**OOM 问题**

1. OOM产生的原因

Java进程在虚拟机中分配对象的堆大小是固定的，当分配的对象超过free堆的大小，虚拟机就会抛出OOM异常。

Java虚拟机进程堆的大小的查看方法：adb shell getprop dalvik.vm.heapsize

抛出OOM异常的过程：

/art/runtime/GC/Heap.cc

/art/runtime/GC/Thread.cc

HTC 对OOM问题的调试在art中添加了DumpHeapInfo流程，会在/data/misc/hprof目录下生成hprof文件。

1. OOM分析的方法

（1）.使用android-sdk/tools/hprof-conv 将生成的hprof文件转换一下。

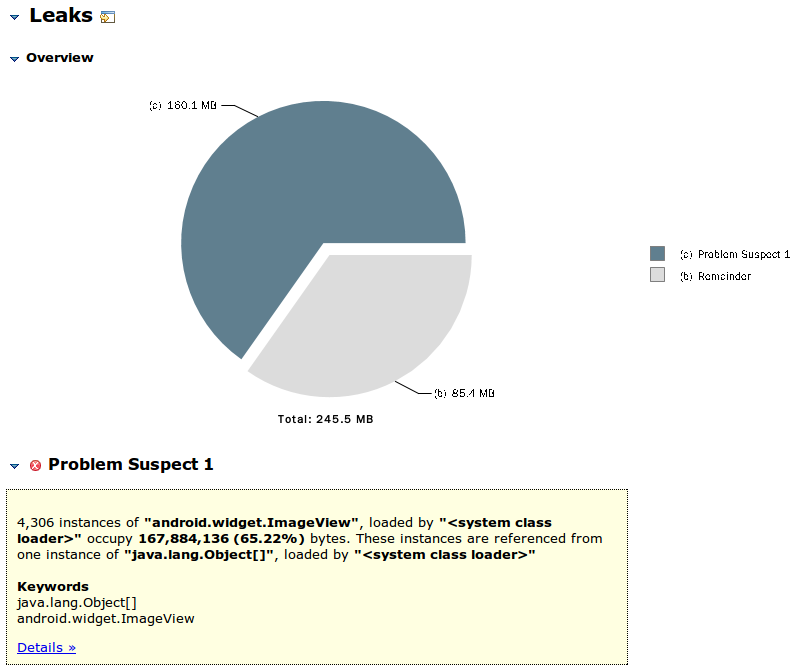
Sample: hprof-conv heap-dump-1461095660-pid4344.hprof convert.hprof

(2).使用MAT工具打开hprof文件进行分析。

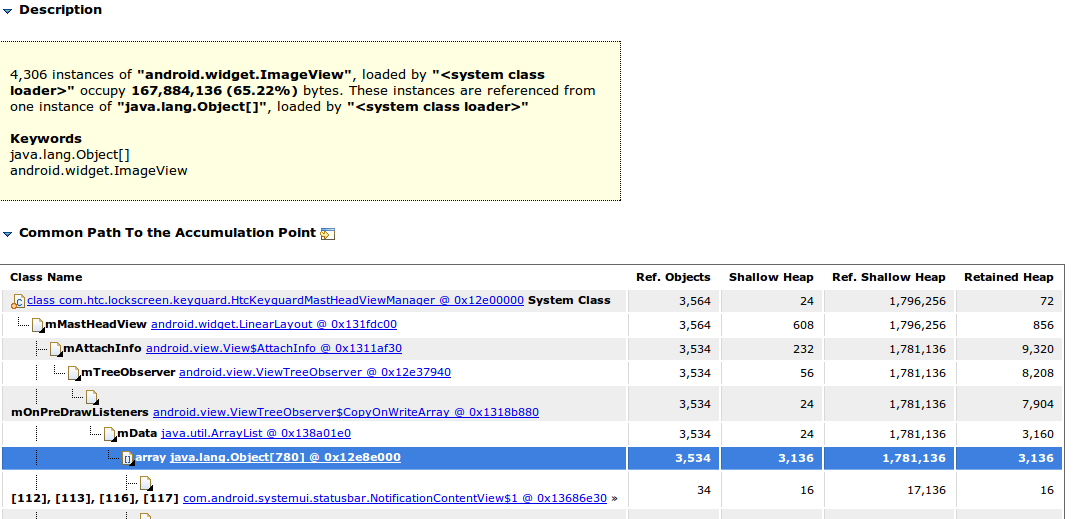
工具路径：<ftp://10.13.11.154/Soft/601_tools/MemoryAnalyzer-1.4.0.20140604-linux.gtk.x86_64.zip>

点击leak suspects查看leak的对象有哪些，结合代码分析leak的对象是否有正常被释放。

Sample：



点击Details查看对象引用详细的信息：



发现在ViewTreeObserver的mOnPreDrawListeners容器中保存了大量的NotificationContentView对象导致的泄漏。

结合代码分析是NotificationContentView 没有进行removeOnPreDrawListener导致的该问题。