

专项测试

推荐：

<https://item.jd.com/11976603.html>

本章大纲

- 移动应用的测试范围

- 服务端性能测试

- APP的启动时间

- CPU测试

- 电量测试

- 流量测试

- 静态扫描工具

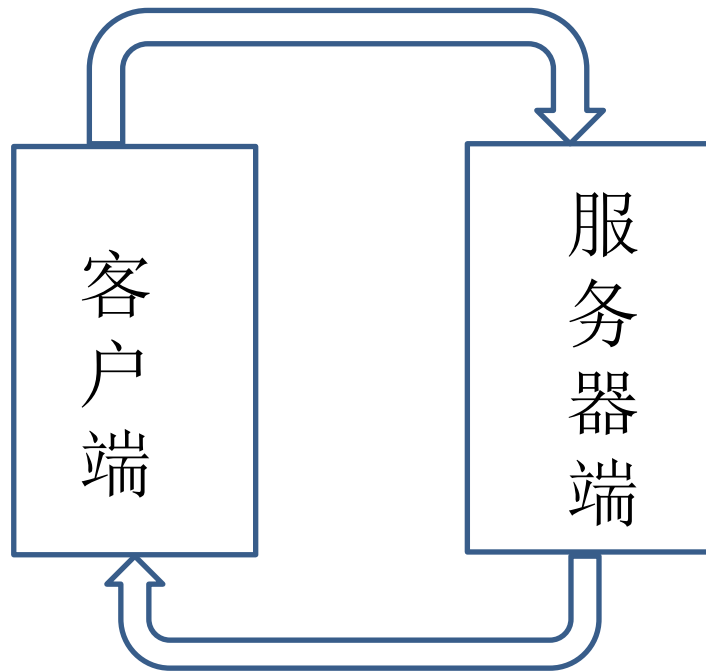
- Traceview

- 内存测试

- 其他工具

移动应用测试

- 移动应用到底测试什么？



本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

服务端性能测试

服务端性能测试的工具：

➤ LoadRunner

➤ JMeter

➤ 自主研发的工具

服务端性能测试

服务端性能测试的指标

➤ CPU

➤ 内存（虚存、实存）

➤ QPS、平均响应时间

服务端性能测试

服务端性能测试的方法

1. 搭建服务器模块，启动服务
2. 监控进程相关指标，CPU、内存（nmon）
3. 监控模块的执行情况，QPS、平均响应时间
4. 收集数据并进行分析，生成曲线图
5. 根据分析结果，得出测试结论

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

APP启动时间测试的方法

Android中APP的启动方式有两种状态，主要分为冷启动和热启动。

冷启动：当启动应用时，后台没有该应用的进程，这时系统会重新创建一个新的进程分配给该应用，这个启动方式就是冷启动。

冷启动因为系统会重新创建一个新的进程分配给它，所以会先创建和初始化Application类，再创建和初始化MainActivity类（包括一系列的测量、布局、绘制），最后显示在界面上。

APP启动时间测试的方法

热启动： 当启动应用时，**后台已有该应用的进程**（例：按back键、home键，应用虽然会退出，但是该应用的进程是依然会保留在后台，可进入任务列表查看），所以在已有进程的情况下，这种启动会从已有的进程中来启动应用，这个方式叫热启动。热启动因为会从已有的进程中来启动，所以热启动就不会走Application这步了，而是直接走MainActivity（包括一系列的测量、布局、绘制），所以热启动的过程只需要创建和初始化一个MainActivity就行了，而不必创建和初始化Application，因为一个应用从新进程的创建到进程的销毁，Application只会初始化一次。

APP启动时间测试的方法

1、代码插入时间打印Log.e

2、命令方式

```
adb shell
```

```
am start -W -n
```

```
com.example.todolist/.LoginActivity
```

-W 启动完成之后，返回启动时间

3、秒表

4、云测平台

APP启动时间测试的方法

5、adb logcat

adb logcat >d:\demo\app_time.txt

启动应用，加载完成后ctrl+c停止

findstr "Displayed"

d:\demo\app_time.txt>d:\demo\app_time1.txt

findstr "com.example.todolist"

d:\demo\app_time1.txt>d:\demo\app_time2.txt

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

CPU测试

1、dumppsys命令

adb shell dumppsys cpuinfo | grep 包名

2、top命令

adb shell top | grep 包名

3、第三方工具、云测平台

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

电量测试

电量测试，就是测试移动设备电量消耗快慢的一种测试方法。一般是用平均电流（电池生产厂家一般都采用mAh来标记电池容量大小，平均电流越小，说明设备使用时间就越长）来衡量电量消耗速度。

电量测试

常用的电量测试方法有以下两种：

1、硬件测试

传统的硬件测试法就是利用电量测试仪测试被测设备（拆除自带电池，使用同型号假电池）的电流，统计一段时间内的平均电流值（都有PC上的配套软件，有些可通过图表输出。如果实在觉得抽象，可以想象下心电图）。

电量测试

2、耗电检测APP

通过第三方软件和程序，模拟计算应用APP的耗电，这种方法一般用来分析APP耗电，不作为平均电量的基准值。

腾讯的GT和Instrument来分别测试Android和iOS的电量。

影响电量的因素

几个典型的耗电场景如下：

- 1) 定位，尤其是调用GPS定位。
- 2) 网络传输，尤其是非Wifi环境。
- 3) cpu频率
- 4) 内存调度频度
- 5) wake_locker时间和次数

<https://sq.163yun.com/blog/article/195983166612123648>

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

流量测试

流量可以从用户使用的相关性角度分为两类，一类是用户的操作直接导致的流量消耗，另一类是后台，即在用户没有直接使用情况下的流量消耗。

后一种情况更加容易出现，因为Android的**消息推送机制**不是借助统一的管道，而是各个app定时启动后台进程到自己的服务端去询问是否有新消息，有就拉去到客户端，而这个询问本身就会带来流量的消耗。

流量测试

方法1：系统API的TrafficStats类来获取基本的流量数据

方法2：手机抓包

wireshark和linux下的tcpdump

方法3：通过网络代理来统计

Windows的fiddler、ios的charles

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

代码静态扫描

Android Lint 是 SDK Tools 16 (ADT 16) 开始引入的一个代码扫描工具，通过对代码进行静态分析，可以帮助开发者发现代码质量问题和提出一些改进建议。除了检查 Android 项目源码中潜在的错误，对于代码的**正确性、安全性、性能、易用性、便利性和国际化**方面也会作出检查。

代码静态扫描

Accessibility 无障碍，例如 ImageView 缺少contentDescription 描述，String 编码字符串等问题

Correctness 正确性

Internationalization 国际化，如字符缺少翻译等问题 **Performance 性能**，例如在 onMeasure、onDraw 中执行 new，内存泄露，产生了冗余的资源，xml 结构冗余等。

Security 安全性，例如没有使用 HTTPS 连接 Gradle，AndroidManifest 中的权限问题等

Usability 易用性，例如缺少某些倍数的切图，重复图标等

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

TraceView

Traceview是android平台配备的一个很好的性能分析工具。它可以通过图形界面的方式让我们了解我们要跟踪的程序的性能，并且能具体到method。

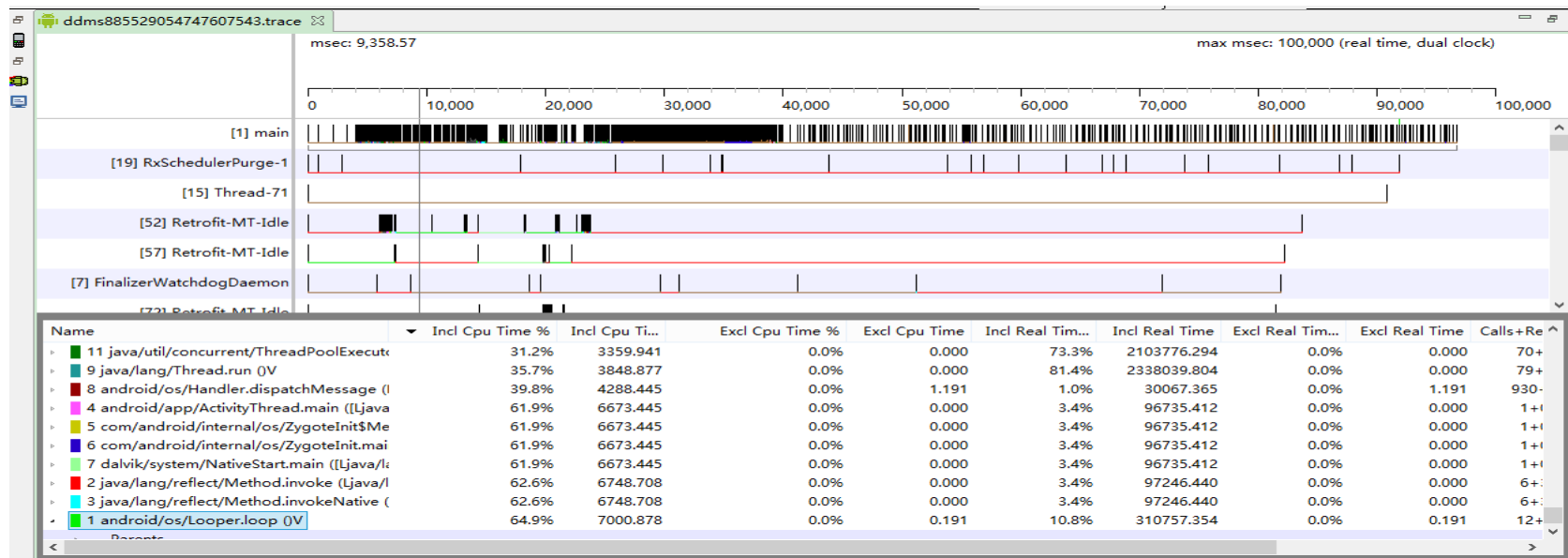
使用TraceView的方法

1. 最简单的方式就是直接打开DDMS，选择一个进程，然后按上面的“Start Method Profiling”按钮，等红色小点变成黑色以后就表示TraceView已经开始工作了。然后可以滑动一下列表。操作最好不要超过5s，因为最好是进行小范围的性能测试。然后再按一下刚才按的按钮，等一会就会出现上面这幅图，然后就可以开始分析了。

适合于检测某一个操作的性能

2. 开发人员在使用关键代码前使用`android.os.Debug.startMethodTracing()` ;和`android.os.Debug.stopMethodTracing()` ;方法，当运行了这段代码的时候，就会有一个trace文件在/sdcard目录中生成，也可以调用`startMethodTracing(String traceName)` 设置trace文件的文件名，最后你可以使用`adb pull /sdcard/test.trace /tmp` 命令将trace文件复制到你的电脑中，然后用DDMS工具打开就会出现第一幅图了

适合检测某一个方法的性能



上面是测试的进程中每个线程的执行情况，每个线程占一行；
下面是每个方法执行的各个指标的值

列名	含义
Name	该线程运行过程中所调用的函数名称
Incl Cpu Time	某函数所占用的CPU时间，包含内部调用其他函数所占用的时间
Excl Cpu Time	某函数所占用的CPU时间，不包含内部调用其他函数所占用的时间
Incl Real Time	某函数运行的真实时间，包含内部调用其他函数所占用的真实时间（毫秒）
Excl RealTime	某函数运行的真实时间，不包含内部调用其他函数所占用的真实时间
Calls + Recur Calls / Total	某函数被调用的次数以及递归调用次数的百分比
Cpu Time / Call	某函数调用CPU时间与调用次数的比值，相当于该函数平均执行时间
Real Time / Call	某函数实际运行时间

	▲ Incl Cpu Time %	Incl Cpu Ti...	Excl Cpu Time %	Excl Cpu Time	Incl Real Tim...	Incl Real Time	Excl Real Tim...	Excl Real Time	Calls+RecurCal
17 android/view/Choreographer.doFrame	17.3%	1869.882	0.0%	0.000	0.3%	7996.551	0.0%	0.000	802+0
Parents									
16 android/view/Choreographer\$Fi	100.0%	1869.882			100.0%	7996.551			802/802
Children									
self	0.0%	0.000			0.0%	0.000			
18 android/view/Choreographer.dc	99.7%	1863.852			99.8%	7979.288			800/800
356 java/lang/System.nanoTime ()J	0.2%	3.482			0.2%	12.011			4/71
2639 android/util/Log.i (Ljava/lang/	0.1%	2.548			0.1%	5.252			2/2

可以找出重复调用的问题

本章大纲

- 移动应用的测试范围
- 服务端性能测试
- APP的启动时间
- CPU测试
- 电量测试
- 流量测试
- 静态扫描工具
- Traceview
- 其他工具

Emmagee

Emmagee是网易杭州研究院QA团队开发的一个简单易上手的Android性能监测小工具，主要用于监控单个App的CPU，内存，流量，启动耗时，电量，电流等性能状态的变化，且用户可自定义配置监控的频率以及性能的实时显示，并最终生成一份性能统计文件。
<https://github.com/NetEase/Emmagee/releases>

GT

GT（随身调）是APP的随身调测平台，它是直接运行在手机上的“集成调测环境”（IDTE, Integrated Debug Environment）。利用GT，仅凭一部手机，无需连接电脑，即可对APP进行快速的性能测试（CPU、内存、流量、电量、帧率/流畅度等等）、开发日志的查看、Crash日志查看、网络数据包的抓取、APP内部参数的调试、真机代码耗时统计。

<https://github.com/Tencent/GT>

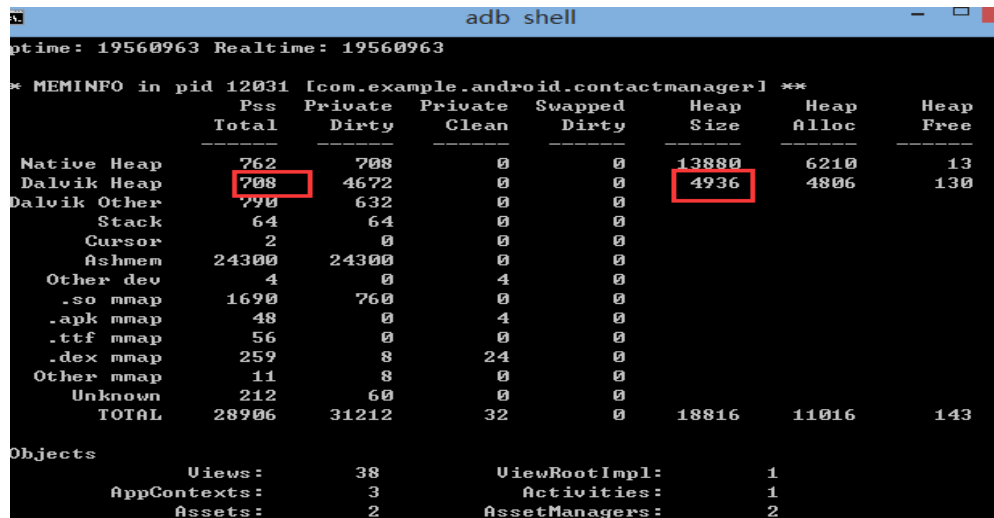
<https://gt.qq.com/>

内存机制简介

在java开发过程中，是通过new 来为对象分配内存，而内存的释放是由垃圾收集器GC来回收的，在开发过程中，不需要显示的去管理内存， jvm会帮助我们回收内存。但是这样有可能在不知不觉中浪费了很多内存，最终导致jvm花费很多时间去进行垃圾回收，更严重的是造成OOM。

如何查看占用的内存情况

- 显示当前时刻的内存：adb shell dumpsys meminfo pakagename, 重点关注两个指标
- DDMS



```
adb shell
otime: 19560963 Realtime: 19560963

* MEMINFO in pid 12031 [com.example.android.contactmanager] **

      Pss   Private Dirty   Private Clean   Swapped Dirty   Heap Size   Heap Alloc   Heap Free
-----
Native Heap      762        708           0           0       13880        6210         13
Dalvik Heap     708        4672           0           0       4936        4806        130
Dalvik Other     790        632           0           0
Stack           64         64           0           0
Cursor           2           0           0           0
Ashmem      24300      24300           0           0
Other dev         4           0           4           0
.so mmap       1690        760           0           0
.apk mmap        48           0           4           0
.ttf mmap        56           0           0           0
.dex mmap       259           8          24           0
Other mmap        11           8           0           0
Unknown        212          60           0           0
TOTAL      28906      31212           32           0      18816      11016         143

Objects
      Views:          38      ViewRootImpl:          1
AppContexts:          3      Activities:          1
      Assets:          2      AssetManagers:        2
```

App的调试

调试App相关的bug常用哪些工具？

1. 网络相关工具：fiddler、wreshark、Charles
2. adb、am、logcat等
3. 云平台、bug复现