360技术嘉年华-测试之美

# 广告SDK专项测试方案

奇虎360-WEB平台部

QTest无线组--VPDong

# 目录

# 一、业务介绍

1、产品形态为安卓平台下的JAR包和AAR包
2、产品技术上通过插件化方式开发，可以按广告类型打不同的包
3、产品组成上为内外核，两者通过反射调用
4、产品逻辑中请求机制复杂，接口调用有一系列的前提和顺序

# 二、持续集成

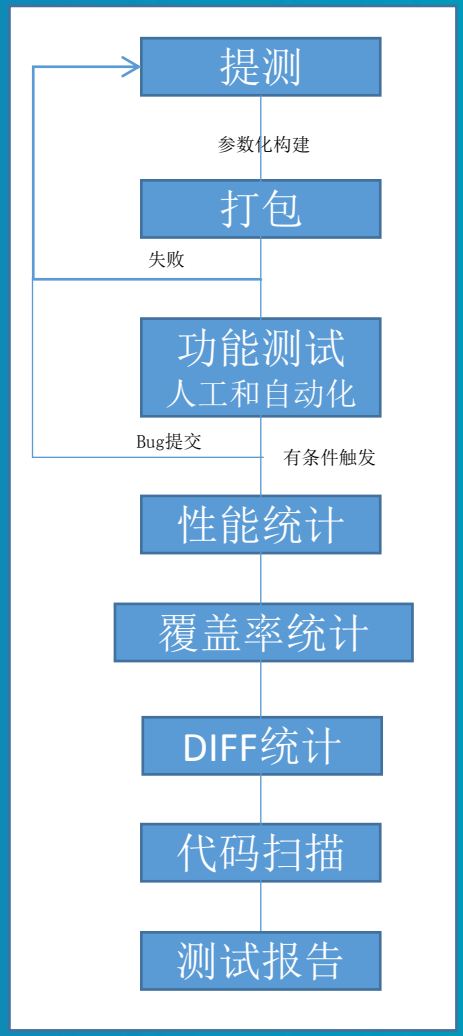
## 测试维度

人工功能测试　　性能统计　　覆盖率统计

自动化功能测试　　代码扫描　　DIFF统计

## 技术支持

Jenkins　　Fireline　　SDKAPM　　Mango



# 三、代码构建

SDK的构建(开发导向)：

1、打包产品--jar\aar

2、参数化打包渠道--game\inner\outer

3、参数化打包组件--ad type && miaozhen

DEMO的构建(测开导向)

1、参数化配置编译选项-buildToolsVersion\sdkVersion\V4pkg

2、参数化集成产品

3、参数化打包组件--ad type && miaozhen

1、参数化配置编译选项--动态配置编译选项

2、参数化集成产品--预置渠道jar/aar，并动态配置依赖

```
println("to add sdk...")
fileTree(dir: scanFolder.absolutePath).each { File scanFile ->
    println("scan file:" + scanFile.absolutePath)
    def compile4Aar = {...}
    compile4Aar()

    def compile4Jar = {
        fileTree(dir: 'libs/jars', include: ['*.aar', '*.jar']).each { File file ->
            if (file.name.endsWith(".aar")) {
                def aarName = file.name.lastIndexOf('.').with {
                    it != -1 ? file.name[0..<it] : file.name
                }
                jarCompile(name: aarName, ext: 'aar')
            }
            if (file.name.endsWith(".jar")) {
                def jarPath = 'libs/jars/' + file.name
                jarCompile files(jarPath)
            }
        }
    }
    compile4Jar()
```

```
task replace {
    doLast {
        println("into replace func...")
```

## 3、参数化打包组件--编码划模块，参数剔除未用代码

```
for (String codePath : codePaths) {
    fileTree(dir: codePath, include: ['*.java', '*.java.bk']).each {
        File file ->
            if (adFlag.equals("true")) {
                if (file.name.endsWith(".java.bk")) {
                    println("bk update to java:" + file.name)
                    file.renameTo(new File(file.parent, file.name.replace(".bk", "")))
                }
            } else if (adFlag.equals("false")) {
                if (file.name.endsWith(".java")) {
                    println("java update to bk:" + file.name)
                    file.renameTo(new File(file.parent, (file.name + ".bk")))
                }
            }
    }
}
```

```
String adBanner = args.getOrDefault('adBanner', "")
toRecode("adBanner", adBanner, 'hand/src/main/java/com/qihoo/video/view/banner')
```

# 四、功能测试

请求广告→校验请求广告数据→手动MOCK广告数据→展示广告→校验展示广告打点→……→曝光、点击、deeplink、下载

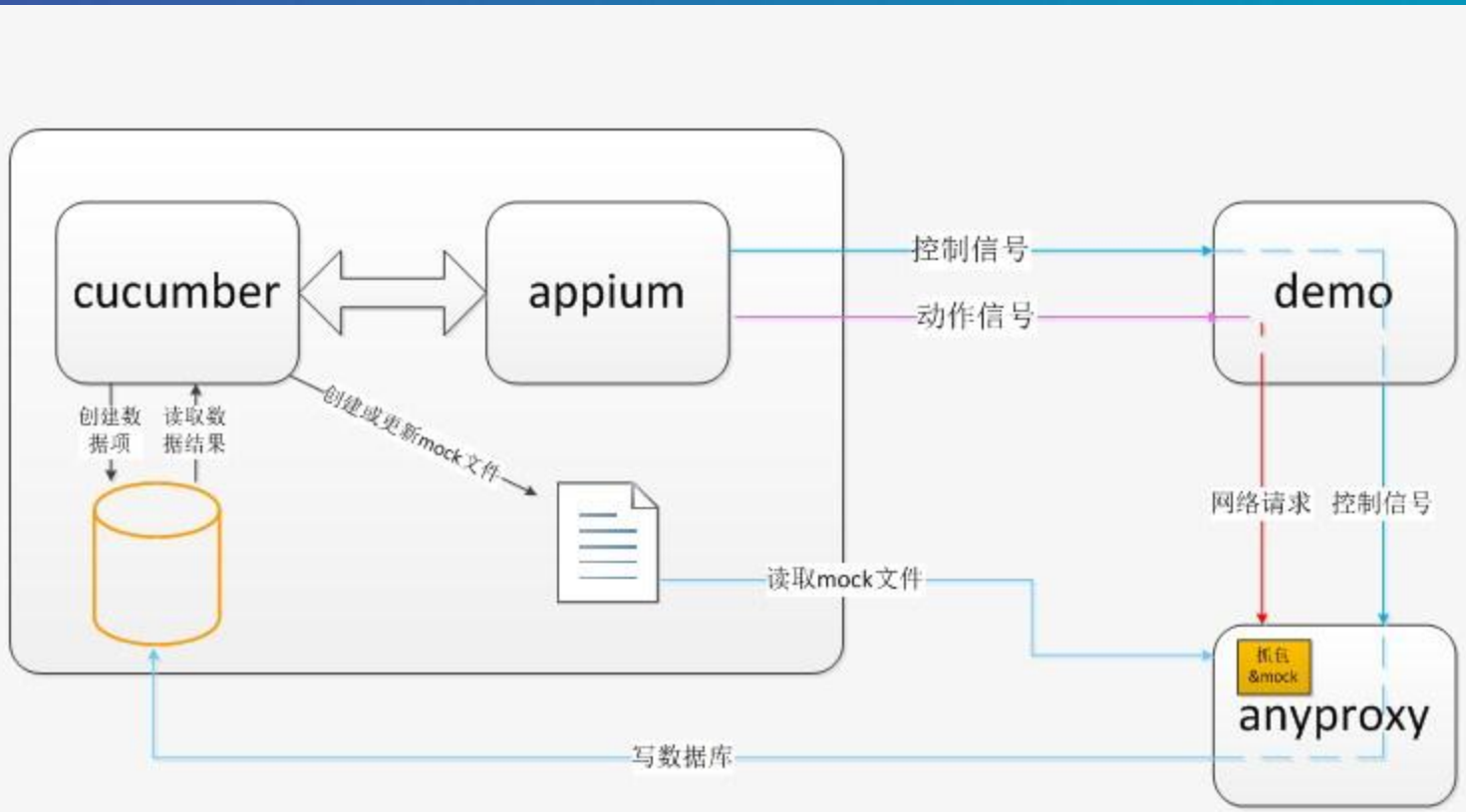
目前接入平台和厂商：十多家

目前迭代版本的频率：大约两周一个版本，不包括hotfix和定制

目前接入的媒体数量：大约两千家外部媒体，不包括内部使用

SO，日子还能过么？

# 自动化功能测试


cucumber
appium
控制信号
动作信号
demo
创建数据项
读取数据结果
创建或更新mock文件
网络请求
控制信号
读取mock文件
抓包&mock
anyproxy
写数据库

# 自动化功能测试--控制端(cucumber)

```
场景:MAX原生广告"请求"及"请求打点"完整性校验
  假设 caseID"10001"
  假设 初始化mock文件为"360原生广告"
  假设 打开原生广告界面
  并且 填写广告条数"3"
  并且 填写广告位条数"1"
  当 点击按钮"获取广告"
  那么 请求的完整性如"juhe-adrequest-schema"描述
  那么 请求及字段值校验如下
    |count|url|field|value|
    |1|https://show-
                                  ?|adtype|3|
    |1|https://show-
                                  ?|adspaceid|5a56tq08xf
|
  那么 请求的完整性如"juhe-tk-request-schema"描述
  并且 聚合请求打点字段校验
    |count|type|field|value|
    |1|0|agspaceid|ag5a56tq08xf|
    |1|0|adspaceid|5a56tq08xf|
    |1|0|agappkey|ag318422|
```

```
@假设("^caseID\"([^\"]*)\"$")
public void getCaseID(String arg0) throws Throwable {
    // Write code here that turns the phrase above into
concrete actions
   // throw new cucumber.api.PendingException();
}

@假设("^打开原生广告界面$")
public void openNativeActivity() throws IOException {

}

@并且("^填写广告条数\"([^\"]*)\"$")
public void fillInAdNum(String adnum) throws
IOException {

}

@当("^点击按钮\"([^\"]*)\"$")
public void clickButton(String btnType){

}

@那么("^请求及字段值校验如下$")
public void checkUrlField(List<UrlRequestField> list){

}
```

https://docs.cucumber.io/cucumber/step-definitions/

# 自动化功能测试--代理端(anyproxy)

具体地，我们提供的接口包括：

- 收到用户请求之后
  - **shouldUseLocalResponse**，是否在本地直接发送响应（不再向服务器发出请求）
  - **dealLocalResponse** 如果shouldUseLocalResponse返回true，会调用这个函数来获取本地响应内容（异步接口）
- 向服务端发出请求之前
  - **replaceRequestProtocol** 替换向服务器发出的请求协议，支持http和https的替换
  - **replaceRequestOption** 替换向服务器发出的请求参数，即nodeJS中的 request option
  - **replaceRequestData** 替换请求的body
- 向用户返回服务端的响应之前
  - **replaceResponseStatusCode** 替换服务器响应的http状态码
  - **replaceResponseHeader** 替换服务器响应的http头
  - **replaceServerResDataAsync** 替换服务器响应的数据（异步接口）
  - **pauseBeforeSendingResponse** 在请求返回给用户前的延迟时间

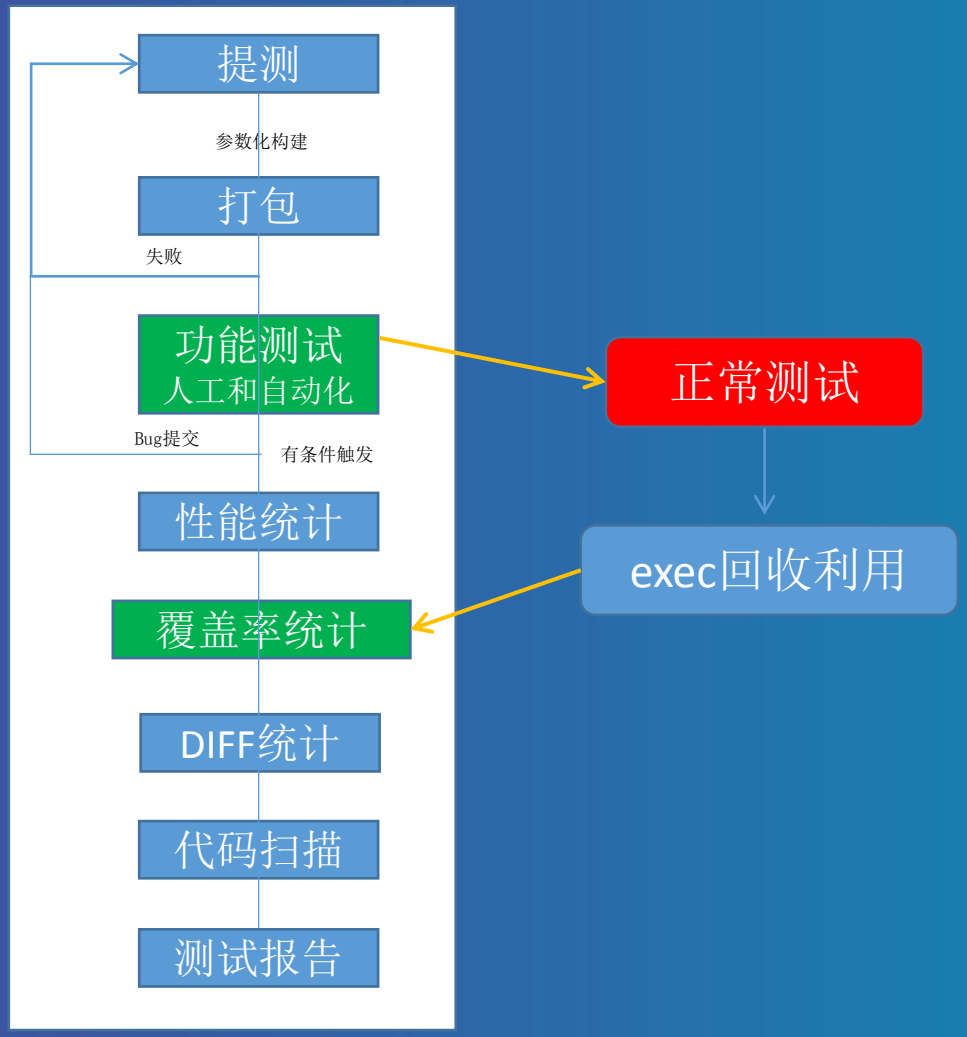
https://github.com/alibaba/anyproxy

# 自动化功能测试--校验(json-schema)

```
@Test
public void testJsonSchema2() {
  String failure = new String("{\"foo\":1234}");
  String Schema = "{\"type\": \"object\", \"properties\" : {\"foo\" : {\"type\" : \"string\"}}}";
  ProcessingReport report = null;
  try {
    JsonNode data = JsonLoader.fromString(failure);
    JsonNode schema = JsonLoader.fromString(Schema);
    report = JsonSchemaFactory.byDefault().getValidator().validateUnchecked(schema, data);
  } catch (IOException e) {
    e.printStackTrace();
  }
  //Assert.assertTrue(report.isSuccess());
  Iterator<ProcessingMessage> it = report.iterator();
  while (it.hasNext()) {
    System.out.println(it.next());
  }
}
```

```
    }
}
```

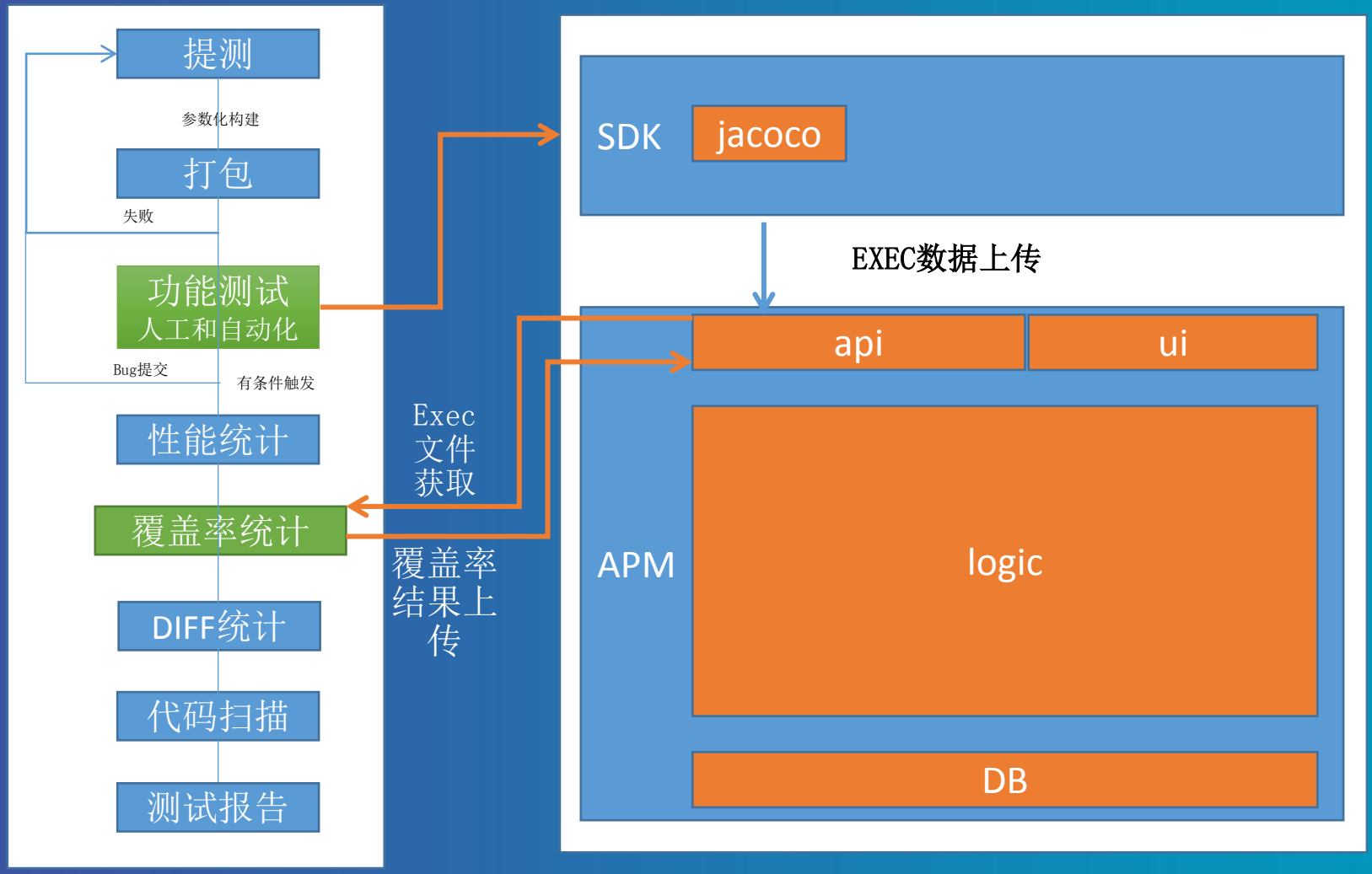
# 五、覆盖率统计




如何在正常测试前提下收集数据？

如何与现有CI流程结合？

# 当前SDK的测试解决方案



# 覆盖率统计--打包注入

```
android.defaultPublishConfig = "debug"
android.publishNonDefault = true
android.buildTypes {
    release {}
    debug {
        initWith release
        testCoverageEnabled = true
    }
}

if (project.name.equals("Bridge") || project.name.equals("sdk-framework")) {
    buildJar.jarExcludes += 'org/jacoco/agent/rt/**'
    buildJar.compliePath = "build/intermediates/transforms/jacoco/debug/folders/1/1/main"
}
```

```
if line != "apply from: \"$project.rootDir/torchsdktest/jacoco/jacoco.gradle\"":
    print("注入目标:" + fpath)
    fo = open(fpath, "a", encoding='UTF-8')
    fo.write("\n")
    fo.write("apply from: \"$project.rootDir/torchsdktest/jacoco/jacoco.gradle\"")
    fo.close()
```

# 六、性能测试

SDK的重点关注点：

1、方法耗时统计

2、内存泄漏与溢出的统计

3、崩溃统计

4、cpu\fps等

# 调研过...

VirtualApp:

VirtualApp是一个开源的Android App虚拟化引擎，其在App内创建一个虚拟空间，在该虚拟空间内任意的安装、启动和卸载APK，这一切都与外部隔离，如同一个沙盒。VA目前被广泛应用于插件化开发、无感知热更新、APP多开、APP云加载、移动办公安全、军队政府保密、手机模拟信息、隐私保护、脚本自动化、自动化测试、游戏手柄免激活等技术领域，但它决不仅限于此，免安装运行APK这一Feature打开了无限可能--------这都取决于您的想象力。----https://github.com/asLody/VirtualApp/blob/master/CHINESE.md

为什么放弃了呢?

针对APP的免安装，对于SDK无能为力

针对测试demo的话，有点杀鸡焉用宰牛刀的寓意，产出比不高

可利用的是什么？

对jar进行dex化后的独立运行--需要借助解析脚本，但是对aar无能为力

后续调研将sdkapm中的sdk部分集成到VA代码中，形成一个通用平台

# 调研过...

Frida:

Frida是个轻量级so级别的hook框架，它可以帮助逆向人员对指定的进程的so模块进行分析。它主要提供了功能简单的python接口和功能丰富的js接口，使得hook函数和修改so编程化，值得一提的是接口中包含了主控端与目标进程的交互接口，由此我们可以即时获取信息并随时进行修改。使用frida可以获取进程的信息（模块列表，线程列表，库导出函数），可以拦截指定函数和调用指定函数，可以注入代码，总而言之，使用frida我们可以对进程模块进行手术刀式剖析。----https://www.frida.re/

为什么放弃了呢?

需要root，非root方案也有，但是功能就较弱。

没有办法增强代码，只是hook后调用已经存在的代码。

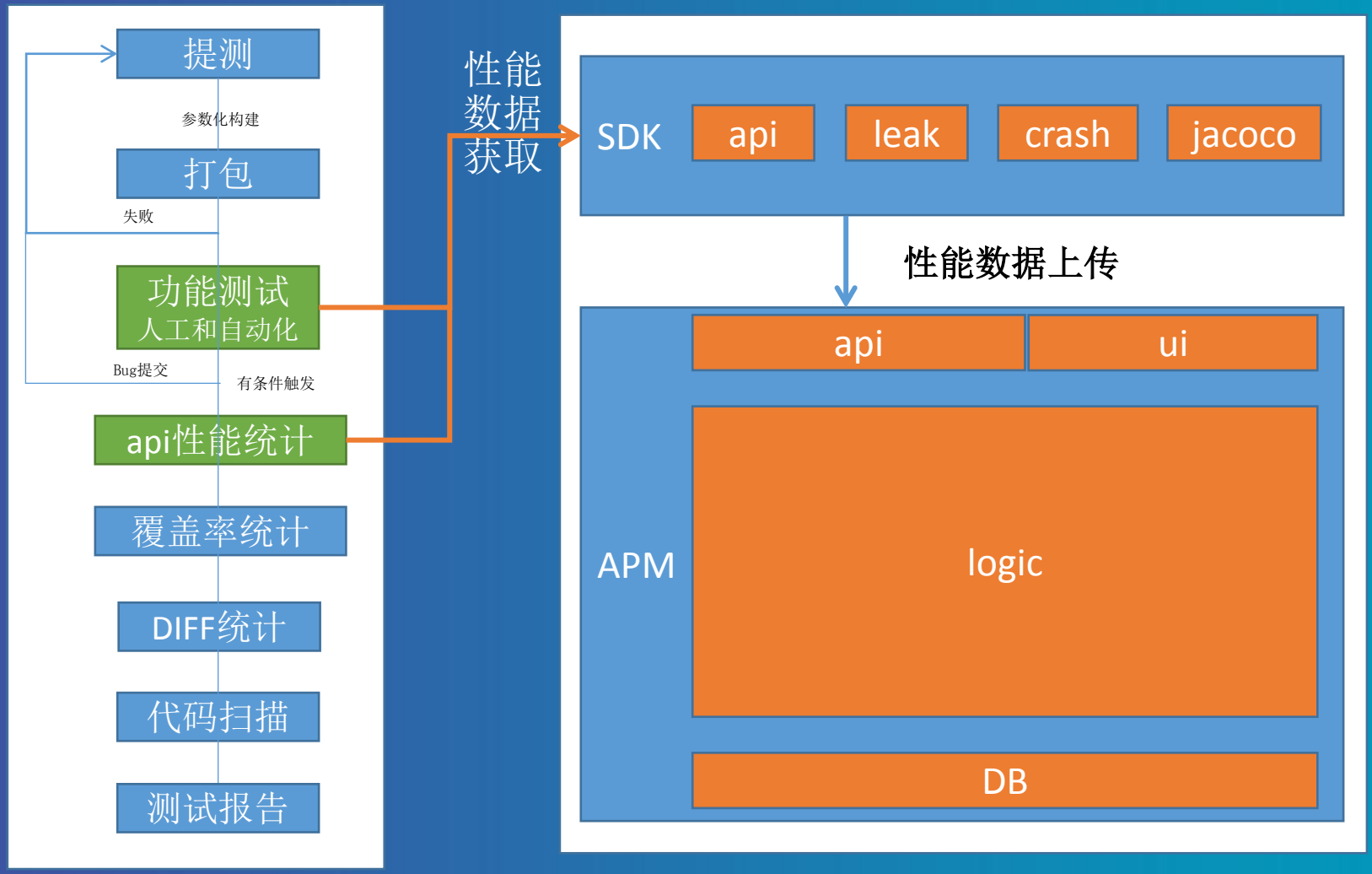
可利用的是什么？

无需注入代码进行hook统计

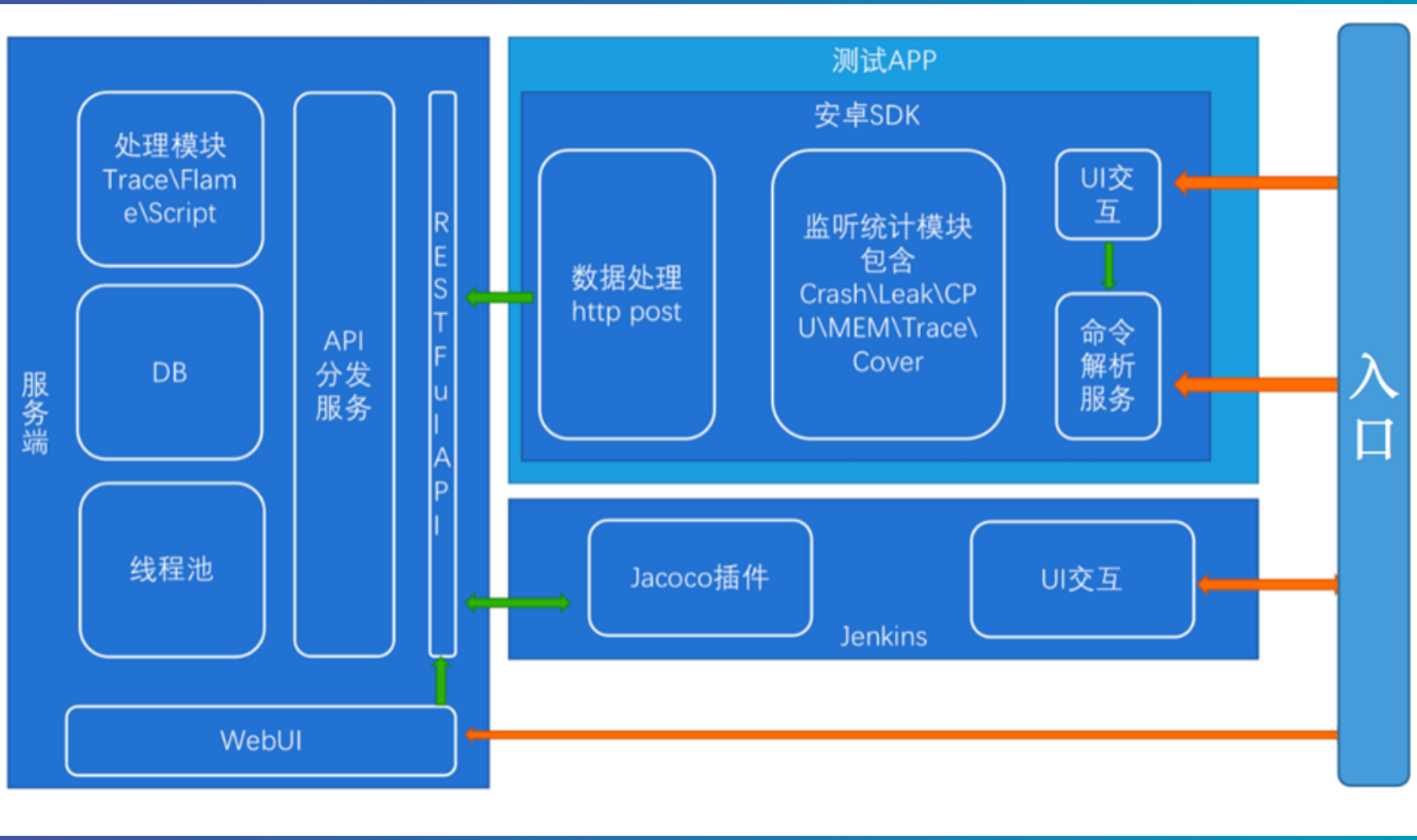
so文件的操作

后续调研增强VA下的操作

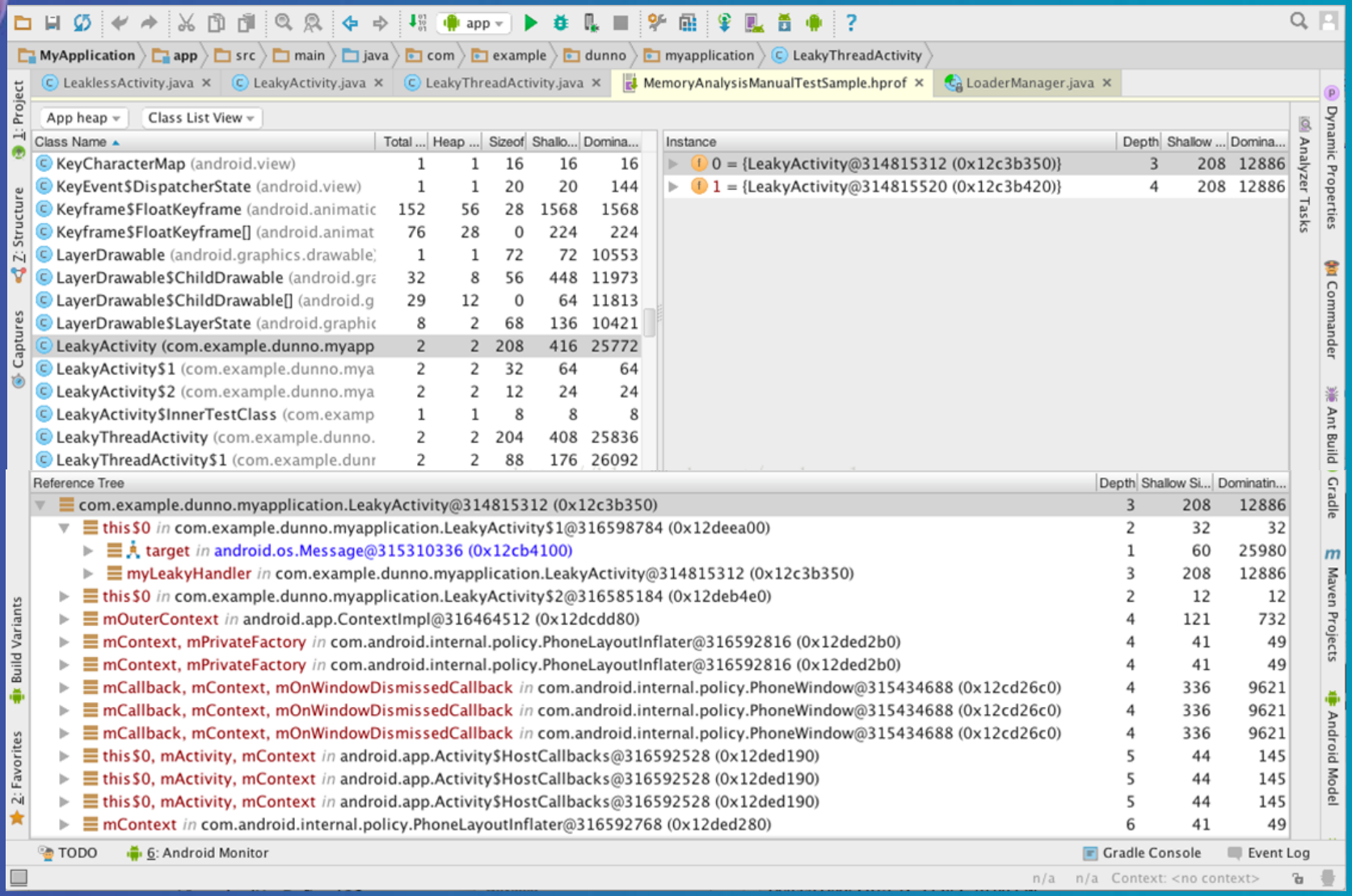
# 当前SDK的测试解决方案



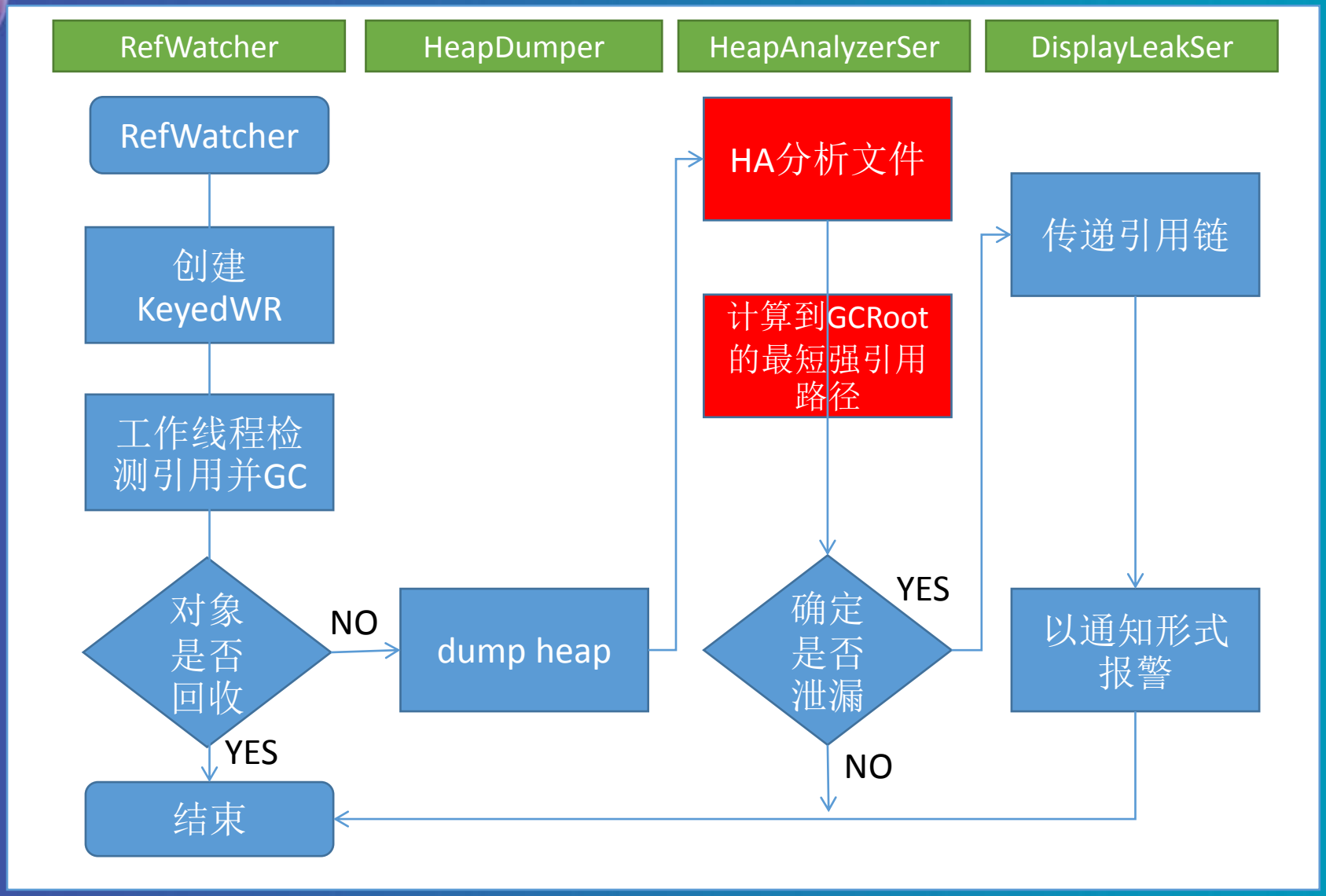
# 性能统计--整体流程


服务端
处理模块
Trace\Flame\Script
DB
线程池
API
分发
服务
RESTFUIAPI
WebUI
测试APP
安卓SDK
数据处理
http post
监听统计模块
包含
Crash\Leak\CP
U\MEM\Trace\
Cover
UI交
互
命令
解析
服务
Jacoco插件
UI交互
Jenkins
入
口

# 性能测试--Leak统计


MyApplication app src main java com example dunno myapplication LeakyThreadActivity
LeaklessActivity.java
LeakyActivity.java
LeakyThreadActivity.java
MemoryAnalysisManualTestSample.hprof
LoaderManager.java
App heap
Class List View
Class Name Total ... Heap ... Sizeof Shallo... Domina...
KeyCharacterMap (android.view) 1 1 16 16 16
KeyEvent$DispatcherState (android.view) 1 1 20 20 144
Keyframe$FloatKeyframe (android.animatic 152 56 28 1568 1568
Keyframe$FloatKeyframe[] (android.animat 76 28 0 224 224
LayerDrawable (android.graphics.drawable) 1 1 72 72 10553
LayerDrawable$ChildDrawable (android.gra 32 8 56 448 11973
LayerDrawable$ChildDrawable[] (android.g 29 12 0 64 11813
LayerDrawable$LayerState (android.graphic 8 2 68 136 10421
LeakyActivity (com.example.dunno.myapp 2 2 208 416 25772
LeakyActivity$1 (com.example.dunno.mya 2 2 32 64 64
LeakyActivity$2 (com.example.dunno.mya 2 2 12 24 24
LeakyActivity$InnerTestClass (com.examp 1 1 8 8 8
LeakyThreadActivity (com.example.dunno. 2 2 204 408 25836
LeakyThreadActivity$1 (com.example.dunr 2 2 88 176 26092
Instance Depth Shallow ... Domina...
0 = {LeakyActivity@314815312 (0x12c3b350)} 3 208 12886
1 = {LeakyActivity@314815520 (0x12c3b420)} 4 208 12886
Reference Tree Depth Shallow Si... Dominatin...
com.example.dunno.myapplication.LeakyActivity@314815312 (0x12c3b350) 3 208 12886
this$0 in com.example.dunno.myapplication.LeakyActivity$1@316598784 (0x12deea00) 2 32 32
target in android.os.Message@315310336 (0x12cb4100) 1 60 25980
myLeakyHandler in com.example.dunno.myapplication.LeakyActivity@314815312 (0x12c3b350) 3 208 12886
this$0 in com.example.dunno.myapplication.LeakyActivity$2@316585184 (0x12deb4e0) 2 12 12
mOuterContext in android.app.ContextImpl@316464512 (0x12dcdd80) 4 121 732
mContext, mPrivateFactory in com.android.internal.policy.PhoneLayoutInflater@316592816 (0x12ded2b0) 4 41 49
mContext, mPrivateFactory in com.android.internal.policy.PhoneLayoutInflater@316592816 (0x12ded2b0) 4 41 49
mCallback, mContext, mOnWindowDismissedCallback in com.android.internal.policy.PhoneWindow@315434688 (0x12cd26c0) 4 336 9621
mCallback, mContext, mOnWindowDismissedCallback in com.android.internal.policy.PhoneWindow@315434688 (0x12cd26c0) 4 336 9621
mCallback, mContext, mOnWindowDismissedCallback in com.android.internal.policy.PhoneWindow@315434688 (0x12cd26c0) 4 336 9621
this$0, mActivity, mContext in android.app.Activity$HostCallbacks@316592528 (0x12ded190) 5 44 145
this$0, mActivity, mContext in android.app.Activity$HostCallbacks@316592528 (0x12ded190) 5 44 145
this$0, mActivity, mContext in android.app.Activity$HostCallbacks@316592528 (0x12ded190) 5 44 145
mContext in com.android.internal.policy.PhoneLayoutInflater@316592768 (0x12ded280) 6 41 49
1: Project
Z: Structure
Captures
Build Variants
2: Favorites
Analyzer Tasks
Dynamic Properties
Commander
Ant Build
Gradle
Maven Projects
Android Model
TODO
6: Android Monitor
Gradle Console
Event Log
n/a n/a Context: <no context>

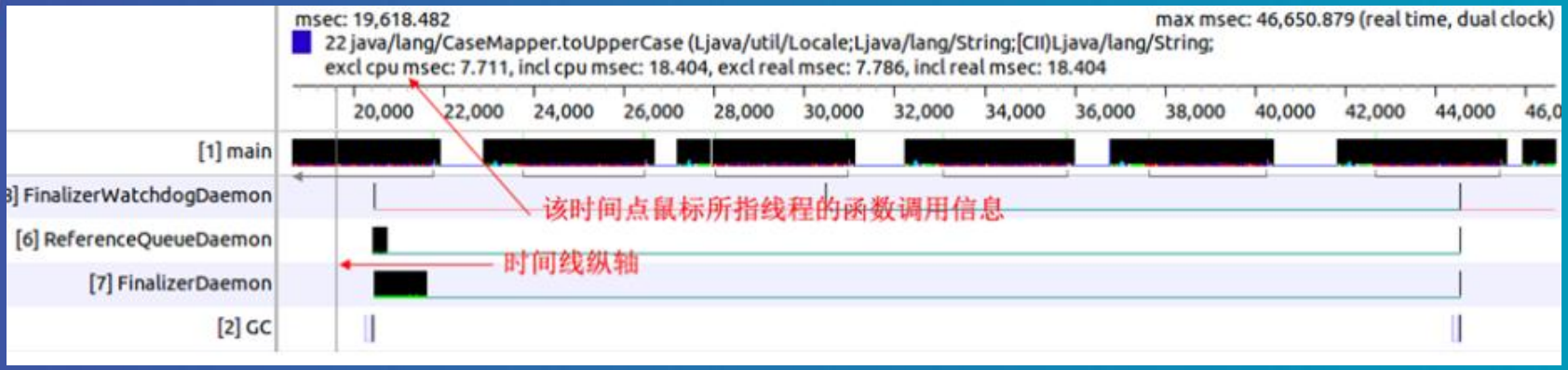
# 性能测试--Leak统计


RefWatcher
HeapDumper
HeapAnalyzerSer
DisplayLeakSer
RefWatcher
创建 KeyedWR
工作线程检测引用并GC
对象是否回收
NO
YES
结束
dump heap
HA分析文件
计算到GCRoot的最短强引用路径
确定是否泄漏
YES
NO
传递引用链
以通知形式报警

# 性能测试--Leak统计

```
@Override
protected void afterDefaultHandling(final HeapDump heapDump, final AnalysisResult result, final Str
    if (!result.leakFound || result.excludedLeak) return;
    if (gLeakListener != null) {
        TaskUtil.executOnPoll(() → {
                if (gWorkSpace != null && gWorkSpace.exists() && heapDump.heapDumpFile.exists()) {
                    String fileName = String.format("%s-%s.hprof", InfoUtil.More.getProcessName(get
                    File infoFile = FileUtil.addFile( isDir: false, gWorkSpace, fileName);
                    try {
                        FileUtil.copyFile(heapDump.heapDumpFile, infoFile);
                        gLeakListener.toLeak(infoFile, leakInfo);
                    } catch (Exception e) {
                        gLeakListener.toLeak(heapDump.heapDumpFile, leakInfo);
                    }
                } else {
                    gLeakListener.toLeak(heapDump.heapDumpFile, leakInfo);
                }
        });
    }
}
```

# 性能测试--API统计




| Name | Cpu Time/Ca | Incl Cpu Time | Incl Cpu Time | Calls+Recur Calls/Total | Excl Cpu Time | Excl Cpu Time | Incl Real Time |
|---|---|---|---|---|---|---|---|
| 34 android/app/Instrumentation.callActivityOnCreate (Landroid/ap | 4619.580 | 10.3% | 4619.580 | 1+0 | 0.0% | 0.126 | 3.0% |
| 35 android/app/Activity.performCreate (Landroid/os/Bundle;)V | 4619.454 | 10.3% | 4619.454 | 1+0 | 0.0% | 0.317 | 3.0% |
| 36 com/example/traceviewdemo/MainActivity.onCreate (Landroid/ | 4618.684 | 10.3% | 4618.684 | 1+0 | 0.0% | 0.628 | 3.0% |
| 0 (toplevel) | 3735.158 | 100.0% | 44821.895 | 12+0 | 0.2% | 100.295 | 100.0% |
| 62 com/example/traceviewdemo/MainActivity.getStringsToShow () | 2921.913 | 6.5% | 2921.913 | 1+0 | 0.3% | 132.456 | 1.8% |
| 71 android/w | | | | | | 0.246 | 1.7% |
| 3 android/wi | | | | | | 2.683 | 16.0% |
| 4 android/wi | | | | | | 4.123 | 15.3% |
| 5 android/wi | | | | | | 1.271 | 15.3% |
| 6 android/wi | | | | | | 2.613 | 15.3% |
| 7 com/examp | | | | | | 2.452 | 15.2% |
| 8 com/examp | | | | | | 25.963 | 15.1% |
| 88 android/a | | | | | | 0.244 | 1.2% |
| 89 com/andr | | | | | | 0.395 | 1.1% |
| 97 com/andr | | | | | | 1.654 | 0.9% |
| 99 com/andr | | | | | | 5.514 | 0.9% |
| 45 android/vi | | | | | | 1.133 | 2.0% |

| Name | Cpu Time/Ca | Incl Cpu Time | Incl Cpu Time | Calls+Recur Calls/Total |
|---|---|---|---|---|
| 34 android/app/Instrumentation.callActivityOnCreate (Landroid/ap | 4619.580 | 10.3% | 4619.580 | 1+0 |
| 35 android/app/Activity.performCreate (Landroid/os/Bundle;)V | 4619.454 | 10.3% | 4619.454 | 1+0 |
| 36 com/example/traceviewdemo/MainActivity.onCreate (Landroid/ | 4618.684 | 10.3% | 4618.684 | 1+0 |
| Parents | | | | |
| 35 android/app/Activity.performCreate (Landroid/os/Bundle;)V | | 100.0% | 4618.684 | 1/1 |
| Children | | | | |
| self | | 0.0% | 0.628 | |
| 62 com/example/traceviewdemo/MainActivity.getStringsToSho | | 63.3% | 2921.913 | 1/1 |
| 88 android/app/Activity.setContentView (I)V | | 36.3% | 1674.574 | 1/1 |
| 759 android/widget/ListView.setAdapter (Landroid/widget/List | | 0.3% | 13.998 | 1/1 |

# 性能测试--API统计

```
*version
1
clock=global
*threads
1 main
6 JDWP Handler
5 Async GC
4 Reference Handler
3 Finalizer
2 Signal Handler
*methods
0x080f23f8 java/io/PrintStream write ([BII)V
0x080f25d4 java/io/PrintStream print (Ljava/lang/String;)V
0x080f27f4 java/io/PrintStream println (Ljava/lang/String;)V
0x080da620 java/lang/RuntimeException <init> ()V
...
0x080f630c android/os/Debug startMethodTracing ()V
0x080f6350 android/os/Debug startMethodTracing ()V

*end
```

```
File format:
* header
* record 0
* record 1
* ...

Header format:
* u4 magic 0x574f4c53 ('SLOW')
* u2 version
* u2 offset to data
* u8 start date/time in usec

Record format:
* u1 thread ID
* u4 method ID | method action
* u4 time delta since start, in usec
```

# 性能测试--API统计

1、MethodStack

<long> enterTime/exitTime

<List> callerList

2、ThreadStack

<Stack> helperStack

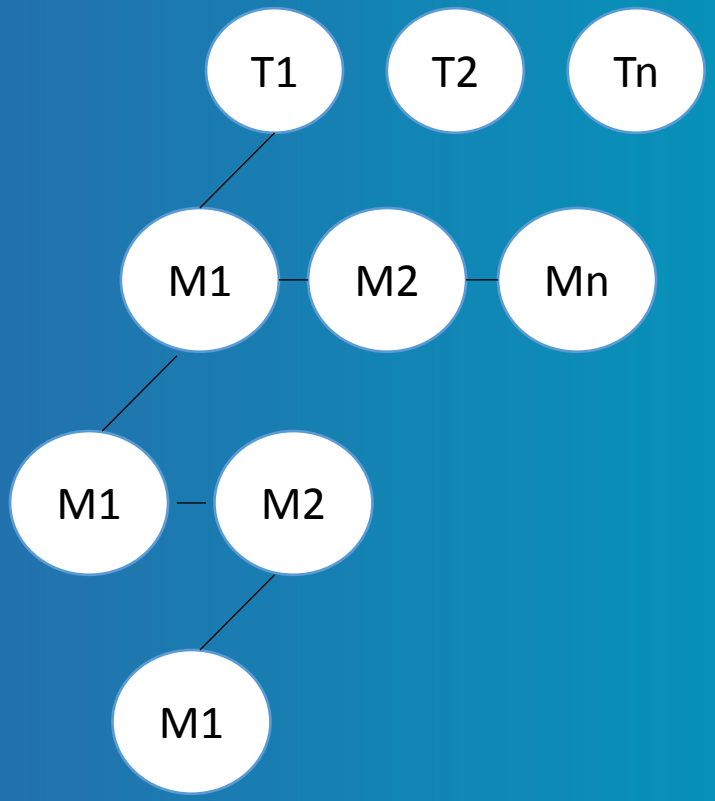
<List> callerList

3、RelationStack

<List> methodDataList

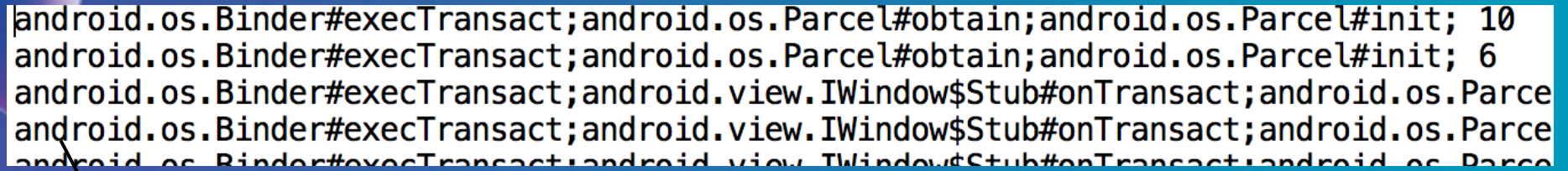
<List> threadDataList

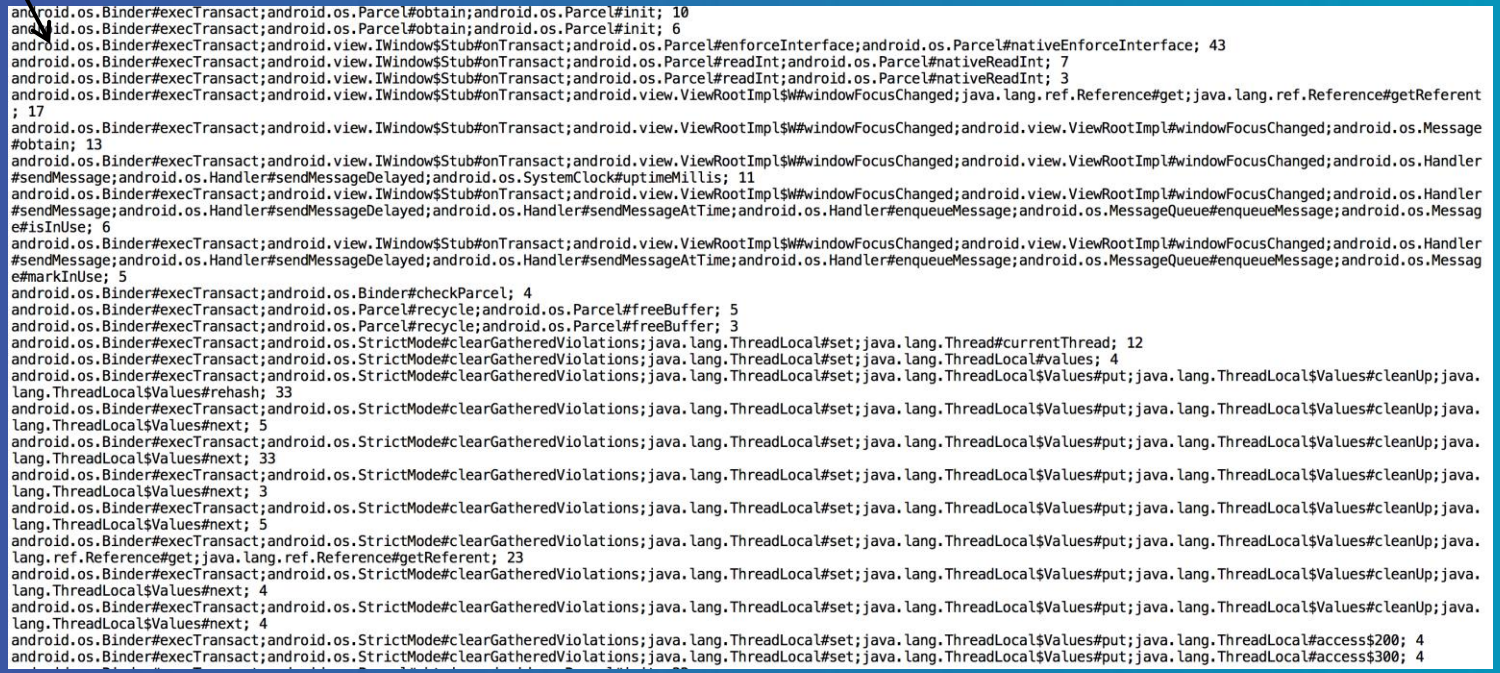
<Map> threadStackMap



```java
public void dealRecord(int tid, int mid, int action, long cpuTime, long realTime) {
    IThreadStack threadStack = threadStackTree.get(tid);
    if (threadStack == null) return;

    if (threadStack.getHelperStack().empty()) {
        IMethodStack methodStackNode = new IMethodStack();
        methodStackNode.setMid(mid);
        methodStackNode.setCpuEnterTime(cpuTime);
        methodStackNode.setRealEnterTime(realTime);

        threadStack.getCallerList().add(methodStackNode);
        if (action == ACTION_ENTER) threadStack.getHelperStack().push(methodStackNode);
    }else {
        IMethodStack methodStackPointer = threadStack.getHelperStack().peek();

        if (action == ACTION_ENTER) {
            // 不论是不是递归都到运行到该处逻辑,添加Node到tree和stack中
            IMethodStack methodStackNode = new IMethodStack();
            methodStackNode.setMid(mid);
            methodStackNode.setCpuEnterTime(cpuTime);
            methodStackNode.setRealEnterTime(realTime);

            methodStackPointer.getCallerList().add(methodStackNode);
            threadStack.getHelperStack().push(methodStackNode);
        } else {
            if (methodStackPointer.getMid() == mid) {
                methodStackPointer.setCpuExitTime(cpuTime);
                methodStackPointer.setRealExitTime(realTime);

                if (!threadStack.getHelperStack().empty()) {
                    threadStack.getHelperStack().pop();
                }
            } else {
                threadStackTree.remove(tid);
                // it may be wrong.example for pointer:
                // a(1)->b(1)->c(1),it should be c(1) or c(2) or d(1),but it is d(2),
                String error = String.format(Locale.CHINA, format: "maybe wrong:tid:%d,mid:%d", tid, mid);
                System.err.println(error);
            }
        }
    }
}
```

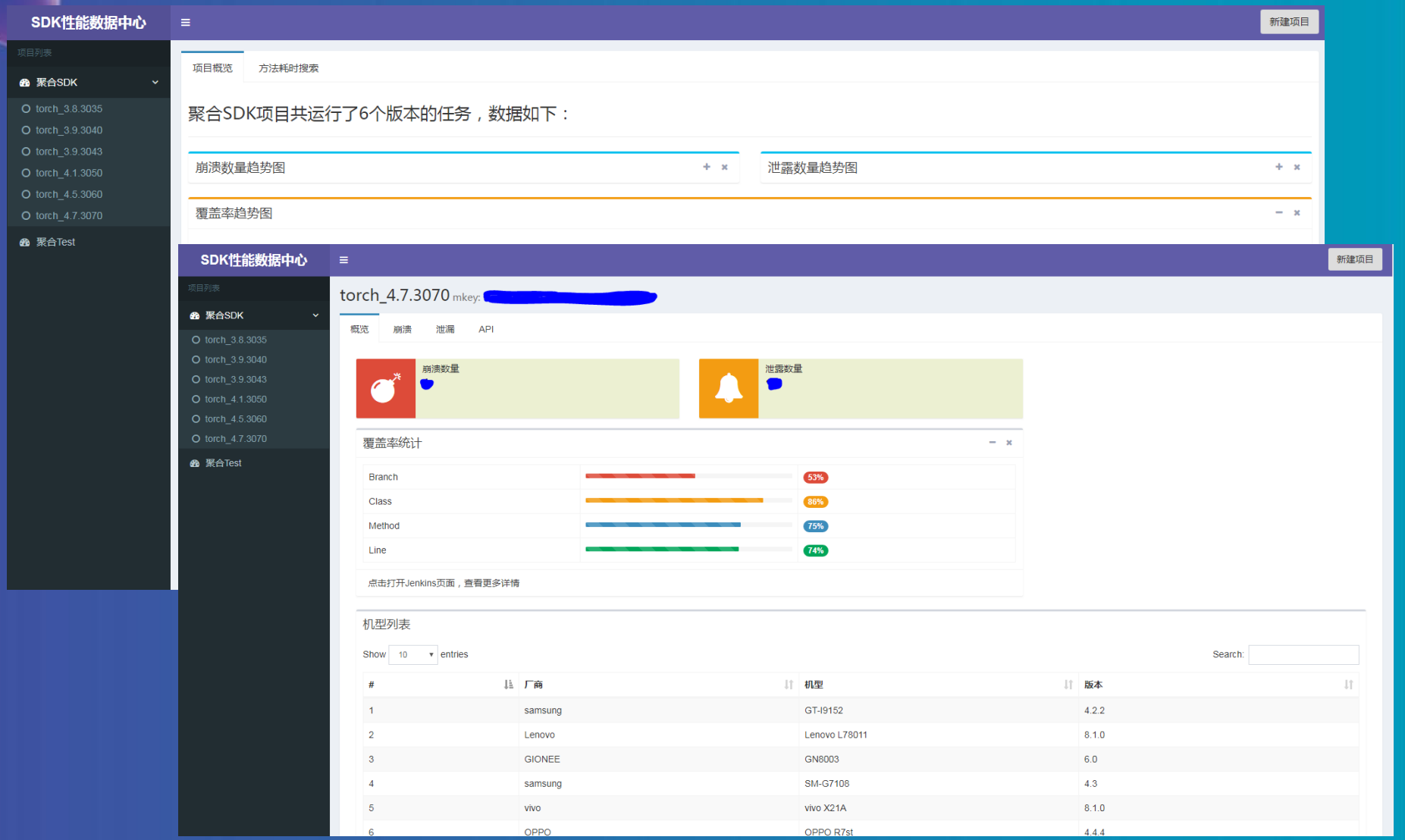
```
private void packRecordsFlameFlat(FileWriter fileWriter, ArrayList<IMethodStack> deepCallerList,
                                  ArrayList<IMethodStack> wideCallerList) throws Exception {
    if (fileWriter == null || deepCallerList == null) return;
    if (wideCallerList != null && wideCallerList.size() > 0) {
        for (IMethodStack iMethodStack : wideCallerList) {
            deepCallerList.add(iMethodStack);
            packRecordsFlameFlat(fileWriter, deepCallerList, iMethodStack.getCallerList());
            deepCallerList.remove(iMethodStack);
        }
    } else {
        deepLine.delete(0, deepLine.length());
        for (int i = 0; i < deepCallerList.size(); i++) {
            IMethodStack iMethodStack = deepCallerList.get(i);
            IMethodData iMethodData = methodDataList.get(iMethodStack.getMid());

            if (iMethodData == null) {
                deepLine.append(FLAG_UNKONW);
                deepLine.append(FLAG_CLASS);
                deepLine.append(FLAG_UNKONW);
                deepLine.append(FLAG_FUNCS);
            } else {
                deepLine.append(iMethodData.getCname());
                deepLine.append(FLAG_CLASS);
                deepLine.append(iMethodData.getMname());
                deepLine.append(FLAG_FUNCS);
            }

            if (i == deepCallerList.size() - 1) {
                deepLine.append(FLAG_SPACE);
                deepLine.append(iMethodStack.getCpuDiffTime());
                deepLine.append(FLAG_ENDLINE);
                if (iMethodStack.getCpuDiffTime() > 0) {
                    fileWriter.append(deepLine);
                    fileWriter.flush();
                }
                deepLine.delete(0, deepLine.length());
            }
        }
    }
}
```




最后通过flamegraph.pl对flat生成svg

# 数据可视化


SDK性能数据中心
新建项目
项目列表
聚合SDK
torch_3.8.3035
torch_3.9.3040
torch_3.9.3043
torch_4.1.3050
torch_4.5.3060
torch_4.7.3070
聚合Test
项目概览
方法耗时搜索
聚合SDK项目共运行了6个版本的任务，数据如下：
崩溃数量趋势图
泄露数量趋势图
覆盖率趋势图
SDK性能数据中心
新建项目
torch_4.7.3070 mkey:
概览
崩溃
泄漏
API
崩溃数量
泄露数量
覆盖率统计
Branch 53%
Class 86%
Method 75%
Line 74%
点击打开Jenkins页面，查看更多详情
机型列表
Show 10 entries
Search:
# 厂商 机型 版本
1 samsung GT-I9152 4.2.2
2 Lenovo Lenovo L78011 8.1.0
3 GIONEE GN8003 6.0
4 samsung SM-G7108 4.3
5 vivo vivo X21A 8.1.0
6 OPPO OPPO R7st 4.4.4

# 数据可视化

## 线程详情

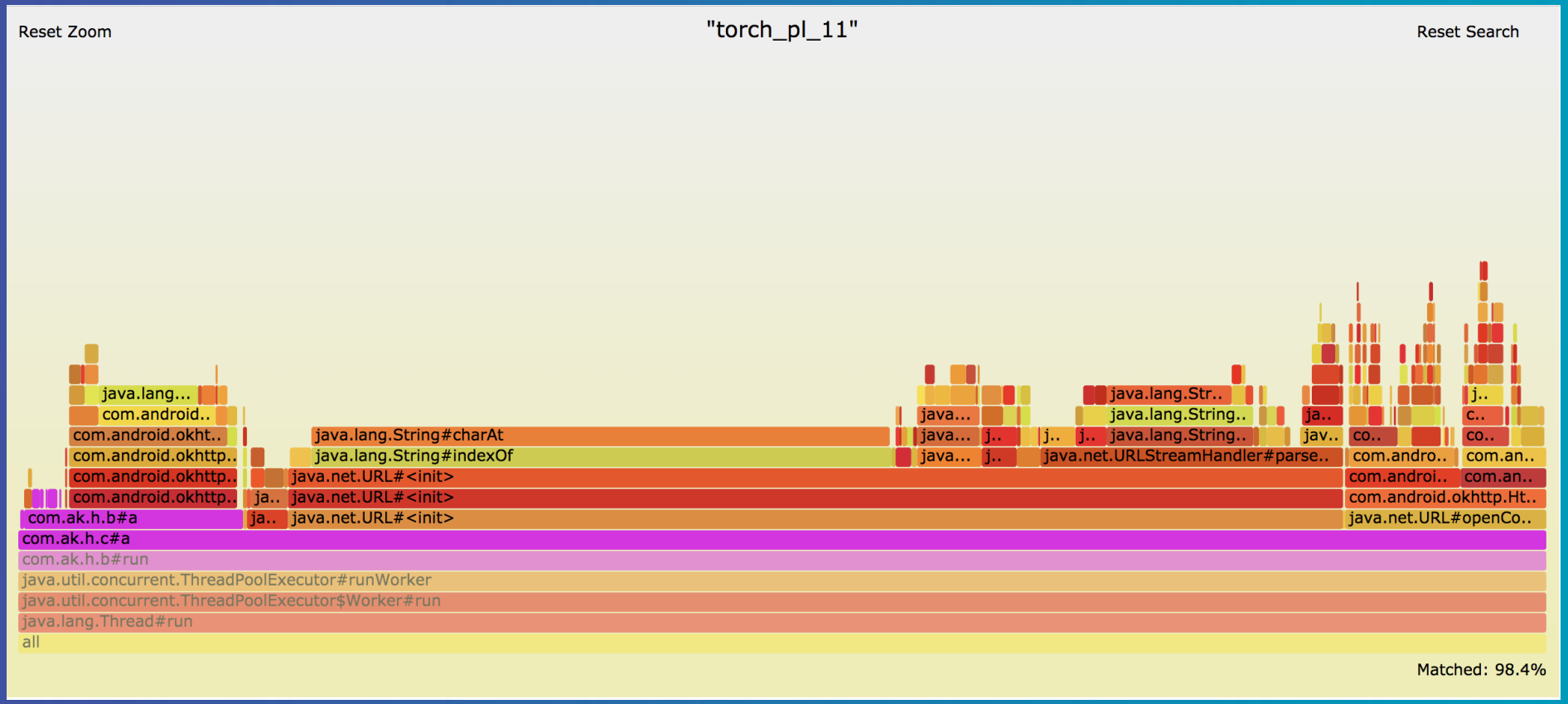
Show 10 entries Search:

| # | 线程名称 | 线程ID | SVG图片 |
|---|---|---|---|
| 1 | TaskSchedulerFo | 11136 | 点击查看 |
| 2 | [illegible]ad | 11137 | 点击查看 |
| 3 | AudioThread | 11139 | 点击查看 |
| 4 | TaskSchedulerSi | 11140 | 点击查看 |
| 5 | TaskSchedulerFo | 11141 | 点击查看 |
| 6 | TaskSchedulerFo | 11143 | 点击查看 |
| 7 | [illegible]11 | 11784 | 点击查看 |
| 8 | GAC_Executor[0] | 11144 | 点击查看 |
| 9 | CleanupReference | 11145 | 点击查看 |
| 10 | [illegible]85 | 12171 | 点击查看 |

Showing 1 to 10 of 84 entries

Previous 1 2 3 4 5 … 9 Next

# 数据可视化


Reset Zoom
"torch_pl_11"
Reset Search
java.lang...
com.android..
com.android.okht..
com.android.okhttp..
com.android.okhttp..
com.android.okhttp..
com.ak.h.b#a
com.ak.h.c#a
com.ak.h.b#run
java.util.concurrent.ThreadPoolExecutor#runWorker
java.util.concurrent.ThreadPoolExecutor$Worker#run
java.lang.Thread#run
all
java.lang.String#charAt
java.lang.String#indexOf
java.net.URL#<init>
java.net.URL#<init>
java.net.URL#<init>
java.lang.Str..
java.lang.String..
java.net.URLStreamHandler#parse..
com.andro..
com.androi..
com.android.okhttp.Ht..
java.net.URL#openCo..
Matched: 98.4%