https://blog.csdn.net/vicwudi/article/details/100191529?spm=1001.2101.3001.6650.5&utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-5.pc\_relevant\_paycolumn\_v3&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-5.pc\_relevant\_paycolumn\_v3&utm\_relevant\_index=10

**1 屏幕刷新机制**

基础概念

**2 卡顿分析利器-Ssystrace工具**

**2.1 抓取systrace**

**2.1.1 通过android studio的Android device monitor抓取**

<https://www.cnblogs.com/blogs-of-lxl/p/10926824.html>

**3 分析sysTrace**

如果在UI Thread上做太多的工作，需要找出哪些方法消耗了太多的CPU时间。一种方法是添加跟踪标记到您认为会导致这些瓶颈的方法，以查看这些函数调用是否显示在systrace中。 如果您不确定哪些方法可能会在UI线程上造成瓶颈，请使用Android Studio的内置CPU分析器，或者生成跟踪日志并使用Traceview查看它们。

　 虽然Systrace无法定位到某一行需要优化的代码，但通过Alerts和Frames以根据TraceView分析具体函数花了多长时间来进一步优化代码提高性能。

**3.1 线程状态查看**

https://androidperformance.com/2019/07/23/Android-Systrace-Pre/#/%E6%AD%A3%E6%96%87

Systrace会用不同的颜色来标识不同的线程状态，通过查看线程状态我们可以知道目前的瓶颈是什么，是cpu执行慢还是因为Binder调用，又或是进行IO操作，又或是拿不到CPU时间片。

线程状态主要有五种。

（1）绿色：运行中（Running）。

只有在该状态的线程才可能在CPU上运行。而同一时刻可能有多个线程处于可执行状态，这些线程的task\_struct结构被放到CPU的可执行队列中。调度器的一个任务就是从各个CPU可执行队列中取出一个线程在指定的CPU上运行。

（2）蓝色：可运行（Runnable）。

作用：Runnable状态的线程持续的时间越长，则表示CPU的调度越忙，没有及时的处理这个任务。

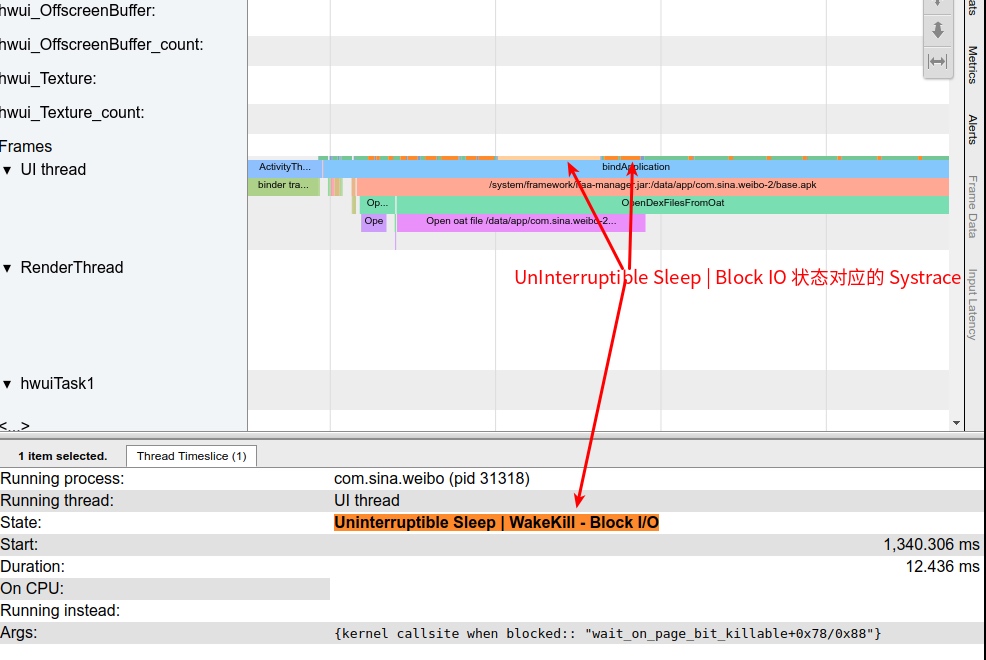
是否后台有太多的任务在跑？没有及时处理是因为频率太低？没有及时处理是因为被限制到某个CPUSet里面，但是CPU很满？此时Running的任务是什么，为什么？

（3）白色：休眠中（Sleep）。

可能是因为线程在互斥锁上被阻塞。

作用：这里一般是在等事件驱动。

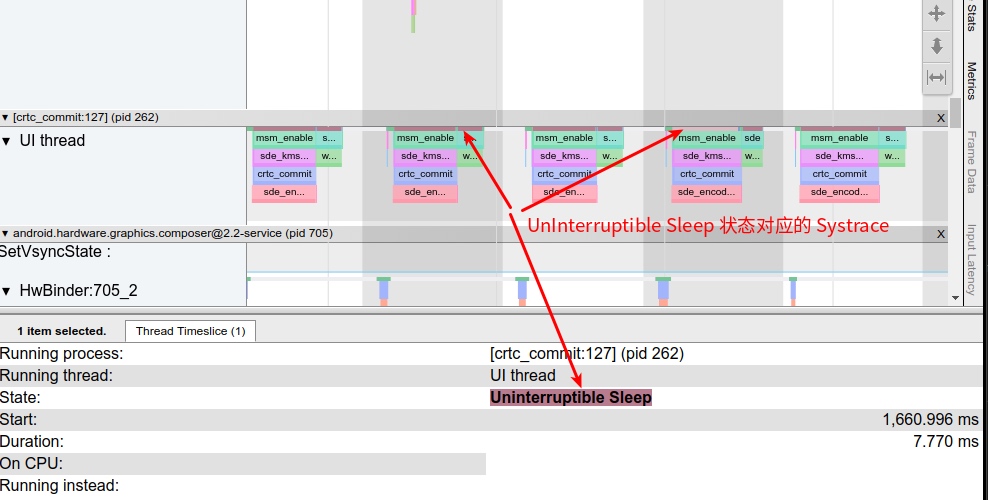
（4）橘色：不可中断的睡眠状态（uninterrupted sleep）。

线程在I/O上被阻塞或等待磁盘操作完成，一般会标识出此时的callsite:wait\_on\_page\_locked\_killable。

作用：这一般是标识I/O操作慢，如果有大量的橘色不可中断的睡眠状态出现，那么一般是进入了低内存状态，申请内存的时候触发pageFault，linux系统的page cache链表中有时候会出现一些还没有准备好的page（即还没有把磁盘中的内容完全读出来），而正好用户在访问这个page的时候就会出现wait\_on\_page\_locked\_killable阻塞了，只有系统当I/O操作很繁忙时候，每笔的I/O操作都需要排队等待时，极其容易出现且等待的时间往往会比较长。

（5）紫色：不可中断的睡眠态（uninterruptible sleep）。

线程在内核操作上被阻塞。

作用：一般是陷入了内核态，有些情况是正常的，有些情况是不正常的，需要具体分析。

**3.2 线程唤醒信息分析**

一个线程被唤醒的信息往往比较重要，知道它被谁唤醒，那么我们也就知道了它们之间的调用等待关系，如果一个线程出现了一段比较长的sleep情况，然后被唤醒，那么我们就可以去看是谁唤醒了这个线程，对应的就可以查看唤醒者的信息，看看唤醒者为什么这么晚才唤醒。

一个常见的情况是：应用的主线程用Binder与SystermServer的AMS进行通信，但是恰好AMS的这个函数在等待锁释放（或者这个函数本身执行的时间比较长），应用主线程就需要等待比较长的时间，那么就会出现性能问题，比如出现响应慢或者卡顿，这就是为什么后台有大量的进程在运行，或者跑完Monkey之后，整机性能会下降的一个主要原因。