# 知乎移动端性能测试实践

陈康康

在移动客户端的测试环节中，性能测试是比较重要的一部分，相对于普通的常规测试，性能测试实施复杂、环境配置困难、测试问题场景很难复现、报告编写和结果分析对测试人员要求较高等等因素导致性能测试很难在实际的测试过程中推行。

实际的测试过程中我们经常会遇见通讯异常、客户端停止运行，还有各种崩溃、卡死、页面加载一直转圈、手机运行过热、耗电严重等等问题，如下是几款典型的性能测试工具对比：



可以看出上述工具的实施难度都比较高，并且在一定程度上都会依赖本地开发环境配置，在跨平台方面常规第三方工具必须添加低侵入式 SDK，生成报告又需要手动获取文件，整理文件数据手动生成各类报表等。所以考虑到性能测试的便捷性、低成本等方面的要求，尽可能在测试阶段解决上述这一类问题，于是设计了一个便捷、无门槛的性能测试工具，我会从工具的使用步骤、设计目标、问题发现、工具实现细节、自动化接入、工具扩展等方面来进行介绍。

**使用步骤**

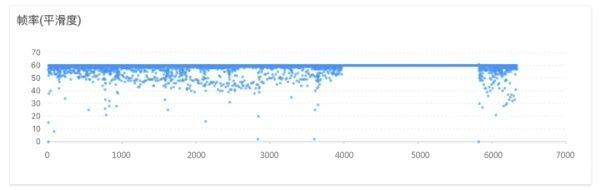
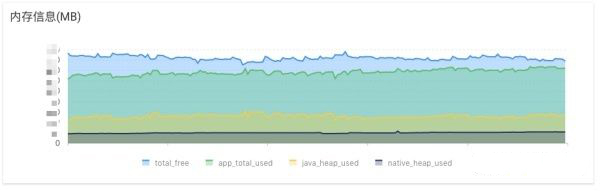
首先简单介绍一下工具的主要步骤如下：

1.打开工具页面

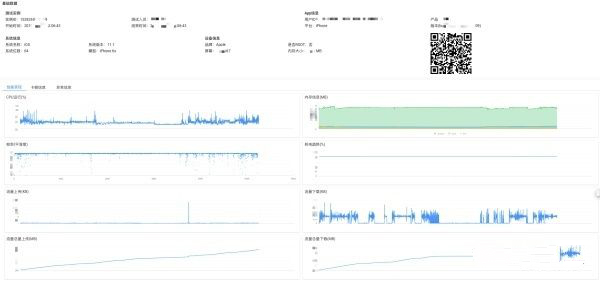


2.APP 扫码  
使用 APP 内置的扫码工具扫描二维码，使 APP 内置的地址修改为测试服务器的地址，这时工具页面会自动跳转至性能数据实时查看页面。

3.APP 实时性能表现查看

CPU/MEM流量上传下载Java GC帧率内存占用

4.测试报告生成和留存



完成测试之后系统会直接生成测试报告，开发、产品、测试可以用来共同分析测试过程中遇到的问题。

**系统设计目标**

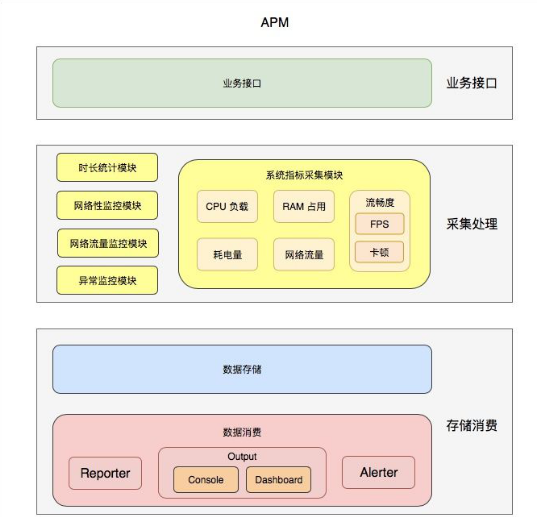
1. 环境配置、易用性  
   在环境配置方面、基本上不需要再依赖本地环境，由客户端内置代码和服务端通讯完成数据上传，完整的测试过程一共只有 4 个步骤即可，测试门槛几乎基本上全部消失、通过扫码 → 报告的方式，最大程度简化了使用流程，带来了极大的便捷性和易用性。
2. 数据展示  
   1) 基础数据展示  
   客户端第一次上传数据时，系统会自动识别客户端的基础信息、包括平台、客户端版本、设备型号、测试人员、测试时间等相关信息。  
   2) 性能数据展示  
   数据展示方面使用了 bizcharts 对数据进行可视化、分别采用区域图、线图、点阵图展示最近 5 - 10 分钟的性能数据，在测试期间、系统会动态定时刷新性能数据。
3. 结果分析  
   在上面的图表中，可以看到‘流量下载’部分的图表中存在某一区域明显上升，在通常的操作过程中这种流量下载的突然增高可能会是用户使用时进行了下载、语言、视频、音乐等操作，在测试过程中实际上并没有上述操作，可以判断是由一些 API 的数据请求造成，复现几次后可以得到实际的请求 API，然后由开发人员判断是否需要针对这个 API 进行瘦身优化，减少请求流量的消耗。
4. 质量报告  
   质量报告的生成没有时间限制，也不需要进行特殊处理，在测试人员完成性能测试、或者手动结束性能测试之后，系统会自动生成性能测试结果报告的页面，并生成唯一的测试报告链接。
5. 实施难度和扩展  
   在设计开发和测试实践过程中，遇到的主要难度是：  
   客户端数据收集上传的过程中、需要保证数据的完整，保证数据的连贯性。所以客户端数据上传时采用队列的方式存储数据，当系统出现问题、网络异常、服务器故障时把数据存入队列，在问题解决之后再将队列中的数据发送到服务端。

**实现过程和细节**

性能数据获取的前提是质量团队和客户端团队对指标的获取达成共识，主要方式由客户端实现后质量团队 review 代码确认后确定数据获取方式，在数据获取方面没有做更加深入的自定义，包括一些指标如下：

1. CPU、内存、网络流量、帧率
2. Android GC 数据
3. 电池、电量相关数据
4. 异常、卡顿、自定义代码运行问题

客户端架构设计：



客户端采取队列方式上传数据，提供可配置的间隔控制和开关。

服务端数据交互  
数据采用 base64 加密后，传输 gzip 压缩，服务接收数据解析

@Throws**(**IOException**::**class**)**

**private** fun **preProcess(**request**:** HttpServletRequest**):** ByteArray **{**

*// 获取输入流*

*// 转 GZIP 输入流*

*// 获取解密方式 base64 解密*

*// 捕获异常并返回字节流数据*

**}**

**自动化性能测试接入**

1. 通过自动化接口调用服务端生成性能测试实例



2. 客户端自动化打开开关进行自动化性能测试

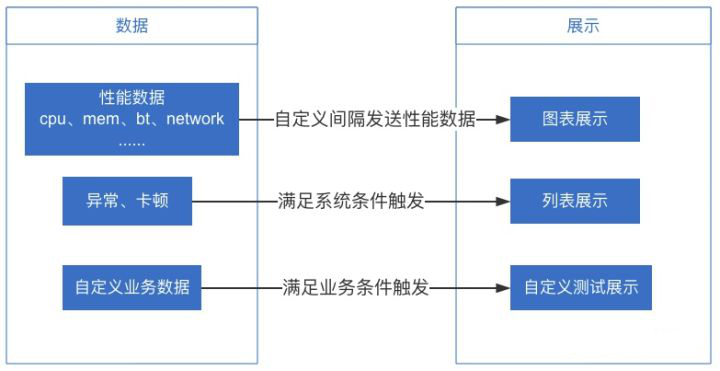
客户端通过自动化脚本打开性能测试开关、同时调整性能测试服务器 url 的指向，然后执行编写完成的自动化性能测试脚本，同时上传数据并记录时间戳，当发现问题时用于核对性能数据和操作步骤。  
3. 调用性能测试结束接口

当自动化性能脚本反复执行完毕之后，达到 2 小时的标准建议或者手动调用测试结束接口，系统会结束测试不再接收后面的数据，生成测试报告同时按照规则生成报告的唯一 url 地址。  
4. 自动获取生成的性能测试报告

自动化性能测试完成之后，会拿到性能的报告地址，可以通过邮件、IM 等方式将报告发送给相关人员或进行下一步的其他测试。

**扩展**

系统还在建设过程中，后续的扩展主要基于下图：



在客户端、浏览器、测试服务器之间建设一个性能测试闭环，同时在后面的性能测试过程中还会扩展关注如下一些指标：

1.卡顿异常（卡顿堆栈信息、异常定位数据）

2.过程耗时（冷启、热启、重要业务耗时）

3.页面切换、加载时间

4.内存泄漏、抖动、页面渲染（FPS）、图片/视频/语音等专项指标