Systrace 使用说明

文件标识：RK-SM-YF-151

发布版本：V1.0.1

日期：2021-03-02

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

产品版本

芯片名称	内核版本
全系列	通用

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	陈谋春	2017-12-25	初始版本
V1.0.1	黄莹	2021-03-02	修改格式

目录

1. 介绍

Systrace 是目前 Android 上最主要的性能调试手段，有以下优点：

- 完全免费，安装和使用都比较简便
- 由于不需要在设备端运行监控程序，所以不需要 root 权限[1]
- 界面友好

同时也有一些缺点：

- 基于 tracepoint，所以只会收集你加过 trace 的函数信息，Android 在大部分模块的重要函数里都加了 trace 了，所以大部分情况下还是够用，同时 Android 也提供了几个函数方便添加自己的 trace。
- 看不到 pmu 计数器的信息，也看不到 gpu 和 memory 的信息（理论上内核驱动如果定时收集这些信息并加到 trace 里，systrace 应该也能看到）

2. 用法

为了更方便介绍 Systrace，我这里举一个实际的性能分析例子：fishtank 在 1000 只鱼的情况下帧率很低

2.1 准备工作

获取 systrace 有三种方式：

1. 下载Android_Sdk_Tool

   路径：/path_to_sdk/platform-tools/systrace

2. 下载Android Studio

   提供了图形化抓取功能，实际上也是嗲用 sdk 里的 systrace

3. 直接用 Android 源码里的

   路径：/path_to_android/external/chromium-trace

2.2 抓取数据

Systrace 的命令格式：

```
$ cd external/chromium-trace/
$ python ./systrace.py -h
Usage: systrace.py [options] [category1 [category2 ...]]

Example: systrace.py -b 32768 -t 15 gfx input view sched freq

Options:
  -h, --help            show this help message and exit
```

```
  -o FILE               write trace output to FILE
  -t N, --time=N        trace for N seconds
  -l, --list-categories
                        list the available categories and exit
  -j, --json            write a JSON file
  --link-assets         (deprecated)
  --from-file=FROM_FILE
                        read the trace from a file (compressed) rather
                        thanrunning a live trace
  --asset-dir=ASSET_DIR
                        (deprecated)
  -e DEVICE_SERIAL_NUMBER, --serial=DEVICE_SERIAL_NUMBER
                        adb device serial number
  --target=TARGET       chose tracing target (android or linux)
  --timeout=TIMEOUT     timeout for start and stop tracing (seconds)
  --collection-timeout=COLLECTION_TIMEOUT
                        timeout for data collection (seconds)

Atrace and Ftrace options:
  -b N, --buf-size=N  use a trace buffer size  of N KB
  --no-fix-threads    don't fix missing or truncated thread names
  --no-fix-tgids      Do not run extra commands to restore missing thread to
                      thread group id mappings.
  --no-fix-circular   don't fix truncated circular traces

Atrace options:
  --atrace-categories=ATRACE_CATEGORIES
                      Select atrace categories with a comma-delimited list,
                      e.g. --atrace-categories=cat1,cat2,cat3
  -k KFUNCS, --ktrace=KFUNCS
                      specify a comma-separated list of kernel functions to
                      trace
  -a APP_NAME, --app=APP_NAME
                      enable application-level tracing for comma-separated
                      list of app cmdlines
  --no-compress       Tell the device not to send the trace data in
                      compressed form.
  --boot              reboot the device with tracing during boot enabled.The
                      report is created by hitting Ctrl+C after the
                      devicehas booted up.

Battor trace options:
  --battor-categories=BATTOR_CATEGORIES
                      Select battor categories with a comma-delimited list,
                      e.g. --battor-categories=cat1,cat2,cat3
  --hubs=HUB_TYPES    List of hub types to check for for BattOr mapping.
                      Used when updating mapping file.
  --serial-map=SERIAL_MAP
                      File containing pregenerated map of phone serial
                      numbers to BattOr serial numbers.
  --battor_path=BATTOR_PATH
                      specify a BattOr path to use
  --update-map        force update of phone-to-BattOr map
  --battor            Use the BattOr tracing agent.

Ftrace options:
  --ftrace-categories=FTRACE_CATEGORIES
                      Select ftrace categories with a comma-delimited list,
```

```
                          e.g. --ftrace-categories=cat1,cat2,cat3
```

Systrace 支持的 atrace 类别有：

```
adb root
python ./systrace.py -l
         gfx - Graphics
       input - Input
        view - View System
     webview - WebView
          wm - Window Manager
          am - Activity Manager
          sm - Sync Manager
       audio - Audio
       video - Video
      camera - Camera
         hal - Hardware Modules
         app - Application
         res - Resource Loading
      dalvik - Dalvik VM
          rs - RenderScript
      bionic - Bionic C Library
       power - Power Management
          pm - Package Manager
          ss - System Server
    database - Database
     network - Network
       sched - CPU Scheduling
        freq - CPU Frequency
        idle - CPU Idle
        load - CPU Load
  memreclaim - Kernel Memory Reclaim
  binder_driver - Binder Kernel driver
  binder_lock - Binder global lock trace

NOTE: more categories may be available with adb root
```

Note: ==有些事件需要设备的 root 权限才能操作，所以最好先切到 root 权限==

除了支持 Android 在 ftrace 基础上扩展的 atrace，Systrace 也是支持 kernel 原生的 ftrace 的，还支持单独抓取某个 kernel 函数，当然前提是这个函数本身有 tracepoint，具体可以参见上面的命令帮助信息。还可以直接用 trace 文件做输入，这种离线分析功能应该在分析 Android 引导过程的时候比较有用。

在抓取前要先大致确定这个场景涉及到哪些模块，再回到我们这次要分析的场景是：浏览器跑 fishtank 中开启 1000 只鱼的时候帧率很低；第一时间能想到的模块有：gfx webview sched freq load workq disk

先在设备上重现问题，然后在 host 端执行如下命令：

```
cd external/chromium-trace
python ./systrace.py -t 10 -o fishtank.html gfx webview sched freq load workq
disk
```

这个 fishtank.html 即我们抓到的数据。为了方便和本文对照，我上传到网盘了。

3. 分析

分析数据需要 chrome 浏览器，版本最好要新一些，太旧可能会有兼容问题，因为这个 html 并不符合 w3c 的标准。

用 chrome 打开以后，界面如下：

左列是抓取的线程名或 trace 名，既然是绘制问题，我们第一个要看肯定是绘制的线程，Android 5.0 以前是在 ui 线程做绘制的，以后的版本都是在 render 线程做绘制，所以我们先拉到 render 线程，可以看到如下：

点击右边的有四个按键，分别对应四种模式如下：

先用”时间线“模式拉个 1s 左右的时间线，然后切到“选择”模式，选择这段时间线内 render 线程的区域，会自动在下方列出这个区域的函数统计：

可以看到帧率确实很低，这段时间的绘制只有 9 次（drawgl 次数），平均耗时 29ms，平均间隔是 111ms，所以主要原因是绘制间隔太大。继续往下分析就要根据浏览器的渲染模型了，我们知道 chromium 里是由光栅化和 canvas 线程完成实际绘制的（内部叫 paint），而 ui 线程或 render 线程来完成贴图（内部叫 draw）。因为这个网页用的 canvas，所以我们先用“时间线”模式拉出绘制间隔，然后顺着时间线往下找绘制线程如下：

可以看到最近这一次的绘制耗时 124ms，拉开看一下具体耗时：

刚好有 1000 个绘制，这里会不会就是那 1000 只鱼，通过查看网页源码，可以确认：

javascript 是单线程运行的，所以这里无法用到多核，javascript worker 技术是让 js 跑多线程，但是这个网页并没有用到这个技术。

要解决这个问题，要么改网页代码，启用 javascript worker 技术，这样应该能让帧率提升不少；还有一种办法就是启用 chromium 的 gpu 光栅化技术，即不调用 skia 做 2d 绘制，直接用 gpu 来绘制，但是目前这个技术缺陷较多，会导致某些场景下闪屏。

1. 有一些kernel的trace需要root权限操作trace节点↩