScrollX();

final int childCount = getChildCount();

for (int i = 0; i < childCount; i++) {

final View child = getChildAt(i);

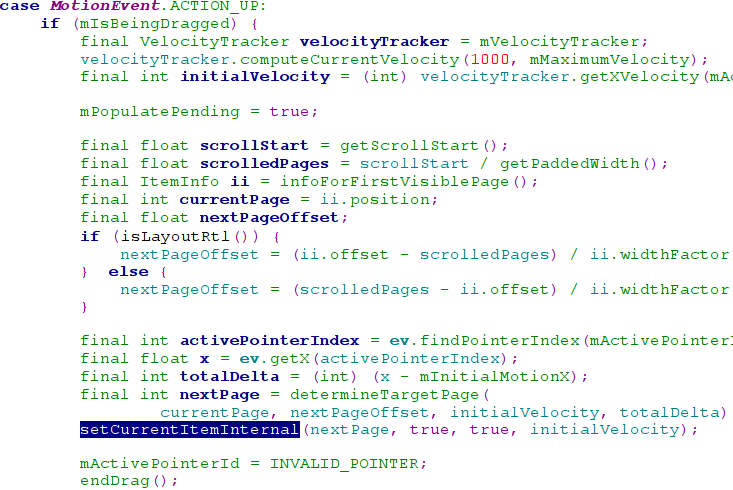
final LayoutParams lp = (LayoutParams) child.getLayoutParams();

if (lp.isDecor) continue;

final float transformPos = (float) (child.getLeft() - scrollX) / getClientWidth();

mPageTransformer.**transformPage**(child, transformPos); //依次通知每个子View的变化

按照理论上说，



UP事件后，会调用setCurrentItemInternal来调转到指定Page。

void setCurrentItemInternal(int item, boolean smoothScroll, boolean always, int velocity) {

接着会调用scrollToItem(item, smoothScroll, velocity, dispatchSelected)，这里有两个关键的地方，偏移位置和startScroll。

**destX** = (int) (width \* Math.max(mFirstOffset,

Math.min(curInfo.offset, mLastOffset))

if (smoothScroll) {

**smoothScrollTo**(destX, 0, velocity);

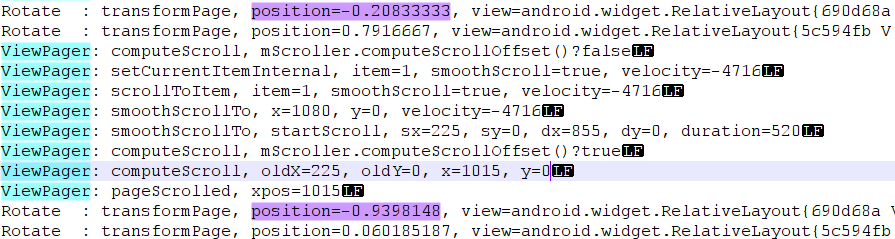
smoothScrollTo()方法通过与mScroller和View绘制中的computeScroll()方法配合完成平滑滑动效果。

mScroller.startScroll(sx, sy, dx, dy, duration);

到时候在ViewPager中加日志这里是重点观察的位置。主要观察初始化的值，以及插值器第一次计算的值是否正确。

## 二、在反编译的ViewPager中添加日志

接下来，就是在ViewPager中加日志了。因为咕咚使用了V4包里面的ViewPager，所以先在V4包的ViewPager中添加相关日志，然后编译出jar包。然后反编译出smali，替换咕咚中相应的文件后回编译成apk。在关键流程添加日志后，发现：



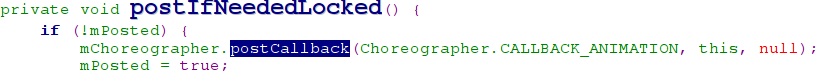
可以看到在手抬起后smoothScrollTo到下次绘制执行computeScroll这两段之间，出现的跳变。通过Scroller计算出来的第一个x就是1015，太大了。

为什么Scroller有问题呢？仔细看了一下从startScroll到computeScrollOffset中间花了209ms。刷新太慢了，而这个默认的插值器又是属于初始速度很快的插值器，所以对于滑动总时长才520ms，209ms不更新，跳变就很明显。

那为什么有209ms没有刷新呢？ViewPager原生在computeScroll的时候都是会调用

ViewCompat.postInvalidateOnAnimation(this)

这个操作会最终向Choreographer的CallbackQueue提交一个叫InvalidateOnAnimationRunnable的回调，回调的时候，会通过View的invalidate来重绘UI完成下一步动画。

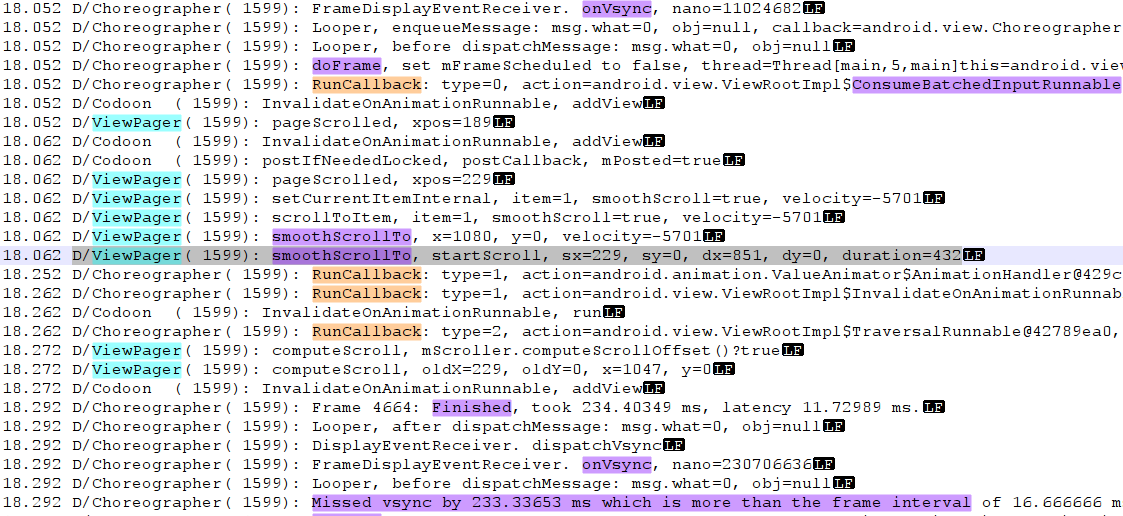


而Choreographer则会等待FrameDisplayEventReceiver传过来的Vsync消息，进行分发输入事件、执行动画Callback、执行View树遍历Callback等。这样按照理论这样UI可以持续更新才对。

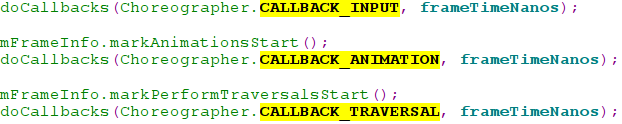
到这一步，光从应用层分析就比较麻烦了。从框架层加入日志会比较好分析。

## 三、修改框架加Log

首先在postInvalidateOnAnimation调用的流程和底层上报Vsync消息的流程添加关键的日志，打开Choreographer的debug日志。经过几轮调试，并在Looper的消息添加和处理的地方都加好日志。



可以看到第一行的VSYNC消息过来之后，把自己作为一个Runnable提交了一个Choreographer中的FrameHandler等待处理。第四行，FrameHandler开始处理doFrame。这里执行了三类重要的回调。



依次为分发输入事件、执行动画Callback、执行View树遍历Callback。

日志依旧是从18.062秒到18.252秒跳变。但这次日志多了，可以看到RunCallback的时候，只跑了第一个分发输入事件的回调，后面就到18.252秒了。这里就开始怀疑是不是在分发UP事件的时候有做耗时操作。

UP事件由ViewPager拦截并在UI线程分发，那是谁做了耗时操作呢？从之前分析来看，有两个类做了ViewPager的事件监听。一个是PageTransFormer，另一个是NewSportsBaseFragment内的内部成员OnPageChangeListener。而PageTransFormer只有一个回调transformPage，不太可能会知道UP事件。只有OnPageChangeListener会知道UP事件。

UP事件的时候，OnPageChangeListener会有两个回调：

1. onPageScrollStateChanged，UP的时候通知进入SCROLL\_STATE\_SETTLING滑动中的状态。
2. onPageSelected，UP的时候通知即将进入哪一页。

最后通过修改分析出是onPageSelected中做了耗时操作。

## 四、修改卡顿问题

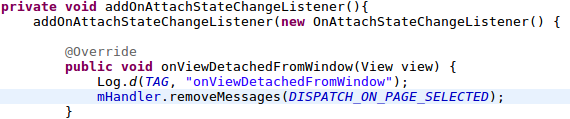
解决耗时操作的方法，最直接的就是另开一个线程。但是如果把onPageSelected用非UI线程分发，那么UI更新就会报错。

“android.view.ViewRootImpl$CalledFromWrongThreadException: **Only the original thread that created a view hierarchy can touch its views**.”

那么只好采用间接的权宜之策，延迟分发。在ViewPager中创建一个Handler，复用主线程的Looper。



延迟的时间就是动画结束的时间。然后添加一个OnAttachStateChangeListener，用于在Activity结束的时候，remove所有的消息。



将修改后的ViewPager相关文件替换，然后回编译，签名。经过测试，滑动的时候没有卡顿了，动画延迟到滑动结束再执行。

当然最终的解决办法还是避免在UI线程做耗时操作。运动圈的滑动同样有卡顿问题，考虑为同类问题。

## 简历

#### 黄盼 Android开发工程师

手机:[18938913845](tel:18938913845) 邮箱: [412625135@qq.com](mailto:412625135@qq.com)

#### 个人简介

目前任职于海能达通信股份有限公司。三年间经历了本公司G2.0产品最核心的通话应用从需求规划到1.0即将发布的全过程。通过该项目对需求、框架设计、模块设计、产品稳定性和质量有了比较深的认识。

#### 项目经历

我们公司的通话应用整合了公网通话，PDT常规通话，PDT集群通话，TETRA直通通话，TETRA集群通话，POC集群通话对所有的通话进行统一管理及显示。

**我做了什么**

* **负责实现通话应用的框架及核心功能。**
* 框架搭建。包括消息驱动模型，通话管理，监听器模型，对接UI层和通话模块，PTT等物理按键分发机制。
  + 编写和维护了PDT常规通话模块，TETRA直通通话模块，UI显示模式控制模块，通知模块等多个重要的模块。
* 快速响应需求变化、快速定位开发和测试问题，对模块进行扩展或者重构。
* **负责设计和实现集成POC通话模块的框架和接口。使得第三方通话模块可以通过SDK集成到我们的通话系统。**
* 跨部门协调沟通，作为APP部门的接口人参与方案的讨论、制定开发计划。该子项目属于跨部门项目，涉及宽带系统、窄带系统、窄带终端、德国子公司、APP部门。其中APP部门作为终端和系统之间的衔接部门，有很重要的协调沟通作用
  + 框架搭建。包括PocManager模块设计、对CallManager接口及对外接口设计。
* 业务实现。自动网络选择模块，定时器模块、异常处理模块、呼叫业务。
* **负责实现通话模块对外的接口，包括通话功能接口，对联系人应用接口，PTT按键注册使用接口。**
* **修改扩展sipdroid客户端和osip服务器源码，使其支持半双工的单呼和组呼。**自定义组呼的SIP信令流程，自定义申请和释放话权信令， 在服务器端和客户端实现这些信令，并相应修改语音流程。

#### 其他

* 扩展Android框架的RemoteView机制，使得框架层可以支持使用了ViewPager的widget，并完善其滑动，点击，EmptyView等功能。我们部门的app包括联系人，短消息，信道，日历等widget都采用这种可以左右滑动的widget。
* 熟悉View的绘制、Touch事件分发、Handler消息原理，Telephony等框架中的重要机制。

#### 自我描述

不到三年时间，薪资从9.5k升至16 k，同一届应届生中应该是提升最快的。作为两个应届生的导师，整个app部门的核心通话应用的二把手，负责了核心的通话管理部分，且负责的通话业务是团队中最多的，跨部门沟通的主要接口人。

**标签: “丰富的开发调试经验” “架构设计和重构经验” “责任心” “团队合作” “电子科技大学” “硕士” “CET6”**

****