# 一Android 的app性能测试

包含几项

资源消耗

内存泄漏

电量功耗

耗时

网络流量消耗

移动终端相关资源利用率

桢率

渲染等

## 1内存

### 1.1内存现象

1. 内存抖动
2. 大内存对象被分配
3. 内存不断增长
4. 频繁gc

### 1.2.引入概念：空闲状态、中等状态、满规格

空闲状态：打开应用后，点击home，所有应用在后台

中等规格：对应用操作时间较长

满规格：对应用操作时间较短

### 1.3.内存测试清单：

空闲状态下内存消耗

中等规格内存消耗

满规格内存消耗

应用内存峰值

应用内存泄漏

应用是否常驻内存

压力测试后内存使用情况

### 1.4.内存需要知道的知识

基本内存耗用分类

* VSS- Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）
* RSS- Resident Set Size 实际使用物理内存（包含共享库占用的内存）
* PSS- Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）
* USS- Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）  
  说明：  
  a.一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS  
  b.实际在统计查看某个进程内存占用情况的时候，看 PSS 是比较客观的

### adb shell dumpsys gfxinfo com.yeqin.enterpriseservice > zinfo.txt1.5测试方法

#### 1.5.1三方工具

网易emmagee（不支持android7以上），腾讯gt（几年未更新源码），等

#### 1.5.2使用adb命令

* adb shell top 类似于linux top命令看到的是实时的数据变化（看的vss和rss）

Usage: top [ -m max\_procs ] [ -n iterations ] [ -d delay ] [ -s sort\_column ] [-t ] [ -h ]

-m num Maximum number of processes to display. 最多显示多少个进程

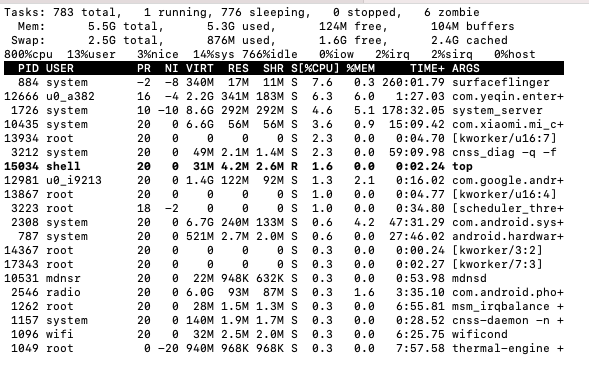
-n num Updates to show before exiting. 刷新次数

-d num Seconds to wait between updates. 刷新间隔时间（默认5秒）

-s col Column to sort by (cpu,vss,rss,thr). 按哪列排序

-t Show threads instead of processes. 显示线程信息而不是进程

-h Display this help screen. 显示帮助文档



重要几列含义：

**VIRT：virtual memory usage 虚拟内存**1、进程“需要的”虚拟内存大小，包括进程使用的库、代码、数据等  
2、假如进程申请100m的内存，但实际只使用了10m，那么它会增长100m，而不是实际的使用量

**RES：resident memory usage 常驻内存/实际使用内存**  
1、进程当前使用的内存大小，但不包括swap out  
2、包含其他进程的共享  
3、如果申请100m的内存，实际使用10m，它只增长10m，与VIRT相反  
4、关于库占用内存的情况，它只统计加载的库文件所占内存大小

**SHR：shared memory 共享内存**  
1、除了自身进程的共享内存，也包括其他进程的共享内存  
2、虽然进程只使用了几个共享库的函数，但它包含了整个共享库的大小  
3、计算某个进程所占的物理内存大小公式：RES – SHR  
4、swap out后，它将会降下来

S 进程状态:D=不可中断的睡眠状态, R=运行, S=睡眠, T=跟踪/停止, Z=僵尸进程

PID 进程id

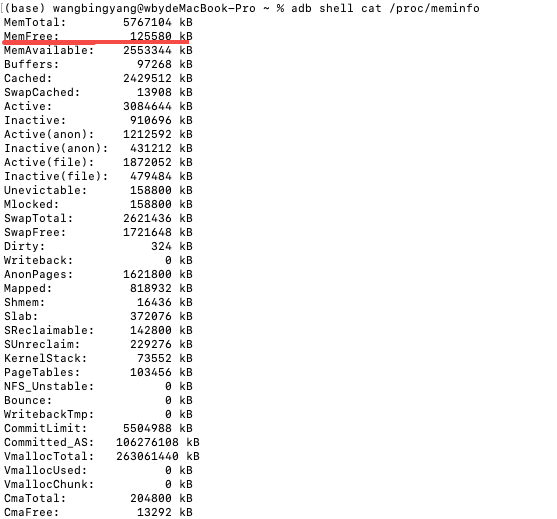
PR 优先级

CPU% 当前瞬时CPU占用率

可以在此基础上筛选出天眼企服的包的内存使用情况

adb shell top |grep com.yeqin.enter

* adb shell cat /proc/meminfo手机总内存



参数说明：

MemTotal: 所有可用RAM大小（即物理内存减去一些预留位和内核的二进制代码大小）

MemFree: LowFree与HighFree的总和，被系统留着未使用的内

Buffers: 用来给文件做缓冲大小

Cached: 被高速缓冲存储器（cache memory）用的内存的大小（等于 diskcache minus SwapCache ）.

SwapCached:被高速缓冲存储器（cache memory）用的交换空间的大小，已经被交换出来的内存，但仍然被存放在swapfile                    中。用来在需要的时候很快的被替换而不需要再次打开I/O端口。

Active: 在活跃使用中的缓冲或高速缓冲存储器页面文件的大小，除非非常必要否则不会被移作他用.

Inactive: 在不经常使用中的缓冲或高速缓冲存储器页面文件的大小，可能被用于其他途径.

HighTotal:

HighFree: 该区域不是直接映射到内核空间。内核必须使用不同的手法使用该段内存。

LowTotal:

LowFree: 低位可以达到高位内存一样的作用，而且它还能够被内核用来记录一些自己的数据结构。Among many

　　　　　other things, it is where everything from the Slab is allocated. Bad things happen when you’re out 　　　　　of lowmem.

SwapTotal: 交换空间的总大小

SwapFree: 未被使用交换空间的大小

Dirty: 等待被写回到磁盘的内存大小。

Writeback: 正在被写回到磁盘的内存大小。

AnonPages：未映射页的内存大小

Mapped: 设备和文件等映射的大小。

Slab: 内核数据结构缓存的大小，可以减少申请和释放内存带来的消耗。

SReclaimable:可收回Slab的大小

SUnreclaim：不可收回Slab的大小（SUnreclaim+SReclaimable＝Slab）

PageTables：管理内存分页页面的索引表的大小。

NFS\_Unstable:不稳定页表的大小

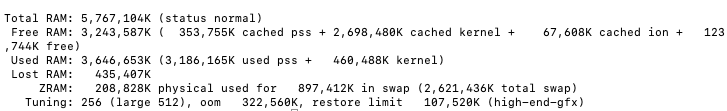
VmallocTotal: 可以vmalloc虚拟内存大小

VmallocUsed: 已经被使用的虚拟内存大小。

VmallocChunk: largest contigious block of vmalloc area which is free

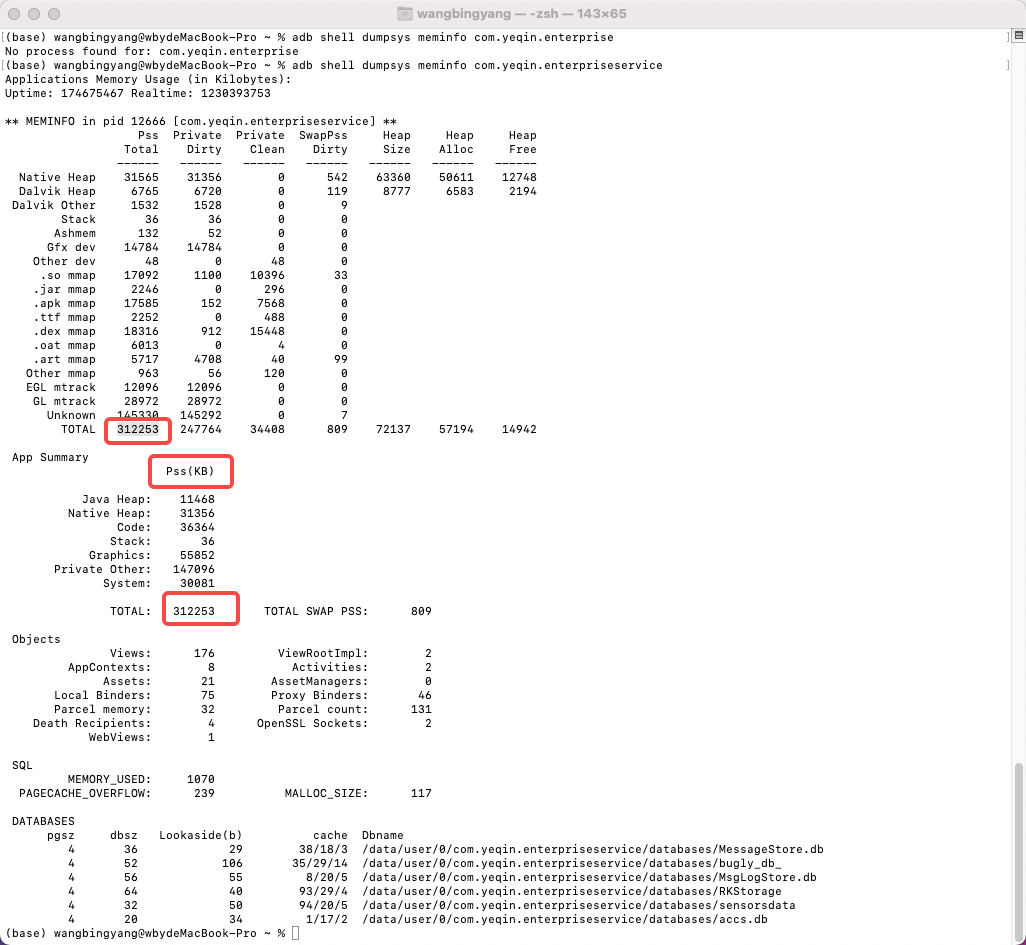
更详细的解释可以参考linux的：<https://blog.csdn.net/whbing1471/article/details/105468139/>

* adb shell dumpsys meminfo：统计的是pss



再次基础上筛选出天眼企服的占用内存

adb shell dumpsys meminfo com.yeqin.enterpriseservice



注意：dumpsys meminfo和cat /proc/meminfo两种方法得到的空闲内存是不一样的，通过手机助手看到的空闲内存是dumpsys meminfo统计的free ram

原因在于dumpsys meminfo获得的free mem = cached pss + cached kernel + free + ion cached + gpu cached（系统cached APP PSS总和 + meminfo的MemFree + Cached + Buffers-Mapped + meminfo的display ion和gpu模块的内存占用）。

所以，dumpsys meminfo是/proc/meminfo中的MemFree加上一些cache和buffer。这主要是因为，在linux中有这么一种思想，内存不用白不用，因此它尽可能的cache和buffer一些数据，以方便下次使用。但实际上这些内存也是可以立刻拿来使用的。所以空闲内存=free+buffers+cached。

#### 1.5.3****android检查内存泄漏步骤****

1）运行Monkey运行压力测试

adb shell monkey -p [包名] --pct-touch 100 --throttle 1000 -s 100 -v -v 50　　//运行APP，50次随机点击时间，每次时间间隔1s，并生成种子值100

参考：<https://mp.weixin.qq.com/s/JO1BNfXcN5j06-px5XPvPg>

2）监控内存值，如果出现过大等递增异常则保存HPROF文件（hprof文件是Java 虚拟机的Heap快照）用于分析查看应用内存的命令：

adb shell dumpsys meminfo cn.microinvestment.weitou(进程名)

如果发现内存过大，则保存HPROF文件：adb shell am dumpheap <进程名> <保存路径>

3）分析hprof文件

用工具MAT来查看，首先还要这个HPROF文件转换成MAT可读的文件

在Android SDK tool里面有个hprof-conv命令：

hprof-conv <原HPROF文件路径> <转换后的HPROF路径>

hprof-conv a.hprof b.hprof

4）用MAT工具打开转换后的HPROF文件

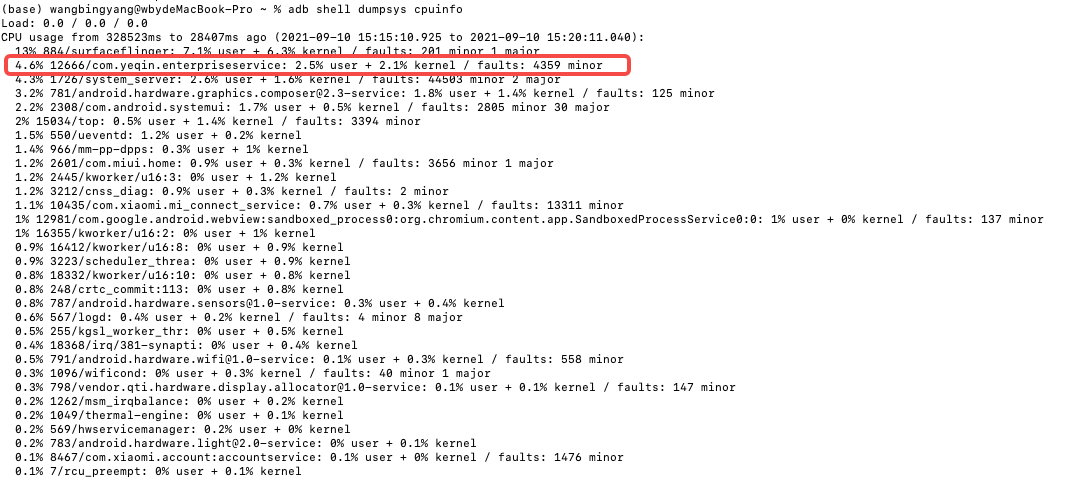
一般选择Leak Suspects Report（通过SQL语句来查询对象有没有被释放掉，如果有多个相同的对象，则会存在内存泄露的问题）

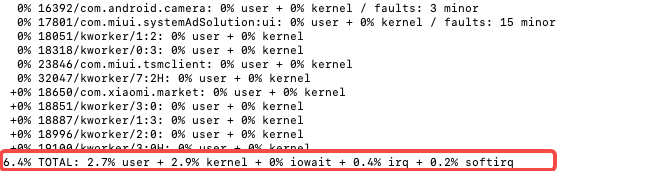
## 2 cpu

2.1 CPU测试中的测试子项  
1）空闲状态下的应用CPU消耗情况  
2）中等规格状态下的应用CPU消耗情况  
3）满规格状态下的应用CPU消耗情况  
4）应用CPU峰值情况

### 2.2 CPU数据获取：

(1)adb shell dumpsys cpuinfo





通过对上述命令进一步筛选，选出天眼企服的

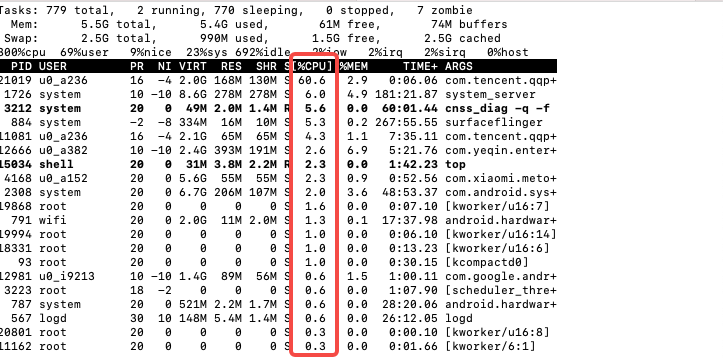
adb shell dumpsys cpuinfo |grep com.yeqin.enterpriseservice



2)top命令  
adb shell top -m 10 -s 9

#查看占用cpu最高的前10个程序（-t 显示进程名称，-n 在退出前刷新几次，-d 刷新间隔，-m 显示最大数量）-s 按指定排序按照数字排序，默认9

0 ,1 ,3 ,4 ,5 ,6 ,7 ,8,9 ,10 ,11 ,12 PID,USER,PR,NI,VIRT,RES,SHR,S,%CPU,%MEM,TIME+,CMDLINE



进一步筛选天眼企服的内存占用

adb shell top -n 1 |grep com.yeqin.enter



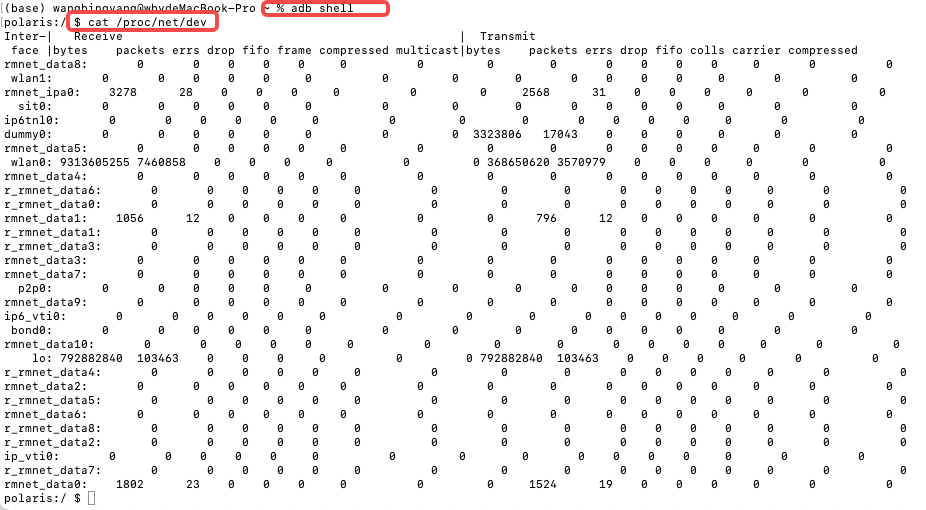
## 3流量

3.1、概念：  
中等负荷：应用正常操作  
高负荷：应用极限操作

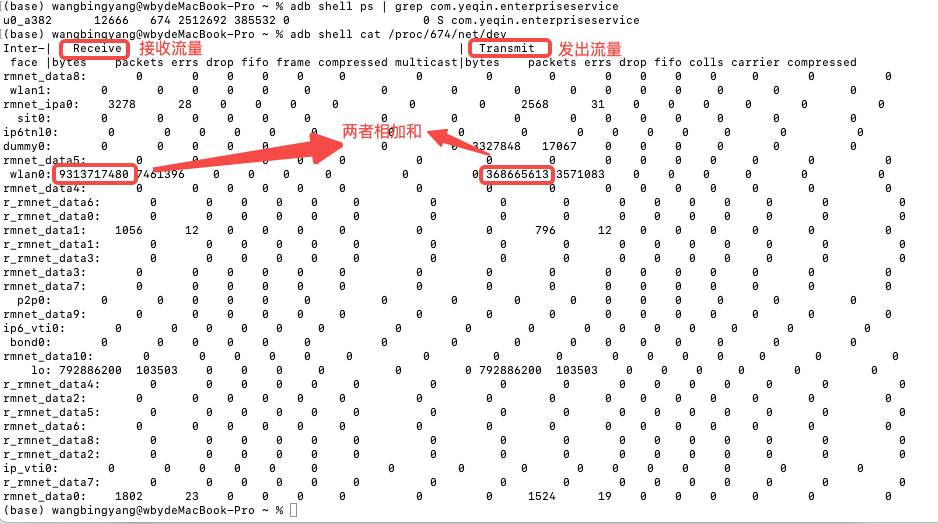
3.2、流量测试中的测试子项：  
1）应用首次启动流量值  
2）应用后台连续运行 2 小时的流量值  
3）应用高负荷运行的流量峰值  
4）应用中等负荷运行时的流量均值

3.3、获取流量数据：  
(1)获取系统流量

cat /proc/net/dev 获取系统的流量信息

  
（2）获取应用流量

第一步查询应用的pid： adb shell ps | grep tataufo #如：31002  
第二步通过PID获取该应用的流量数据： adb shell cat /proc/31002/net/dev  
（wlan0代表wifi上传下载量标识, 单位是字节可以/1024换算成KB, 打开手机飞行模式再关掉就可以将wlan0中的值初始化0）



1、receive是指当前进程接收的数据，transmit是指当前进程发出请求的数据，流量是这两者之和

2、wlan0代表WiFi，单位是字节，/1024可换算成KB

3、wlan0这些值如何初始化0 很简单 你打开手机飞行模式再关掉就清0了

#### 其他方式

4）查询应用的pid： adb shell ps | grep tataufo #如：31002  
　通过PID获取UID：adb shell cat /proc//status  
　通过UID获取：adb shell cat /proc/net/xt\_qtaguid/stats | grep 31002  
5）通过adb shell dumpsys package来获取应用的uid信息，然后在未操作应用之前，通过查看 ：  
　adb shell cat /proc/uid\_stat/uid/tcp\_rcv  
　adb shell cat /proc/uid\_stat/uid/tcp\_snd  
获取到应用的起始的接收及发送的流量，然后我们再操作应用，再次通过上述2条命令可以获取到应用的结束的接收及发送的流量，通过相减及得到应用的整体流量消耗

6）Android代码：Android的TrafficStats类

## 4 电量

### 4.1测试子项

a、测试手机安装目标APK前后待机功耗无明显差异。

b、常见使用场景中能够正常进入待机，待机电流在正常范围内。

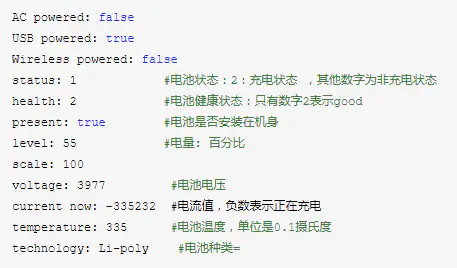
c、长时间连续使用应用无异常耗电现象。

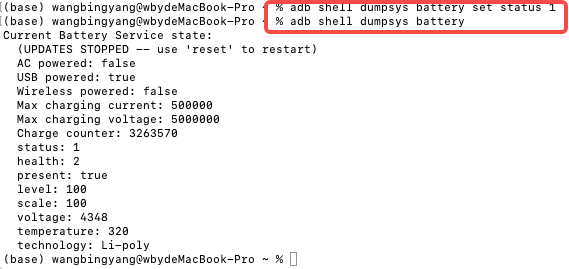
### 4.2命令

首先需要将手机切换到非充电状态，使用命令：adb shell dumpsys battery set status 1

(adb shell dumpsys battery set status 2：是将手机切回充电状态)

然后使用命令：adb shell dumpsys battery 获取手机电池信息

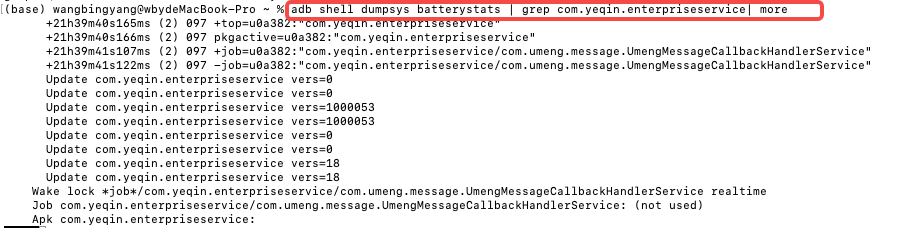




电量的消耗信息：

获取整个设备的电量消耗信息： adb shell dumpsys batterystats | more



获取某个apk的电量消耗信息： adb shell dumpsys batterystats | grep com.yeqin.enterpriseservice| more 

加上 >D:\xxx.txt 可将信息输出到文件中查看

使用命令  **python historian.py xxx.txt > xxx.html** 可将获得的数据转换为可视化的html文件。但之前需要先安装battery-historian工具，[具体方法见简友的这篇文章](https://www.jianshu.com/p/0fa57924f228" \t "_blank)

## 5.GPU

### ****5.1 概念****

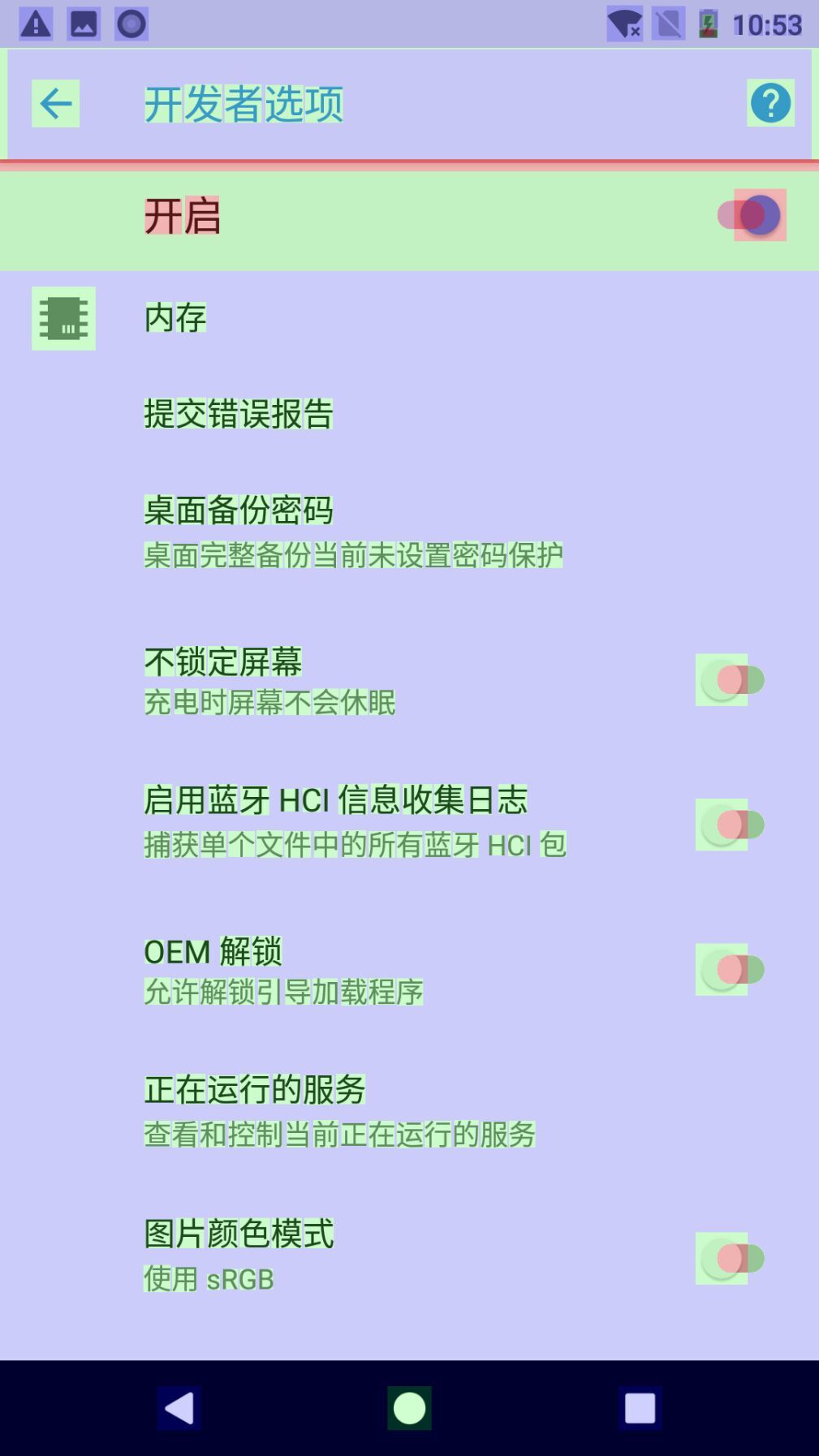
过度绘制：界面显示的activity套接了多层而导致  
帧率：屏幕刷新率

帧方差：屏幕滑动平滑度\*\*FPS：\*\*Frames Per Second 每秒显示的帧数 根据人眼的生理结构，帧率高于24时就被认为是连贯的。对于游戏画面30fps是最低能接受的，60fps逼真感，如果帧率高于屏幕刷新频率就是浪费。要达到30fps，每帧所占用的时间要小于33毫秒

5.2GPU测试中的测试子项：  
1）界面过度绘制  
2）屏幕滑动帧速率  
3）屏幕滑动平滑度

### ****5.3****过度渲染：****（人工进行测试）****

是指：屏幕中的某个像素在同一帧的时间内被绘制了多次  
打开开发者选项中的显示GPU过度绘制（Debug GPU overdraw）  
验收的标准:  
　1）不允许出现黑色像素  
　2）不允许存在4x过度绘制  
　3）不允许存在面积超过屏幕1/4区域的3x过度绘制（淡红色区域）



颜色越深代表渲染越过度

### ****5.4屏幕滑动帧速率测试：****

FPS（frames per second）：每秒的帧数（60是很流畅的一个状态）

#### 第一步

（1）进入手机系统设置，打开开发者选项，找到“GPU呈现模式分析”，选择任一模式都行。如果选择条形图，页面会出现如下画面：

  
那一条绿线表示FPS为60，超过这条线说明有些许卡顿

#### 第二步：启动SDK工具Systrace

此处参考网页：<https://blog.csdn.net/vicwudi/article/details/100191529>

如果是升级了系统以后，Android Device Monitor无法使用，可以使用通过python脚本抓取（推荐）

（1）装python2.X版本，Systrace脚本不支持3.X版本（macos自带python2）

（2）通过python脚本执行AndroidSDK\platform-tools\systrace\目录下的systrace.py文件

可以配置一些参数，类似于通过Android Device Monitor抓取时步骤2配置的显示信息，若不选择则默认全部抓取。

配置一些其他实用参数:

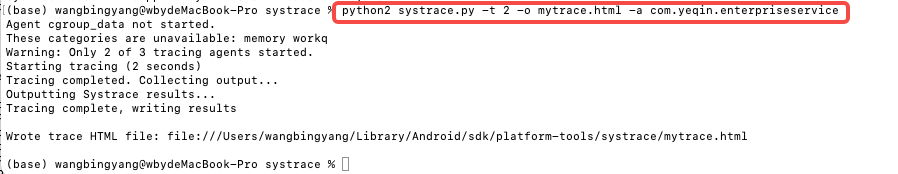
-o： 指定文件输出位置和文件名

-t： 抓取systrace的时间长度

-a： 指定特殊进程包名（自己加Label时必须加上）

抓取脚本示例：

|  |
| --- |
| python2 systrace.py -t 2 -o mytrace.html -a com.yeqin.enterpriseservice |



#### 第三步：

手工滑动界面可以通过节拍来进行滑动或者扫动，帧率数据会保存到刚刚设置的地址

#### 第四步：

通过浏览器打开保存的mytrace.html页面



对具体结果的分析也可以查看刚刚的那个网址

<https://pumpkin.blog.csdn.net/article/details/117289356>

<https://blog.csdn.net/vicwudi/article/details/100191529>

<https://blog.csdn.net/zhendong_hu/article/details/103858660>

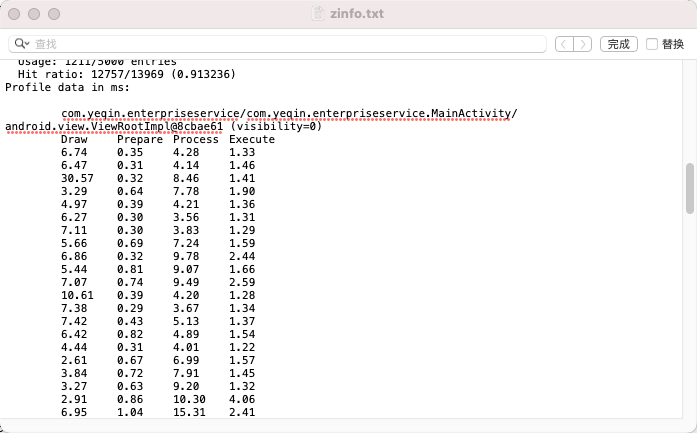
<https://www.jianshu.com/p/ab22238a9ab1>

5.5屏幕滑动平滑度的测试：  
方法如同帧率测试，唯一的差异就是最后的结果计算公式的差异

### ****5.6捕获app帧率（android流畅度FPS测试）：****

1）打开手机开发者选项，勾选GPU显示配置文件（系统会记录保留每个界面最后128帧图像绘制的相关时间信息）

  
2）adb shell dumpsys gfxinfo com.yeqin.enterpriseservice > zinfo.txt  
3）结果数据分析  
Profile data in ms部分：  
Draw： 创建显示列表的时间（DisplayList），所有View对象OnDraw方法占用的时间  
Process： Android 2D渲染引擎执行显示列表所花的时间，View越多时间越长  
Execute：将一帧图像交给合成器（compsitor）的时间，较小



## 6响应时间篇

**1、理解：**  
1）从单击事件触发到容器启动NativeAPP消耗的时间（埋点）  
2）NativeAPP完整启动消耗的时间（可以通过system.log获取）  
3）Native调用RPC请求方法的延迟时间（埋点）  
4）RPC请求发出去过程中的具体数据（req\_size req\_header req\_time等，通过埋点获取）  
5）RPC请求返回的具体数据（res\_size res\_header res\_time等，通过埋点获取）  
6）本地解析返回数据所消耗的时间（埋点或者TraceView工具可获取）  
7）界面渲染的时间（可以通过慢速摄像机或者埋点获取）

**2、android app启动时间测试**

（安卓Activity启动过程性能剖视: <http://www.rudy-yuan.net/archives/59/>）

**3、应用的启动时间的测试，分为三类：**  
　1）首次启动 --应用首次启动所花费的时间  
　2）非首次启动 --应用非首次启动所花费的时间  
　3）应用界面切换--应用界面内切换所花费的时间

**4、应用启动时间数据获取：**  
1）adb logcat > /address/logcat.txt #所有activity打印的日志  
find “Displayed” /address/logcat.txt > /newaddress/fl.txt #通过日志过滤关键字Displayed来过滤  
find “ActivityName” /newaddress/fl.txt > /newaddress/last.txt #通过activity名来过滤获取所测应用  
通过计算activity最后剩余的时间之和即可  
2）硬件测试， 使用高速相机或者手机采用录像的方法把应用启动过程给录制下来，然后通过人工数帧或者程序数帧的方式计算启动时间

## 七、弱网测试

**1、测试方法：**  
1）使用真实的SIM卡、运营商网络来进行测试（移动无线测试中存在一些特别的BUG必须在特定的真实的运营商网络下才会发现）  
2）通过代理的方式模拟弱网环境进行测试（charles 硬延迟）  
3）连接模拟弱网的热点进行测试

**2、热点模拟方法：**

1）通过设置iPhone的开发者模式之后共享热点（硬延迟）  
2）FaceBook开源的ATC（可使用树莓派来搭建ACT环境）

**3、用户体验需要做的：**  
1）在应用中统一弱网加载的界面样式、动画效果、菊花icon等  
2）统一网络错误、服务端错误、超时等展现给用户的界面和提示语句  
3）定义清楚在每个中间过程是的用户交互行为

4、Chales配置

Proxy->Throttling Settings

☑️勾选［Enable Throttling］使的限制网速可用，相当于开启了限制网速的功能

配置指标：

弱网、2G、3G建议的上下行速率如下，同时还可以控制丢包率的数据

网络 上行 下行

弱网 10 30

2G 15 50

3G 384 2800

## 2 性能工具推荐

（1）android studio：

之前使用的android device monitor在android studio3以后淘汰掉了，开始使用profiler做性能，profiler这个工具模拟器可以直接使用，真机需要root以后才能看到安装的应用，如果不想root需要有天眼企服app的源码，使用android studio编译才能测试企服app的性能，不过开启profiler以后企服app点击一会容易闪退～

（2）Mobileperf：目前发现的比较强大的开源android性能工具，缺点未可视化

<https://cloud.tencent.com/developer/article/1801395>

<https://testerhome.com/topics/23449>

<https://github.com/alibaba/mobileperf>

（3）perdog：性能狗，傻瓜式安装使用，结果图形可视化，做性能很方便但是9.1以后开始收费了，最便宜一年3k只能录制3000min

（4）Emmagee：很多年前的工具，不支持android7

（5）gt：<https://github.com/Tencent/GT>

android gt可以直接安装在手机上，之后在手机上查看各个数值

ios需要在app源码中把gt的源码打进去

（6）itest: <https://www.cnblogs.com/fanf/p/12419882.html>

需要注册账号

|  |  |
| --- | --- |
| 测试工具 | 测试指标 |
| leakcanary | 内存泄漏 |
| GT Home | CPU、内存、流量、点亮、帧率/流畅度、查看开发日志、crash 日志、抓取网络数据包、APP 内部参数调试、真机代码耗时统计 |
| iTest | cpu、内存、流量、电量等 |
| Emmagee | cpu、内存、流量、电量等 |
| PerfDog | cpu、内存、流量、电量等 |

# 二ios的app性能测试

## 1.途径

### ****2.1.1Instruments****

Instruments 是 Xcode 自带的，用来动态跟踪和分析 MacOS 和 iOS 代码的实用工具，以独立 APP 形式存在的工具集，包含了很多强大的检测功能：其中包括在真机和模拟器上的 APP 进行性能测试。

主要功能有：

* 分析程序进程行为
* 分析内存分配及释放细节
* 查找 CPU 问题
* 查找程序性能瓶颈
* 检查设备相关的功能，比如：Wi-Fi、蓝牙等。
* 查找 App 中的内存问题，比如内存泄露(Leaked memory)、废弃内存(Abandoned memory)、僵尸(zombies)等。

具体查看官方文档： [https://help.apple.com/instruments/mac/current/#/dev7b09c84f5](http://testingpai.com/forward?goto=https%3A%2F%2Fhelp.apple.com%2Finstruments%2Fmac%2Fcurrent%2F%23%2Fdev7b09c84f5)

### 2.1.2第三方 SDK

很多第三方的专门用于性能检测和用户行为、属性分析的 SDK，比如 **Bugly，OneAPM，听云，Firebase Analytics**，把它们接入项目可以短期内达成性能检测目标。

这些第三方的工具原理都是类似的，利用 swizzle 的方法进行 AOP（面向切面编程）处理，在关键函数之前和之后自动埋点记录上报。有的平台也支持上传符号表文件精确定位代码执行位置以及以埋点的方式手工添加日志记录。使用起来还是比较方便的，基本上引入 SDK 和相关库，在程序入口处启动检测即可。

然而使用第三方 SDK 的缺点也是非常明显的：

* 首先是缺乏定制性，我们需要的一些指标的统计 SDK 没有，SDK 有的我们又不完全需要。
* 存在安全性问题，这些 SDK 涉及的统计数据都是 APP 的商业机密信息，对于有一定市场影响力的 APP 会有顾忌。
* 这类产品是收费的，平时自己开发个 demo 练手也不适合连这种 SDK。

### 2.1.3三方工具

Moonlight：需要自己写性能测试的代码，引入moonlight 的api接口

<https://github.com/AgoraIO-Community/MoonLight/blob/master/README.zh.md>

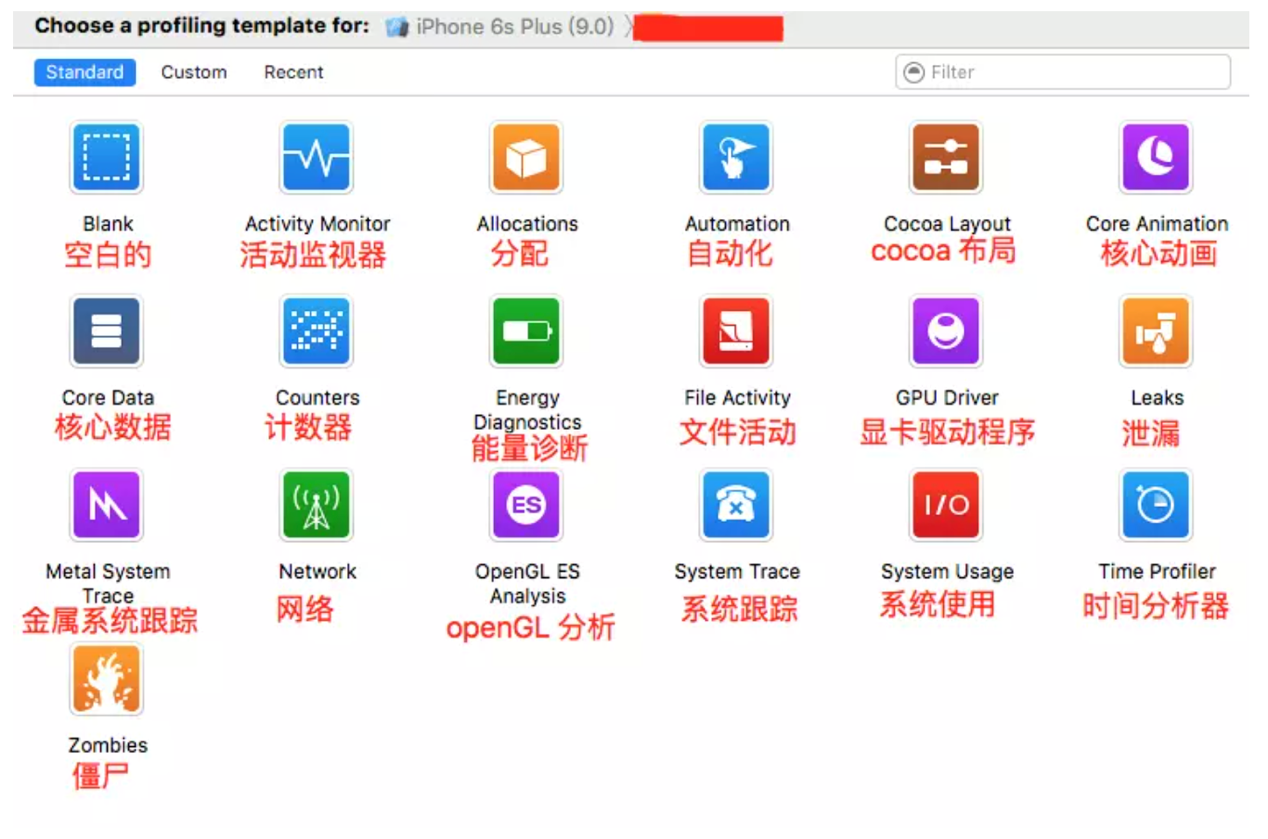
matrix：需要自己写性能测试的代码，引入moonlight 的api接口

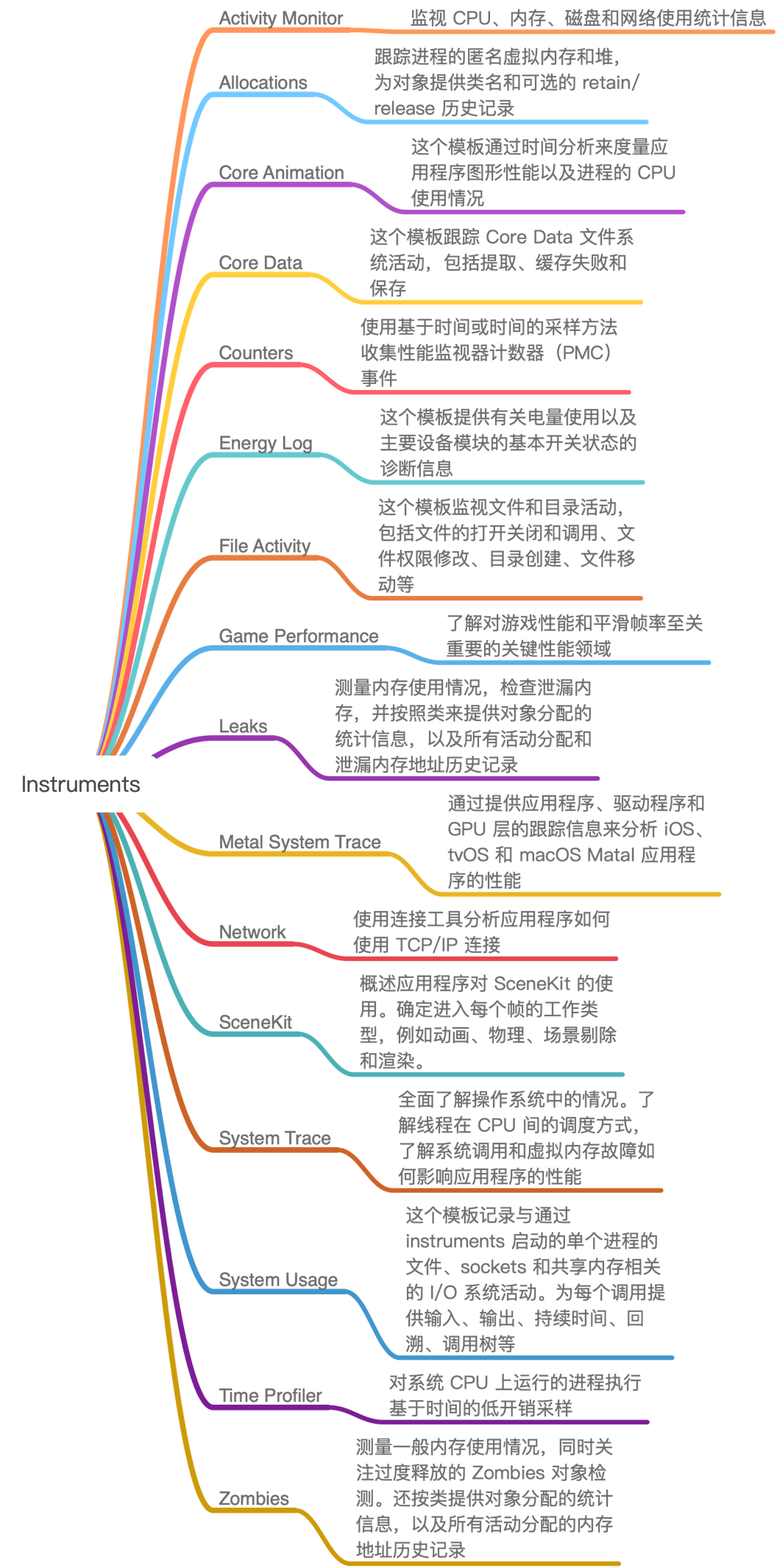
<https://github.com/tencent/matrix#matrix_ios_cn>

没有找到命令获得性能数据，目前找到的相对容易点的方法是使用xcode instrument，缺点是需要源码编译

## 2 xcode性能工具instrument简单介绍

打开方式Xcode -> Open Developer Tool -> Instruments





2、常用模块

* Allocations：用来检查内存分配，跟踪过程的匿名虚拟内存和堆的对象提供类名和可选保留/释放历史
* Leaks：一般的查看内存使用情况，检查泄漏的内存，并提供了所有活动的分配和泄漏模块的类对象分配统计信息以及内存地址历史记录
* Time Profiler：分析代码的执行时间，执行对系统的CPU上运行的进程低负载时间为基础采样
* Zombies：检查是否访问了僵尸对象

详细介绍：

<https://www.jianshu.com/p/1dde55c20303>

<https://www.cnblogs.com/R-bear/p/15024719.html>