Android内存泄露分析简要思路

泡在网上的日子 发表于 2015-06-14 12:09 第 593 次阅读 内存泄露 , MAT

2

来源 http://blog.csdn.net/ljphhj/article/details/44966023

工作中遇到挺多需要分析内存泄露问题的情况，现在大致简要写下思路，等之后时间相对比较充裕再进行补充。

1.明白内存泄露的判断依据?

个人总结为：持续增加，只增不减!

理解一下这8个字，配合几个命令和工具来确定一下你的应用是否存在内存泄露问题，这是很关键的，如果一开始就判断错误了，那么没有继续往下进行的理由。

命令如下:

adb shell

dumpsys meminfo 应用包名 [当然，比较粗略地话，可以用adb shell procrank]

这时候你可以看到一个内存使用情况表

```
shell@cancro:/ $ dumpsys meminfo com.android.mms
dumpsys meminfo com.android.mms
Applications Memory Usage (kB):
Uptime: 64975704 Realtime: 243757046

** MEMINFO in pid 2786 [com.android.mms] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap        0        0        0        0    12592     9288     2791
  Dalvik Heap    18156    18048        0    16392    40676    32361     8315
 Dalvik Other     5082     5028        0      324
        Stack      264      264        0       20
      Cursor        4        0        0        0
      Ashmem      128      128        0        0
   Other dev     3142     3048        4        0
    .so mmap     3793     1500     1440     1636
   .jar mmap        1        0        0        0
   .apk mmap      177        0       32        0
   .ttf mmap      302        0       36        0
   .dex mmap     1962      208     1160       28
  Other mmap       16        4        0        0
     Unknown     9182     9176        0      244
       TOTAL    42209    37404     2672    18644    53268    41649    11106

 Objects
               Views:      696         ViewRootImpl:        3
         AppContexts:       13           Activities:        4
              Assets:        5        AssetManagers:        5
      Local Binders:      131        Proxy Binders:       48
   Death Recipients:        2
    OpenSSL Sockets:        2

 SQL
         MEMORY_USED:        0
  PAGECACHE_OVERFLOW:        0          MALLOC_SIZE:        0
```

而我们首先关注的是TOTAL这个值，譬如你会执行某个脚本，或者Monkey在不断操作你的应用，你可以隔半个小时观察一次TOTAL的情况，如果持续增加，那么十有八九是有内存泄露的问题了，但是也并不是绝对，这时候你就要配合上DDMS工具来做分析了。（DDMS里面有一项Heap，里面可以Cause GC，如果你发现你执行了GC之后，依旧内存没有什么太大的减少，那么恭喜你，这部分内存泄露了。）

2.根据第一步所阐述的一个情况，大概可以判断出是否有内存泄露，那么接着需要做什么呢?

如果内存泄露，那么我们该做些什么呢?

思路：必定我们想要找到泄露点，那么如何找到?

（1）. 你需要抓取hprof文件，这个文件可以呈现出一些内存的使用情况，不过切记，你别清除了进程再来抓这个文件，然后配合着MAT把这个hprof文件分析了

（MAT分析hprof文件的这方面知识网上太齐全了，读者们可以自己去学习一下哈，这里就不累赘）

（2）.如果MAT分析出来的东东，你实在看不出什么猫腻，那么再次恭喜你，你中大招了。

这时候你要继续观察dumpsys meminfo 包名， 输出的结果信息，关注点放在 UnKnown那一行 和 Native Heap 那一行，关注Heap Alloc 或者 Pss Total,

如果你的总TOTAL一直再增加，但是是由于这两行的增加，那么这个问题你不需要再继续在MAT上花时间了，因为这种内存泄露问题，出在Native层（C）

那么你需要去找你程序中使用到JNI的地方，so库或者其他一些特殊调用上，分析它们是否可能造成内存泄露问题。

（3）当然兴许你依旧没有头绪，那么没关系，另一个命令就是为了你而存在的，（首先某个应用的PID号，用dumpsys meminfo 包名，那边已经可以查到）

譬如我上面那个mms, PID号为2786， 接着adb shell showmap -a PID号 (adb shell showmap -a 2786)

然后根据结果[....]这的信息，在去google上面找关键字， 譬如：[anon] bash的堆

（4）当你最终还是不知道是由哪边的.so库引起的话，你可以查看下Native Heap的内存分配情况，这时候你依旧需要借助DDMS，

需要先执行以下命令：

adb shell setprop libc.debug.malloc 1

adb shell stop

adb shell start

然后你还需要改一下eclipse中的配置参数值【因为如果你不配置的话，你的DDMS打开默认是看不到Native Heap那个Tab项的】

在ddms.cfg文件（实在找不到的话，就用Everything搜索下吧）最后增加一行native=true并save。ddms.cfg位于c:\Users\xxx\.android目录下。

在Device中选择好你要的应用的包名项，然后按下Snapshot按钮， 就可以观察到Native Heap的使用情况了，然后反复执行脚本，再观察观察，你会找到你需要的东西的。

（5）内存泄露的问题可能很复杂，但是有的时候有了这些技巧，相对更重要。所以第（5）小点，额外再提供一些其他的命令技巧，帮助查找另外的情况出现内存泄露的问题。

1. adb shell dumpsys activity 查询activity栈的情况（曾遇到过，就是activity一直堆积，导致的内存溢出OOM）
2. adb shell dumpsys cpuinfo
3. adb shell dumpsys battery

如果你在这方面有自己其他的一些方法，那么也不妨留言跟大家分享下！

文件：/proc/meminfo

/proc/buddyinfo

简单粗暴就直接通过: showmap 命令 多次，然后对比一下，看哪个增长得最明显。