

移动应用性能瓶颈 与 解决方案

张文欣



1

移动应用性能测试指标

2

常见维度的采集方法

3

低侵入的数据采集方式

4

OneAPM性能测试数据分析

移动应用性能指标



启动时间



内存



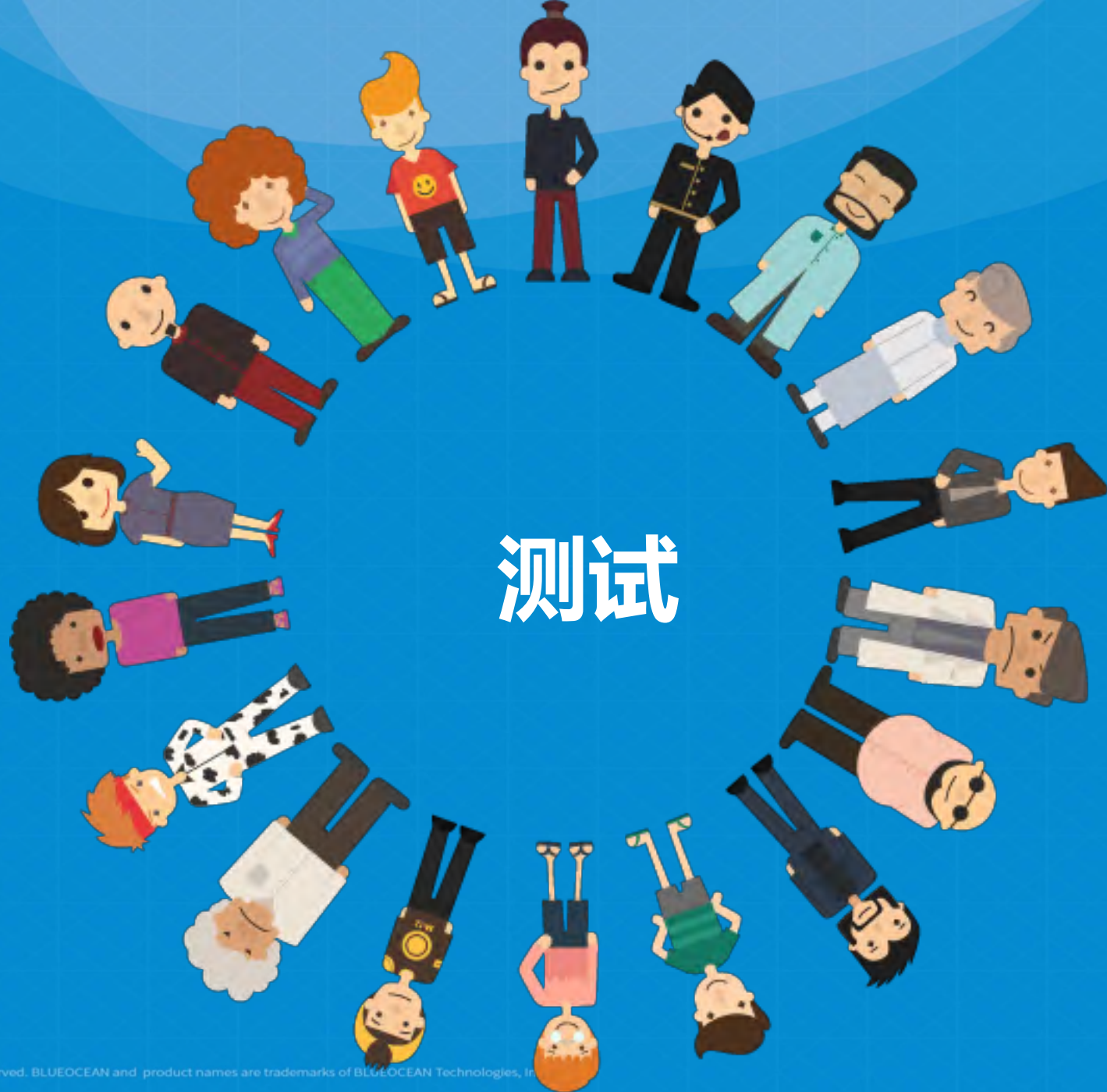
CPU



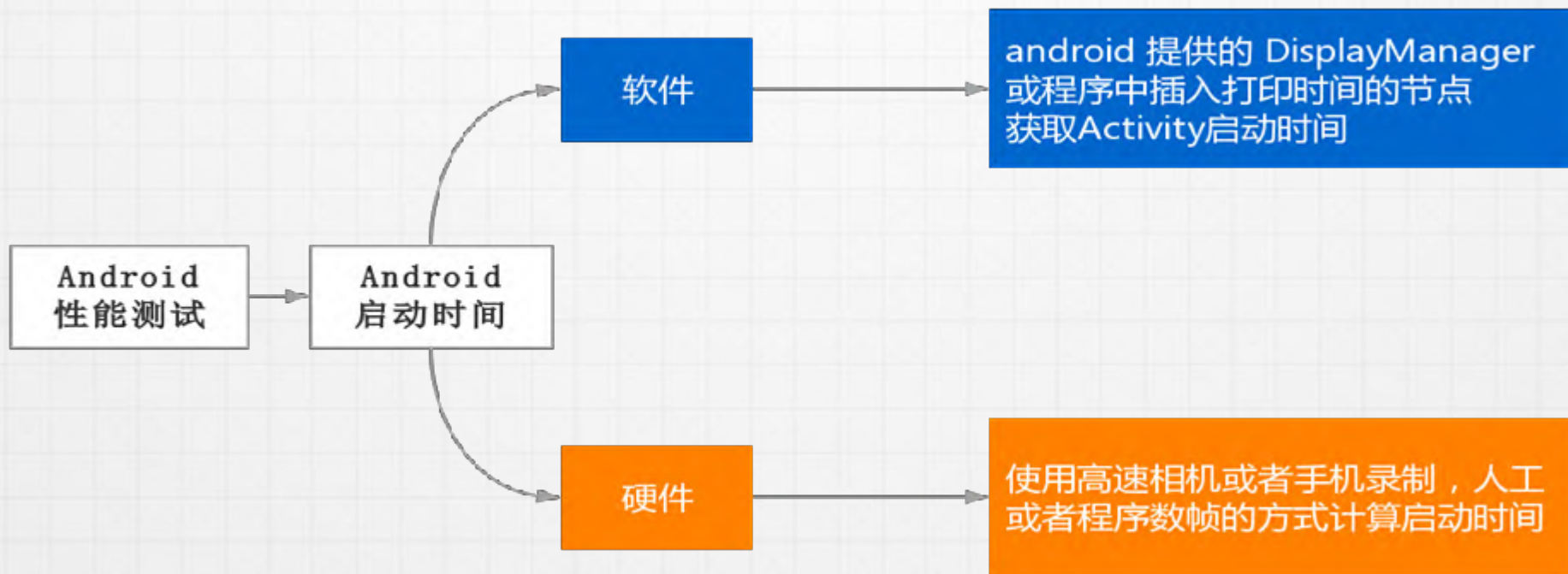
流量

如何获取这些性能数据并对其进行分析？

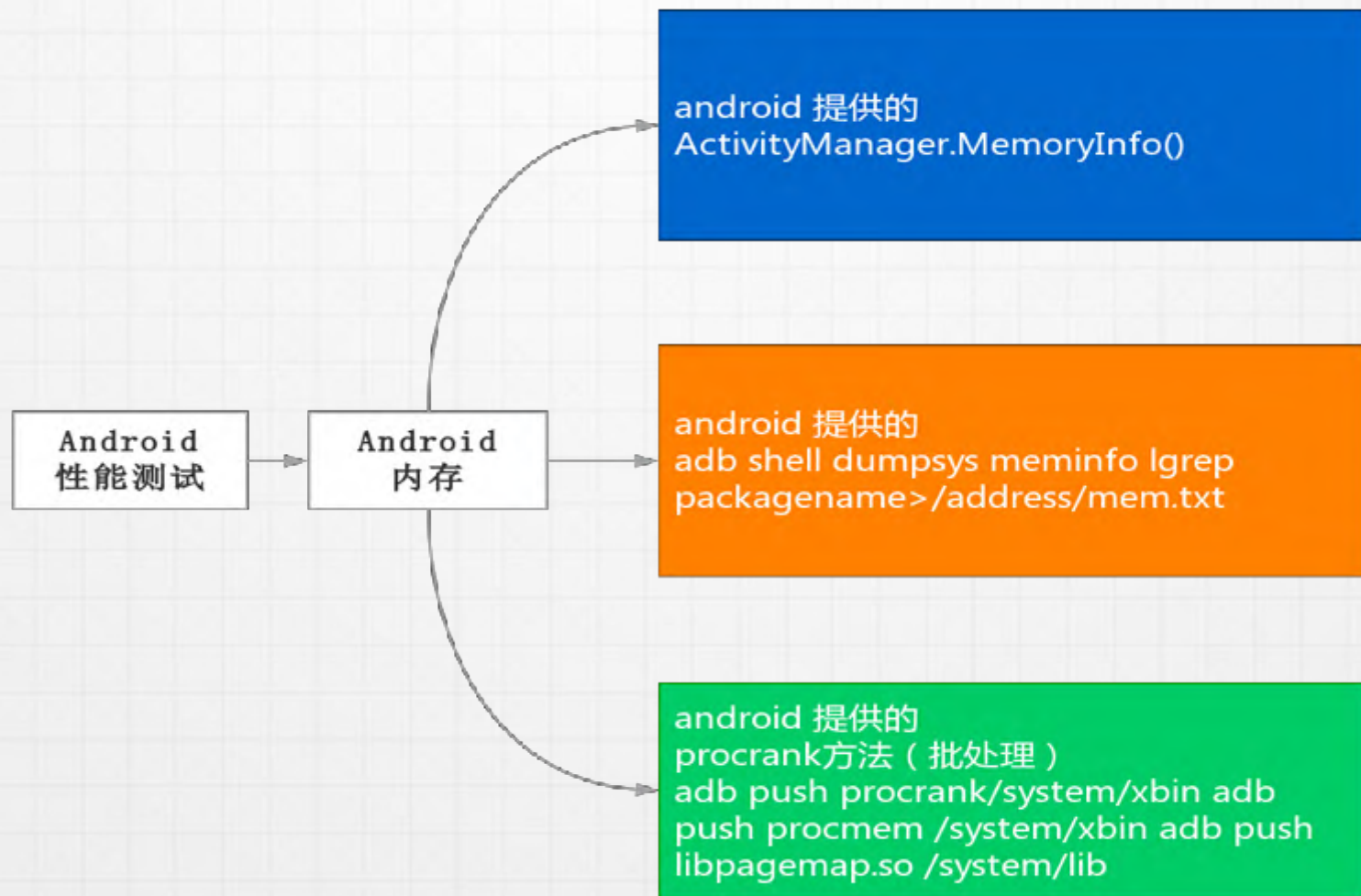
测试



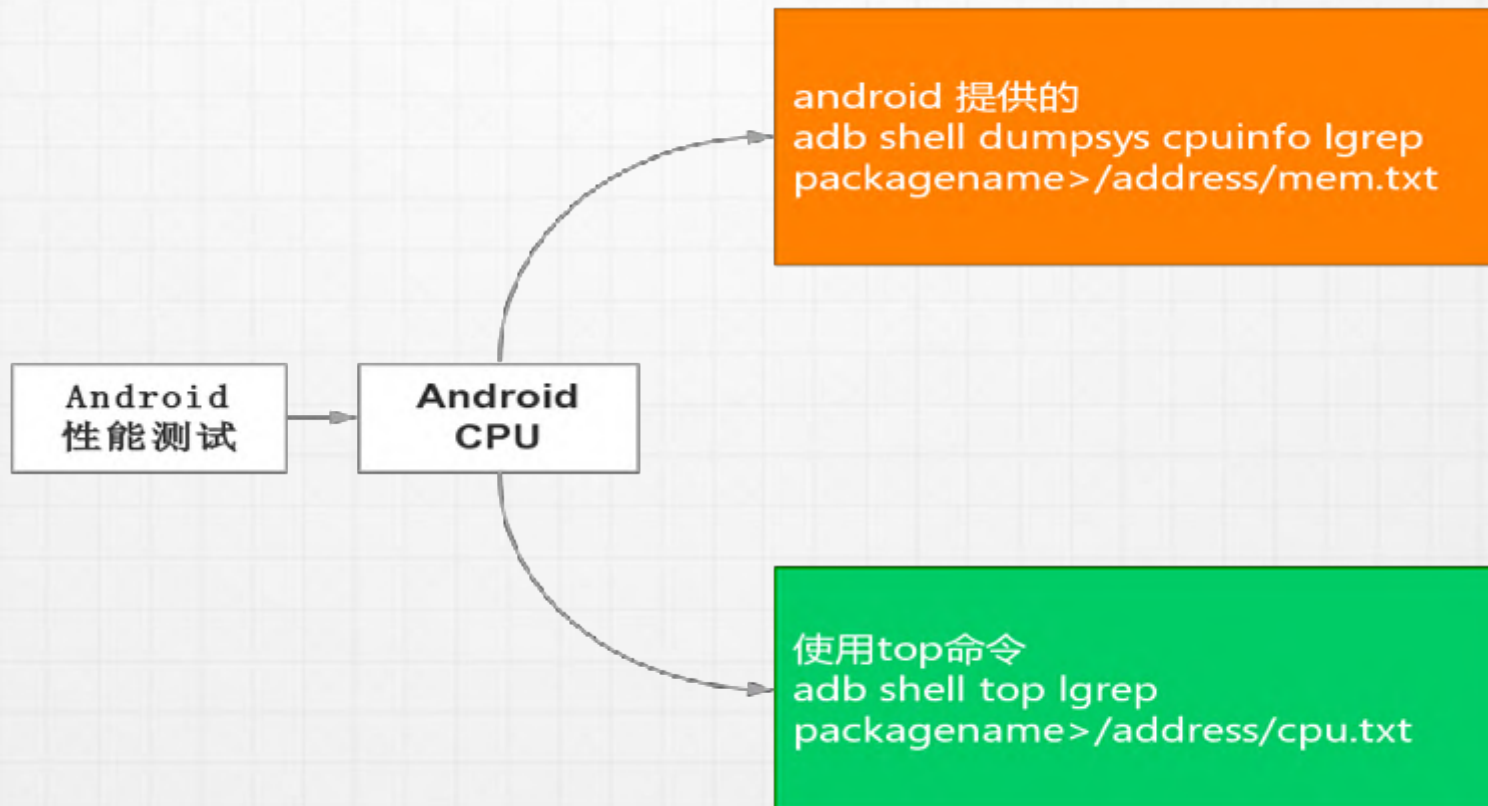
启动时间的采集方法



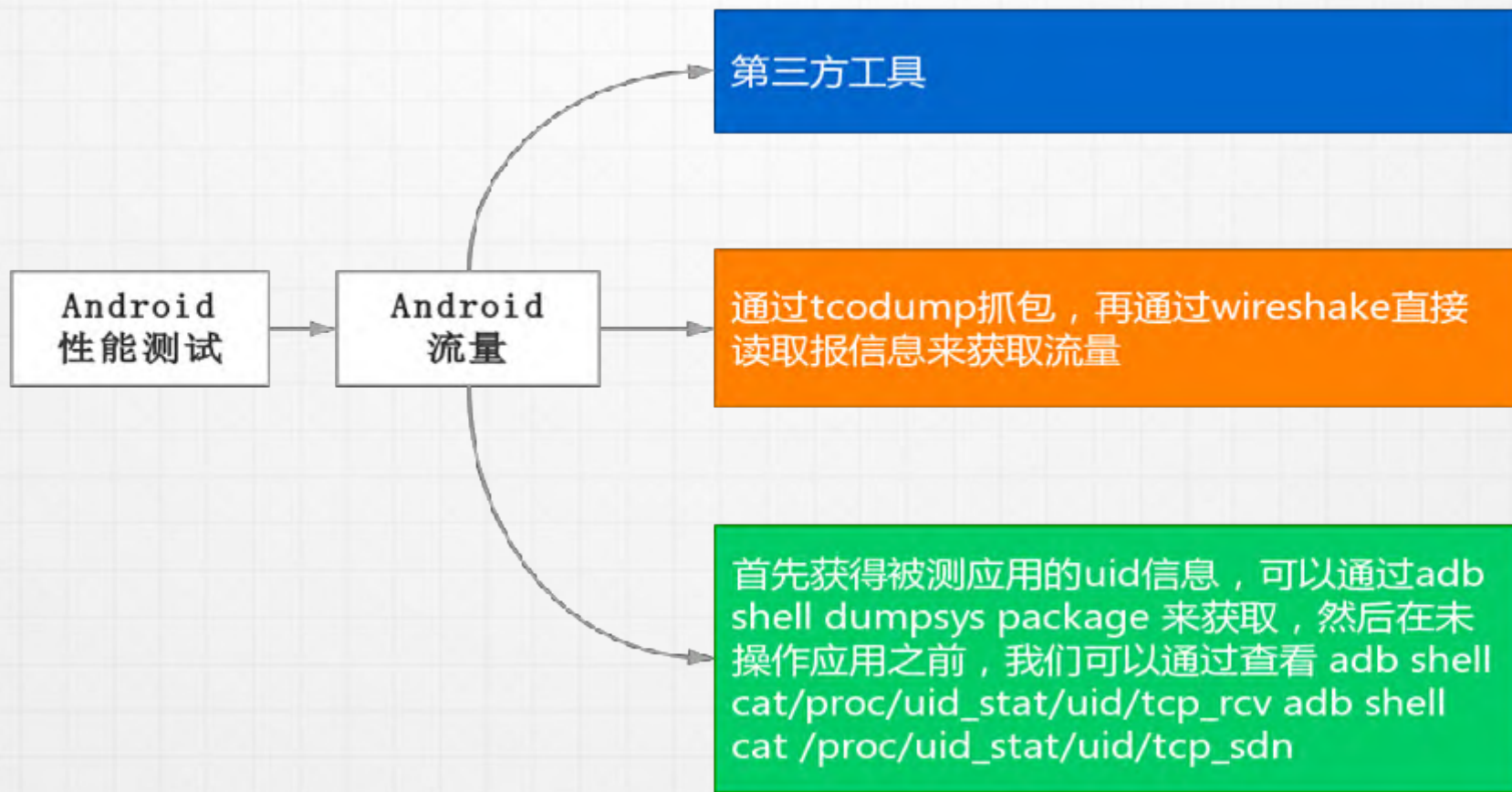
内存的采集方法



Cpu的采集方法



流量的采集方法



移动应用性能测试

1

移动应用性能取决于客户端及服务器端

2

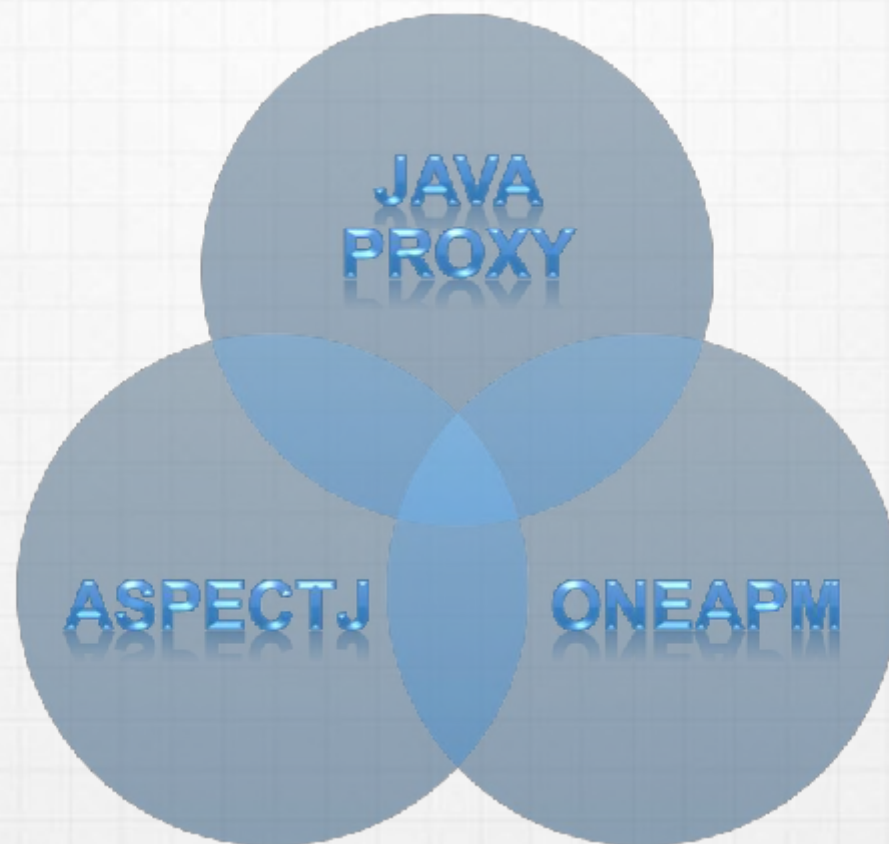
传统的测试很难全面的测试客户端与服务端之间的交互，通常会借助打印log日志以及fiddler等工具

3

即使测出了http请求响应时间，ViewLoading的时间，依旧很难测试bitmap解析，json解析的时间。进而相应的数据分析，就变得更困难了。

如何更好地采集性能测试维度数据并进行分析？

低侵入的数据采集方式



低侵入的数据采集方式



低侵入的数据采集方式

如何使用类**AOP**的方法
采集移动应用性能数据？

数据采集方式

```
public class Utils{
    public static long getCPU(){
        BufferedReader localBufferedReader = new BufferedReader
(new InputStreamReader(new FileInputStream("/proc/" + pid + "/stat")), 1000);
        String str = localBufferedReader.readLine();
        localBufferedReader.close();
        arrayOfString = str.split(" ");

        if(arrayOfString!=null&&arrayOfString.length>=16) {
            l = Long.parseLong(arrayOfString[13]) + Long.parseLong(arrayOfString[14
]) + Long.parseLong(arrayOfString[15]) + Long.parseLong(arrayOfString[16]);
        }
        return l;
    }
}
```


数据采集方式

```
public aspect CpuAspect {  
  
    pointcut getCpuState(): execution(* Activity+.*(..)) ;  
  
    before() : getCpuState() {  
  
        String method =thisJoinPoint.getSignature().toShortString();  
        Long cpuState = Utils.getCpu();  
  
        LOG.info(method + "\t" + System.currentTimeMillis() + "\t" +  
cpuState )  
    }  
}
```

插桩技术让测试如此容易



以前大量的测试成本消耗在测试执行阶段



使用插桩技术将测试的精力可以集中在核心的数据分析



产品发布前后都可以利用相同的技术进行监控



性能测试执行需要消耗

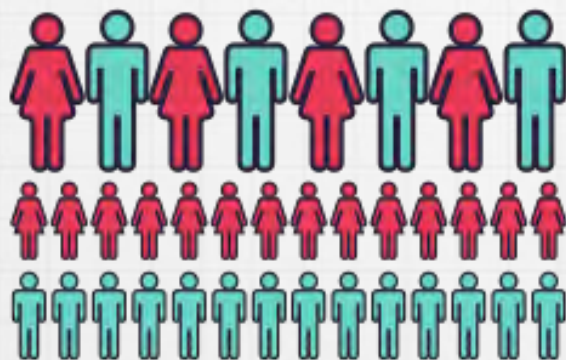
- ✓ 时间成本(投入的执行时间)
- ✓ 物料成本(机器成本)
- ✓ 技术成本(掌握特定的技术工具, [httpwatch](#), [fiddler](#))

第三方平台



性能测试数据分析

OneAPM 监测了一周的数据 (6 月 12 日至 6 月 18 日), 对 37164418 位活跃用户的请求数据进行统计分析



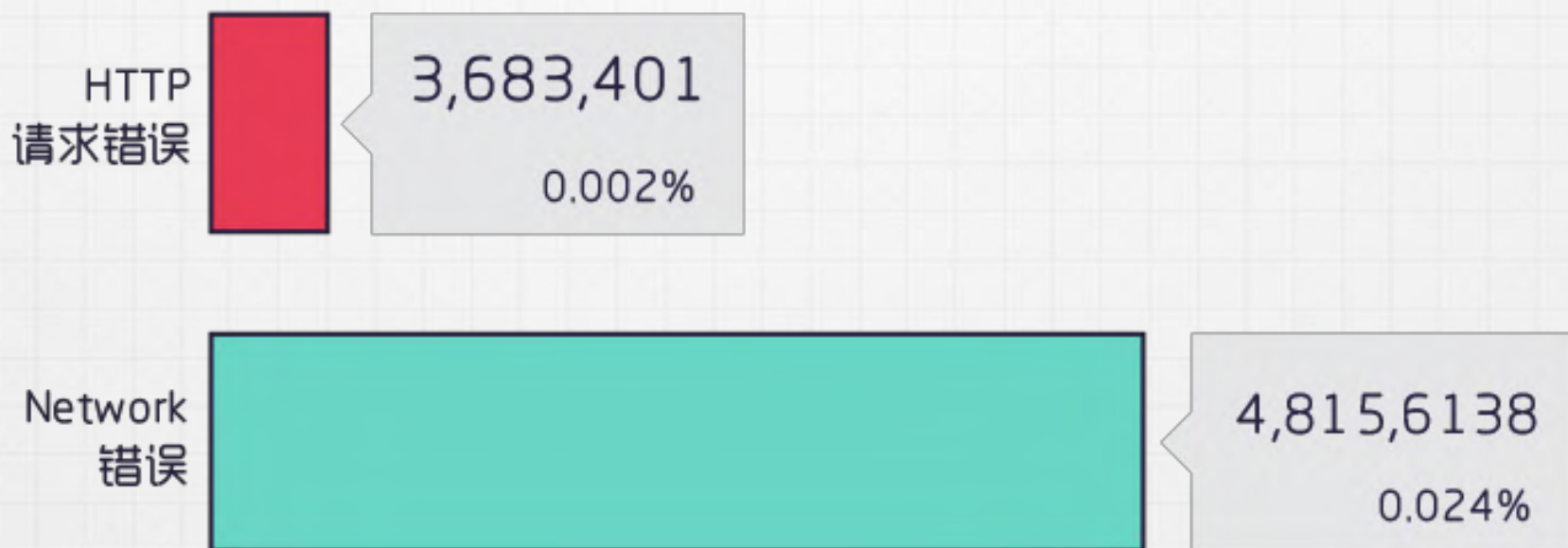
37,164,418 人



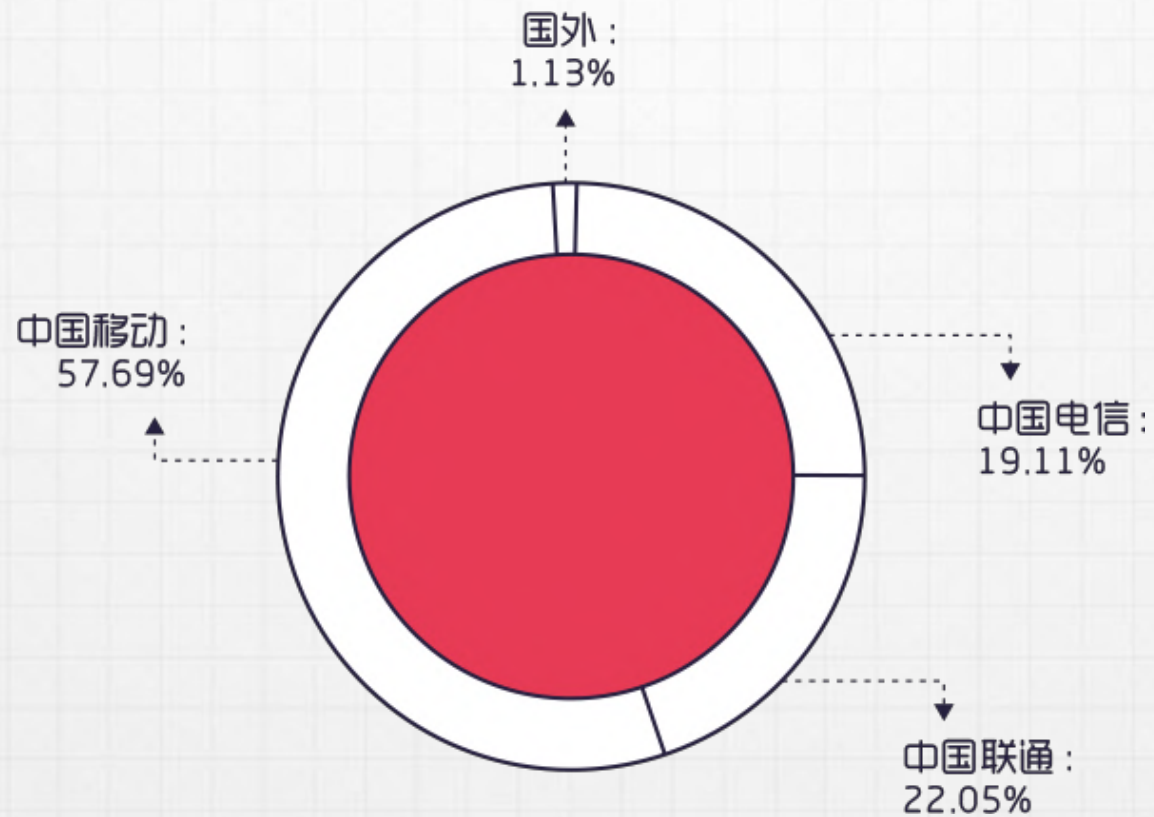
平均响应时间 1.45s

性能测试数据分析

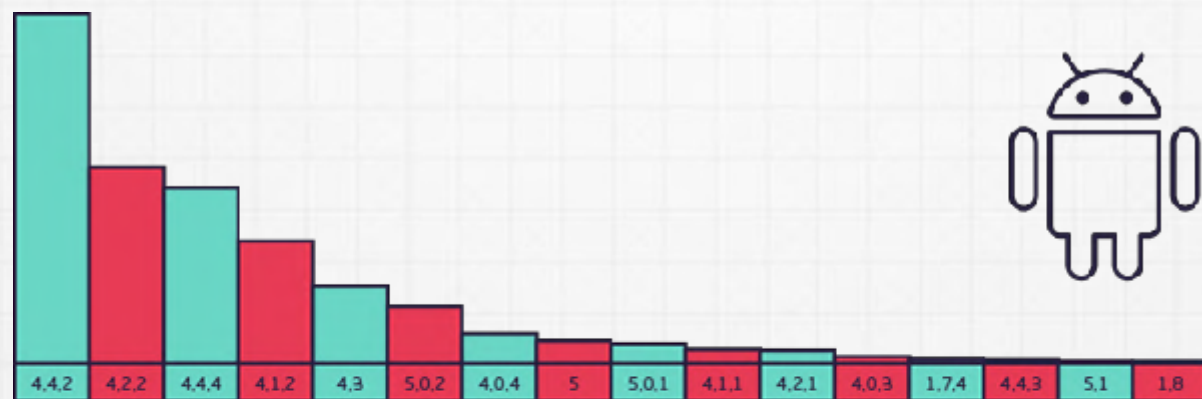
OneAPM 针对 2,039,121,367 次数据请求进行了分析



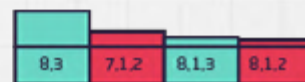
性能测试数据分析



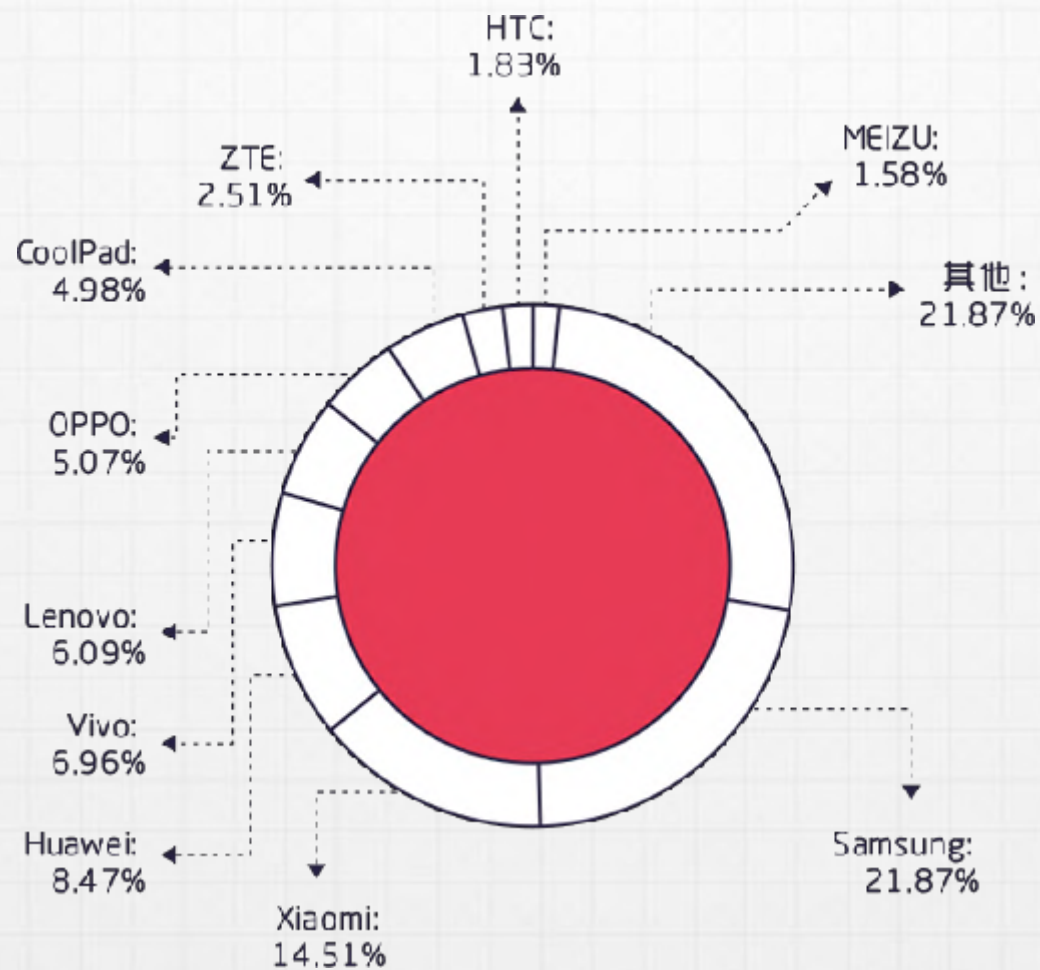
性能测试数据分析



iOS



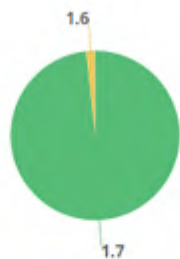
性能测试数据分析



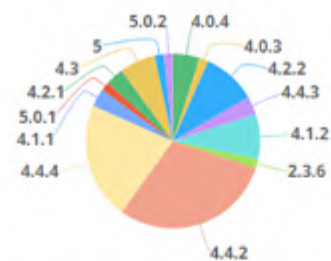
性能测试数据分析

采集次数：60

APP版本占比



操作系统占比

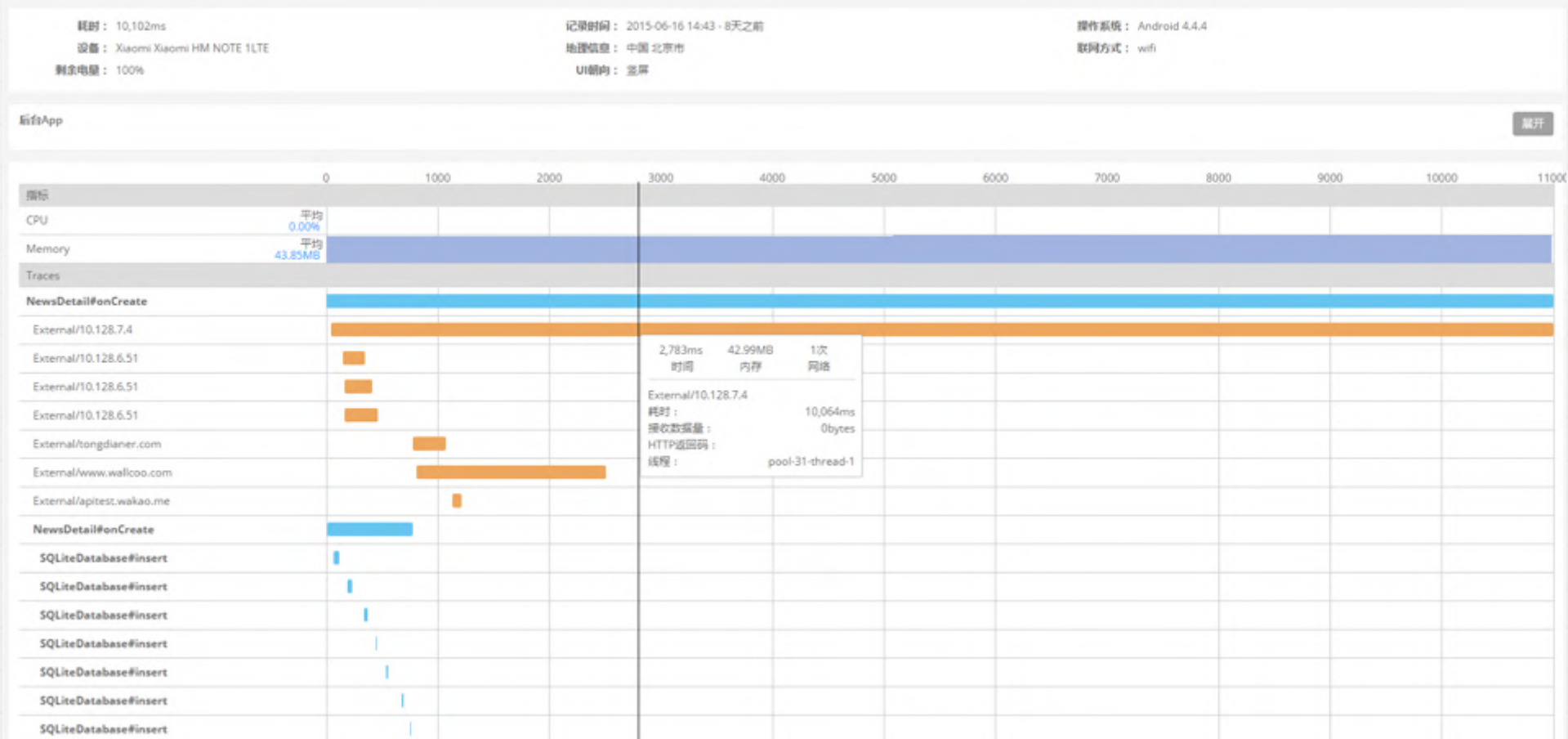


慢交互追踪列表

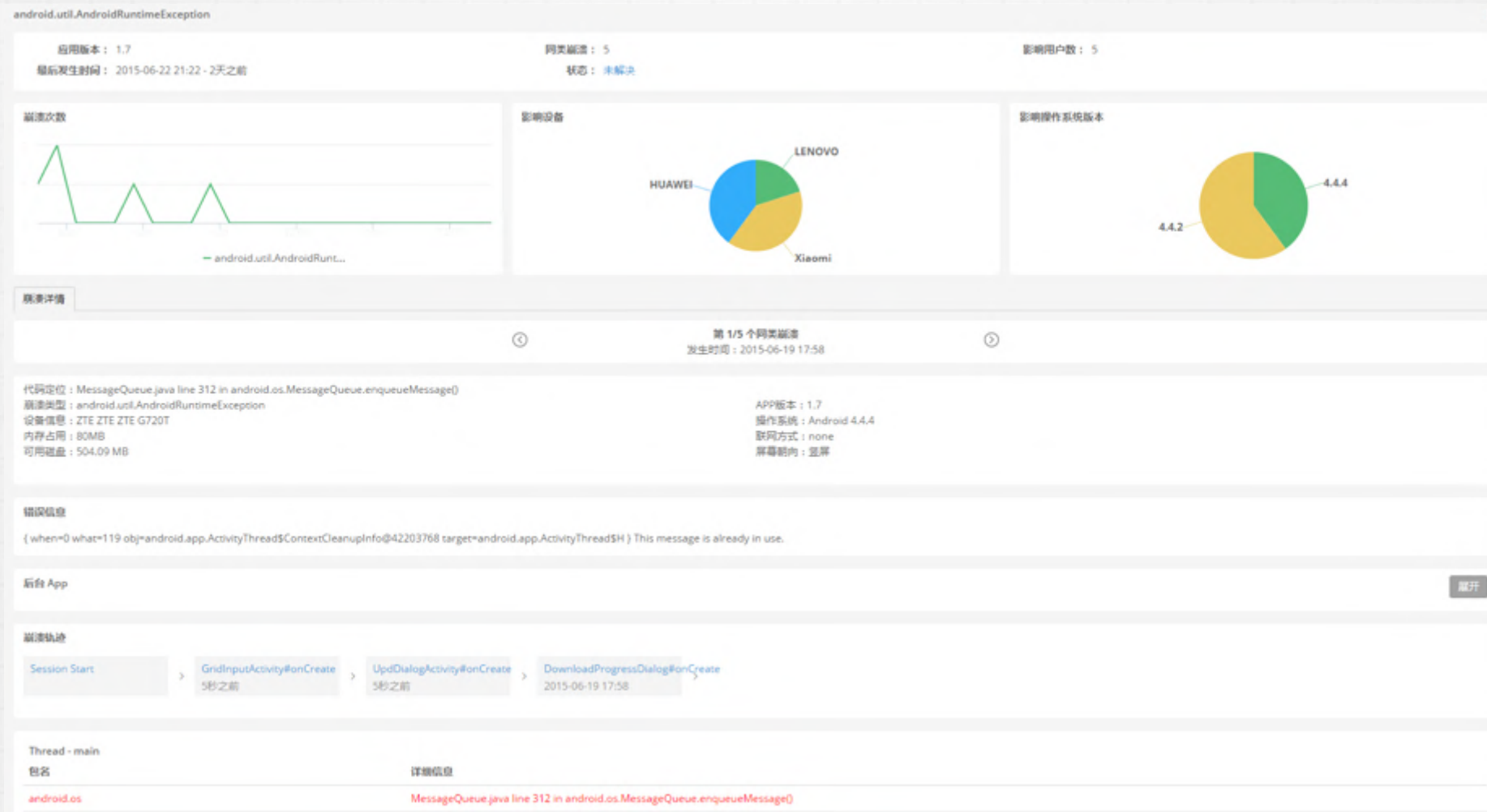
方法名	耗时(ms)	发生时间 +	操作系统及版本
DownloadProgressDialog#onCreate	29,874	2015-06-24 17:01	Android 4.3
DownloadProgressDialog#onCreate	15,420	2015-06-24 16:31	Android 4.1.2
DownloadProgressDialog#onCreate	33,796	2015-06-24 16:09	Android 4.4.4
DownloadProgressDialog#onCreate	48,105	2015-06-24 14:12	Android 4.2.2
DownloadProgressDialog#onCreate	16,051	2015-06-24 13:43	Android 4.1.2
DownloadProgressDialog#onCreate	1,117	2015-06-24 11:00	Android 4.4.2
DownloadProgressDialog#onCreate	1,510	2015-06-24 10:07	Android 4.3
DownloadProgressDialog#onCreate	15,775	2015-06-24 10:06	Android 4.2.2
DownloadProgressDialog#onCreate	11,469	2015-06-24 10:04	Android 4.4.4
DownloadProgressDialog#onCreate	4,005	2015-06-24 10:01	Android 4.1.1

[显示更多](#)

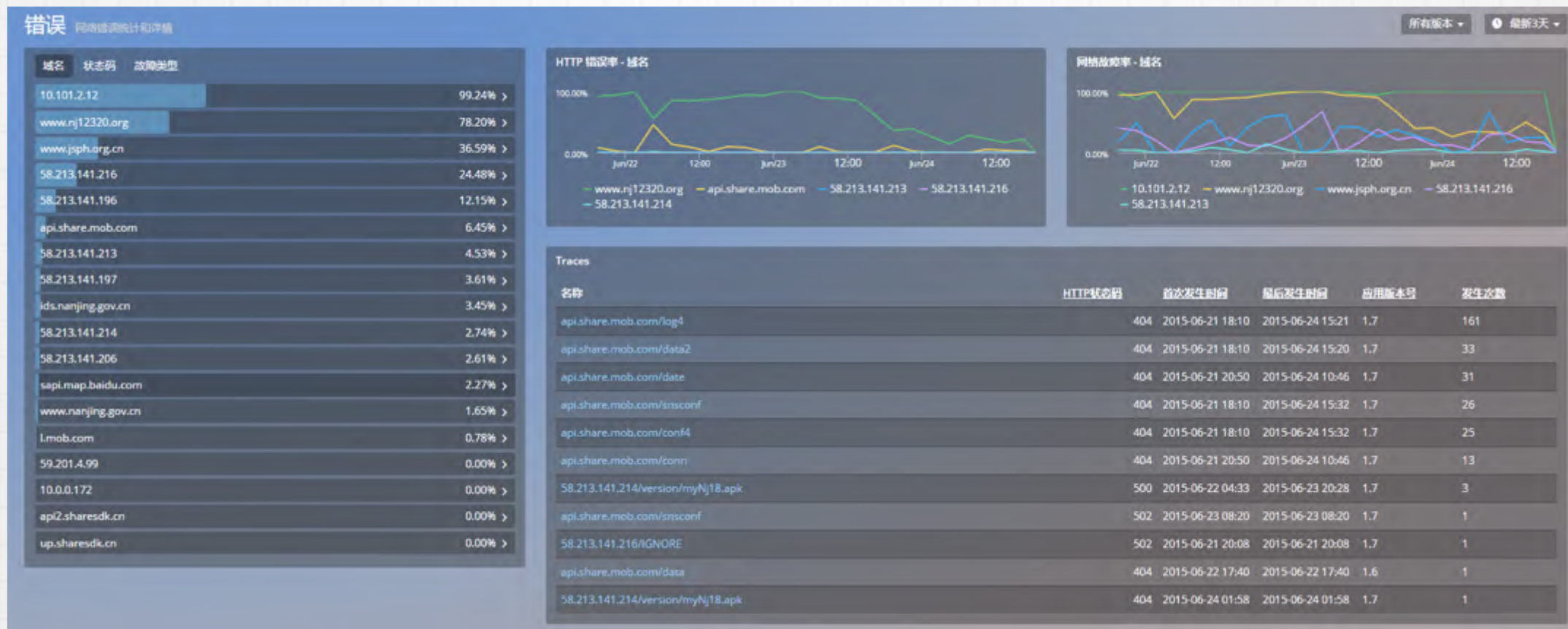
性能测试数据分析



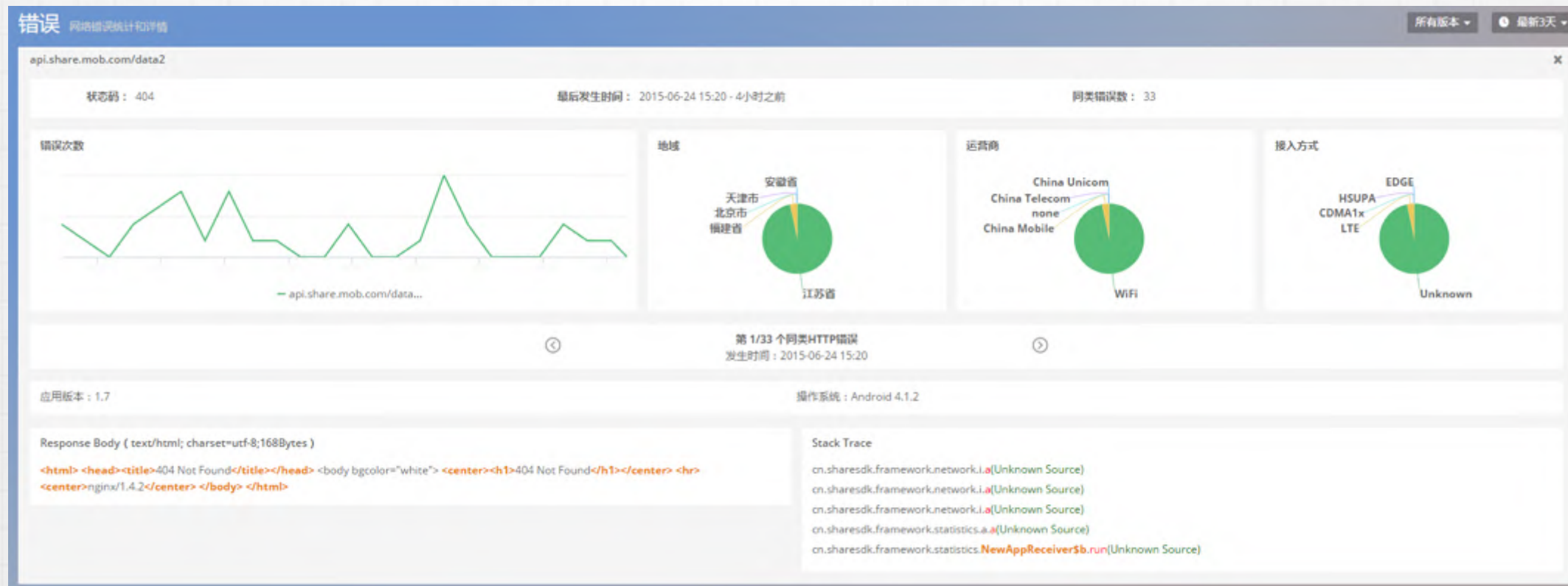
性能测试数据分析



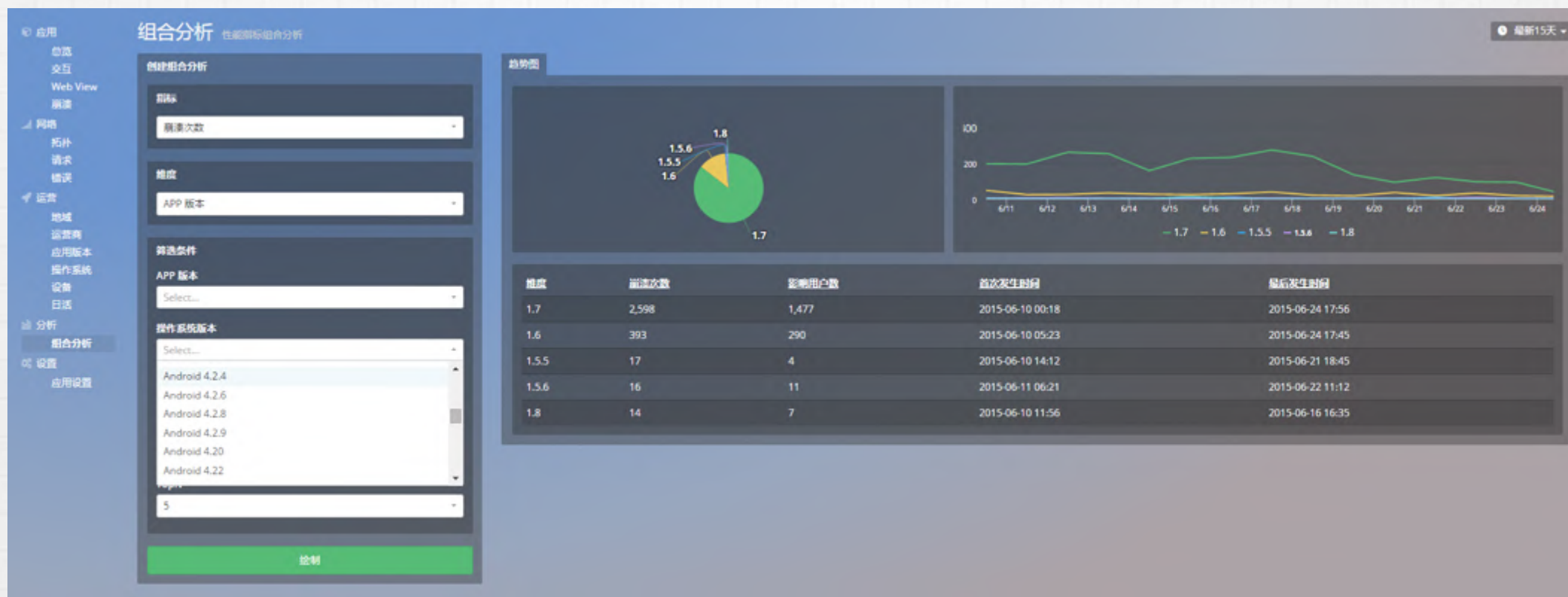
性能测试数据分析



性能测试数据分析



性能测试数据分析



加入我们



THANK YOU

软件定义世界，我们洞悉软件

