|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D:\02-资料分类\04-文档模板\扉页图1-new.jpg | | | | |
|  | AndroidR\_Java内存泄漏调试指南 | | |  |
| 文档版本 V1.1 |  | |
| 发布日期 2020-08-17 |  | |
|  | | | | |
|  | 紫光展锐科技有限公司 | |  |  |

|  |
| --- |
| 版权所有 © 紫光展锐科技有限公司。保留一切权利。  本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。  请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。 |



|  |
| --- |
| 紫光展锐科技有限公司 |

前 言

概述

AddressSanitizer用于检测android native层的内存覆盖，本文档讲述其使用方法。

读者对象

本文档适用于需要在android native层调试的内存覆盖的同事，及其他对此感兴趣的同事。

缩略语

|  |  |  |
| --- | --- | --- |
| 缩略语 | 英文全名 | 中文解释 |
|  |  |  |
|  |  |  |

变更信息

| 文档版本 | 发布日期 | 作者 | 修改说明 |
| --- | --- | --- | --- |
| V1.0 | 2018-11-27 | Jasson.zhang | 初稿 |
| V1.1 | 2020-08-17 | Jasson.zhang | 在AndroidR上使用修订 |
| V |  |  |  |
| V |  |  |  |

关键字

Java、内存泄漏。

目 录

[1 简介 4](#_Toc46767355)

[1.1 java内存泄漏分析流程 4](#_Toc46767356)

[1.2 Java内存泄漏的判别 5](#_Toc46767357)

[1.2.1 GC时输出进程空间占用 5](#_Toc46767358)

[1.2.2 log 6](#_Toc46767359)

[1.3 hprof文件的获取 8](#_Toc46767360)

[1.3.1 通过DDMS抓取hprof 8](#_Toc46767361)

[1.3.2 通过命令行抓取hprof 9](#_Toc46767362)

[1.4 hprof文件的分析转换 10](#_Toc46767363)

[1.4.1 方法 10](#_Toc46767364)

[1.4.2 举例： 10](#_Toc46767365)

[2 Mat工具使用简介 11](#_Toc46767366)

[2.1 LeakSuspects 11](#_Toc46767367)

[2.2 histogram 12](#_Toc46767368)

[2.2.1 差异比较 13](#_Toc46767369)

[2.2.2 其他 14](#_Toc46767370)

[3 问题分析中的应用 15](#_Toc46767371)

[3.1 例一 15](#_Toc46767372)

[3.2 例二 17](#_Toc46767373)

# 简介

之前对native层内存泄漏的抓取分析方法学习并总结过文档，最近在工作中遇到java层内存泄漏的问题，对java层内存泄漏的抓取和分析定位进行了学习整理，本文对其进行介绍。

## java内存泄漏分析流程

我们知道android上层内存泄漏涉及java层泄漏和native层的泄漏，对于java层的泄漏，一般是：

* 首先，通过分析其代码中的log判断其可能存在内存泄漏
* 然后，根据其复现步骤抓取hprof文件进行对比分析，判断泄漏的对象
* 之后，通过分析代码或加log定位具体泄漏点

## Java内存泄漏的判别

我们可以通过查看log中的GC值来检查其是否存在内存泄漏的可能。

我们以 system\_server 进程为例，检查system\_server进程是否存在内存泄漏的可能，

我们找到system\_server进程GC时的log，查看GC后占用的内存空间。

### GC时输出进程空间占用

我们首先来看一下进程GC时打印log的代码：

其对应代码位于art中：

<http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc>

|  |
| --- |
| [2698](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2698)  **void** [**Heap**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#Heap)::**[LogGC](http://10.0.1.79:8081/s?refs=LogGC&project=sprdroidr_trunk)**([GcCause](http://10.0.1.79:8081/s?defs=GcCause&project=sprdroidr_trunk) **[gc\_cause](http://10.0.1.79:8081/s?refs=gc_cause&project=sprdroidr_trunk)**, **[collector](http://10.0.1.79:8081/s?refs=collector&project=sprdroidr_trunk)**::[GarbageCollector](http://10.0.1.79:8081/s?defs=GarbageCollector&project=sprdroidr_trunk)\* [collector](http://10.0.1.79:8081/s?defs=collector&project=sprdroidr_trunk)) {  [2699](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2699)   **const** [size\_t](http://10.0.1.79:8081/s?defs=size_t&project=sprdroidr_trunk) **[duration](http://10.0.1.79:8081/s?refs=duration&project=sprdroidr_trunk)** = [GetCurrentGcIteration](http://10.0.1.79:8081/s?defs=GetCurrentGcIteration&project=sprdroidr_trunk)()->[GetDurationNs](http://10.0.1.79:8081/s?defs=GetDurationNs&project=sprdroidr_trunk)();  [2700](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2700)   **const** [std](http://10.0.1.79:8081/s?defs=std&project=sprdroidr_trunk)::[vector](http://10.0.1.79:8081/s?defs=vector&project=sprdroidr_trunk)<[uint64\_t](http://10.0.1.79:8081/s?defs=uint64_t&project=sprdroidr_trunk)>& **[pause\_times](http://10.0.1.79:8081/s?refs=pause_times&project=sprdroidr_trunk)** = [GetCurrentGcIteration](http://10.0.1.79:8081/s?defs=GetCurrentGcIteration&project=sprdroidr_trunk)()->[GetPauseTimes](http://10.0.1.79:8081/s?defs=GetPauseTimes&project=sprdroidr_trunk)();  [2701](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2701)   // Print the GC if it is an explicit GC (e.g. Runtime.gc()) or a slow GC  [2702](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2702)   // (mutator time blocked >= long\_pause\_log\_threshold\_).  [2703](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2703)   **bool** **[log\_gc](http://10.0.1.79:8081/s?refs=log_gc&project=sprdroidr_trunk)** = [**kLogAllGCs**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#kLogAllGCs) || [gc\_cause](http://10.0.1.79:8081/s?defs=gc_cause&project=sprdroidr_trunk) == [kGcCauseExplicit](http://10.0.1.79:8081/s?defs=kGcCauseExplicit&project=sprdroidr_trunk);  [2704](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2704)   **if** (!**[log\_gc](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "log_gc)** && [**CareAboutPauseTimes**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#CareAboutPauseTimes)()) {  [2705](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2705)   // GC for alloc pauses the allocating thread, so consider it as a pause.  [2706](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2706)   [**log\_gc**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#log_gc) = [duration](http://10.0.1.79:8081/s?defs=duration&project=sprdroidr_trunk) > [long\_gc\_log\_threshold\_](http://10.0.1.79:8081/s?defs=long_gc_log_threshold_&project=sprdroidr_trunk) ||  [2707](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2707)   ([gc\_cause](http://10.0.1.79:8081/s?defs=gc_cause&project=sprdroidr_trunk) == [kGcCauseForAlloc](http://10.0.1.79:8081/s?defs=kGcCauseForAlloc&project=sprdroidr_trunk) && [duration](http://10.0.1.79:8081/s?defs=duration&project=sprdroidr_trunk) > [long\_pause\_log\_threshold\_](http://10.0.1.79:8081/s?defs=long_pause_log_threshold_&project=sprdroidr_trunk));  [2708](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2708)   **for** ([uint64\_t](http://10.0.1.79:8081/s?defs=uint64_t&project=sprdroidr_trunk) **[pause](http://10.0.1.79:8081/s?refs=pause&project=sprdroidr_trunk)** : [**pause\_times**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_times)) {  [2709](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2709)   [**log\_gc**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#log_gc) = [**log\_gc**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#log_gc) || [pause](http://10.0.1.79:8081/s?defs=pause&project=sprdroidr_trunk) >= [long\_pause\_log\_threshold\_](http://10.0.1.79:8081/s?defs=long_pause_log_threshold_&project=sprdroidr_trunk);  [2710](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2710)   }  [2711](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2711)   }  [2712](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2712)   **if** (**[log\_gc](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "log_gc)**) {  [2713](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2713)   **const** [size\_t](http://10.0.1.79:8081/s?defs=size_t&project=sprdroidr_trunk) **[percent\_free](http://10.0.1.79:8081/s?refs=percent_free&project=sprdroidr_trunk)** = [**GetPercentFree**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#GetPercentFree)();  [2714](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2714)   **const** [size\_t](http://10.0.1.79:8081/s?defs=size_t&project=sprdroidr_trunk) **[current\_heap\_size](http://10.0.1.79:8081/s?refs=current_heap_size&project=sprdroidr_trunk)** = [GetBytesAllocated](http://10.0.1.79:8081/s?defs=GetBytesAllocated&project=sprdroidr_trunk)();  [2715](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2715)   **const** [size\_t](http://10.0.1.79:8081/s?defs=size_t&project=sprdroidr_trunk) **[total\_memory](http://10.0.1.79:8081/s?refs=total_memory&project=sprdroidr_trunk)** = [**GetTotalMemory**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#GetTotalMemory)();  [2716](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2716)   [std](http://10.0.1.79:8081/s?defs=std&project=sprdroidr_trunk)::[ostringstream](http://10.0.1.79:8081/s?defs=ostringstream&project=sprdroidr_trunk) **[pause\_string](http://10.0.1.79:8081/s?refs=pause_string&project=sprdroidr_trunk)**;  [2717](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2717)   **for** ([size\_t](http://10.0.1.79:8081/s?defs=size_t&project=sprdroidr_trunk) **[i](http://10.0.1.79:8081/s?refs=i&project=sprdroidr_trunk)** = 0; [i](http://10.0.1.79:8081/s?defs=i&project=sprdroidr_trunk) < [**pause\_times**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_times).[size](http://10.0.1.79:8081/s?defs=size&project=sprdroidr_trunk)(); ++[i](http://10.0.1.79:8081/s?defs=i&project=sprdroidr_trunk)) {  [2718](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2718)   [**pause\_string**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_string) << [PrettyDuration](http://10.0.1.79:8081/s?defs=PrettyDuration&project=sprdroidr_trunk)(([**pause\_times**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_times)[[i](http://10.0.1.79:8081/s?defs=i&project=sprdroidr_trunk)] / 1000) \* 1000)  [2719](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2719)   << (([i](http://10.0.1.79:8081/s?defs=i&project=sprdroidr_trunk) != [**pause\_times**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_times).[size](http://10.0.1.79:8081/s?defs=size&project=sprdroidr_trunk)() - 1) ? "," : "");  [2720](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2720)   }  [2721](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2721)   [LOG](http://10.0.1.79:8081/s?defs=LOG&project=sprdroidr_trunk)([INFO](http://10.0.1.79:8081/s?defs=INFO&project=sprdroidr_trunk)) << [gc\_cause](http://10.0.1.79:8081/s?defs=gc_cause&project=sprdroidr_trunk) << " " << [collector](http://10.0.1.79:8081/s?defs=collector&project=sprdroidr_trunk)->[GetName](http://10.0.1.79:8081/s?defs=GetName&project=sprdroidr_trunk)()  [2722](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2722)   << " GC freed " << [current\_gc\_iteration\_](http://10.0.1.79:8081/s?defs=current_gc_iteration_&project=sprdroidr_trunk).[GetFreedObjects](http://10.0.1.79:8081/s?defs=GetFreedObjects&project=sprdroidr_trunk)() << "("  [2723](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2723)   << [PrettySize](http://10.0.1.79:8081/s?defs=PrettySize&project=sprdroidr_trunk)([current\_gc\_iteration\_](http://10.0.1.79:8081/s?defs=current_gc_iteration_&project=sprdroidr_trunk).[GetFreedBytes](http://10.0.1.79:8081/s?defs=GetFreedBytes&project=sprdroidr_trunk)()) << ") AllocSpace objects, "  [2724](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2724)   << [current\_gc\_iteration\_](http://10.0.1.79:8081/s?defs=current_gc_iteration_&project=sprdroidr_trunk).[GetFreedLargeObjects](http://10.0.1.79:8081/s?defs=GetFreedLargeObjects&project=sprdroidr_trunk)() << "("  [2725](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2725)   << [PrettySize](http://10.0.1.79:8081/s?defs=PrettySize&project=sprdroidr_trunk)([current\_gc\_iteration\_](http://10.0.1.79:8081/s?defs=current_gc_iteration_&project=sprdroidr_trunk).[GetFreedLargeObjectBytes](http://10.0.1.79:8081/s?defs=GetFreedLargeObjectBytes&project=sprdroidr_trunk)()) << ") LOS objects, "  [2726](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2726)   << [**percent\_free**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#percent_free) << "% free, " << [PrettySize](http://10.0.1.79:8081/s?defs=PrettySize&project=sprdroidr_trunk)([**current\_heap\_size**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#current_heap_size)) << "/"  [2727](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2727)   << [PrettySize](http://10.0.1.79:8081/s?defs=PrettySize&project=sprdroidr_trunk)([**total\_memory**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#total_memory)) << ", " << "paused " << [**pause\_string**](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5#pause_string).[str](http://10.0.1.79:8081/s?defs=str&project=sprdroidr_trunk)()  [2728](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2728)   << " total " << [PrettyDuration](http://10.0.1.79:8081/s?defs=PrettyDuration&project=sprdroidr_trunk)(([duration](http://10.0.1.79:8081/s?defs=duration&project=sprdroidr_trunk) / 1000) \* 1000);  [2729](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2729)   [VLOG](http://10.0.1.79:8081/s?defs=VLOG&project=sprdroidr_trunk)([heap](http://10.0.1.79:8081/s?defs=heap&project=sprdroidr_trunk)) << [Dumpable](http://10.0.1.79:8081/s?defs=Dumpable&project=sprdroidr_trunk)<[TimingLogger](http://10.0.1.79:8081/s?defs=TimingLogger&project=sprdroidr_trunk)>(\*[current\_gc\_iteration\_](http://10.0.1.79:8081/s?defs=current_gc_iteration_&project=sprdroidr_trunk).[GetTimings](http://10.0.1.79:8081/s?defs=GetTimings&project=sprdroidr_trunk)());  [2730](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2730)   }  [2731](http://10.0.1.79:8081/xref/sprdroidr_trunk/art/runtime/gc/heap.cc?r=4f9d62b5" \l "2731)  } |

我们可以看到log输出中 current\_heap\_size / total\_memory 分别对应GC后的堆大小及占用空间

### log

出问题时的log中，进程 system\_server 进程占用的内存空间。

|  |
| --- |
| 12-08 17:38:15.205 756 787 I system\_server: Background concurrent copying GC freed 493352(23MB) AllocSpace objects, 66(1960KB) LOS objects, 10% free, 207MB/231MB, paused 421us total 4.223s  12-08 17:39:54.902 756 787 I system\_server: Background concurrent copying GC freed 523315(22MB) AllocSpace objects, 8(352KB) LOS objects, 10% free, 211MB/235MB, paused 5.801ms total 1.153s  12-08 17:42:17.043 756 787 I system\_server: Background concurrent copying GC freed 666167(32MB) AllocSpace objects, 14(600KB) LOS objects, 10% free, 207MB/231MB, paused 413us total 2.623s  12-08 17:43:40.164 756 787 I system\_server: Background concurrent copying GC freed 482481(29MB) AllocSpace objects, 17(540KB) LOS objects, 10% free, 205MB/229MB, paused 340us total 3.489s  12-08 17:46:14.000 756 787 I system\_server: Background concurrent copying GC freed 563440(30MB) AllocSpace objects, 2(40KB) LOS objects, 10% free, 211MB/235MB, paused 333us total 2.653s |

正常log中，进程 system\_server 进程占用的内存空间。

|  |
| --- |
| 12-12 21:22:41.739 736 786 I system\_server: Explicit concurrent copying GC freed 113168(6MB) AllocSpace objects, 20(520KB) LOS objects, 42% free, 26MB/46MB, paused 422us total 2.834s  12-12 21:22:49.051 736 786 I system\_server: Explicit concurrent copying GC freed 44551(2MB) AllocSpace objects, 15(428KB) LOS objects, 42% free, 27MB/47MB, paused 1.649ms total 1.762s  12-12 21:23:01.538 736 751 I system\_server: NativeAlloc concurrent copying GC freed 89521(5MB) AllocSpace objects, 25(820KB) LOS objects, 42% free, 26MB/46MB, paused 467us total 645.702ms  12-12 21:23:02.469 736 786 I system\_server: Explicit concurrent copying GC freed 14855(893KB) AllocSpace objects, 3(124KB) LOS objects, 42% free, 25MB/45MB, paused 312us total 691.443ms  12-12 21:23:27.319 736 786 I system\_server: Explicit concurrent copying GC freed 134752(7MB) AllocSpace objects, 47(1260KB) LOS objects, 42% free, 26MB/46MB, paused 324us total 922.202ms  12-12 21:23:30.909 736 786 I system\_server: Explicit concurrent copying GC freed 29576(1996KB) AllocSpace objects, 16(448KB) LOS objects, 42% free, 26MB/46MB, paused 235us total 668.274ms |

我们可以看到正常log 中 system\_server 堆大小和占用空间26MB/46MB左右，而出问题时相关值已超过200M，因此我们怀疑其存在内存泄漏的可能。

## hprof文件的获取

我们可以根据需求通过图形界面工具DDMS来抓取进程的hprof文件，也可以通过命令行方式抓取进程的hprof文件。

### 通过DDMS抓取hprof

我们可以通过 DDMS 工具很方便的抓取进程的hprof文件。

1、打开DDMSTools-->Android-->Android Device Monitor

2、操作APP查看内存分配

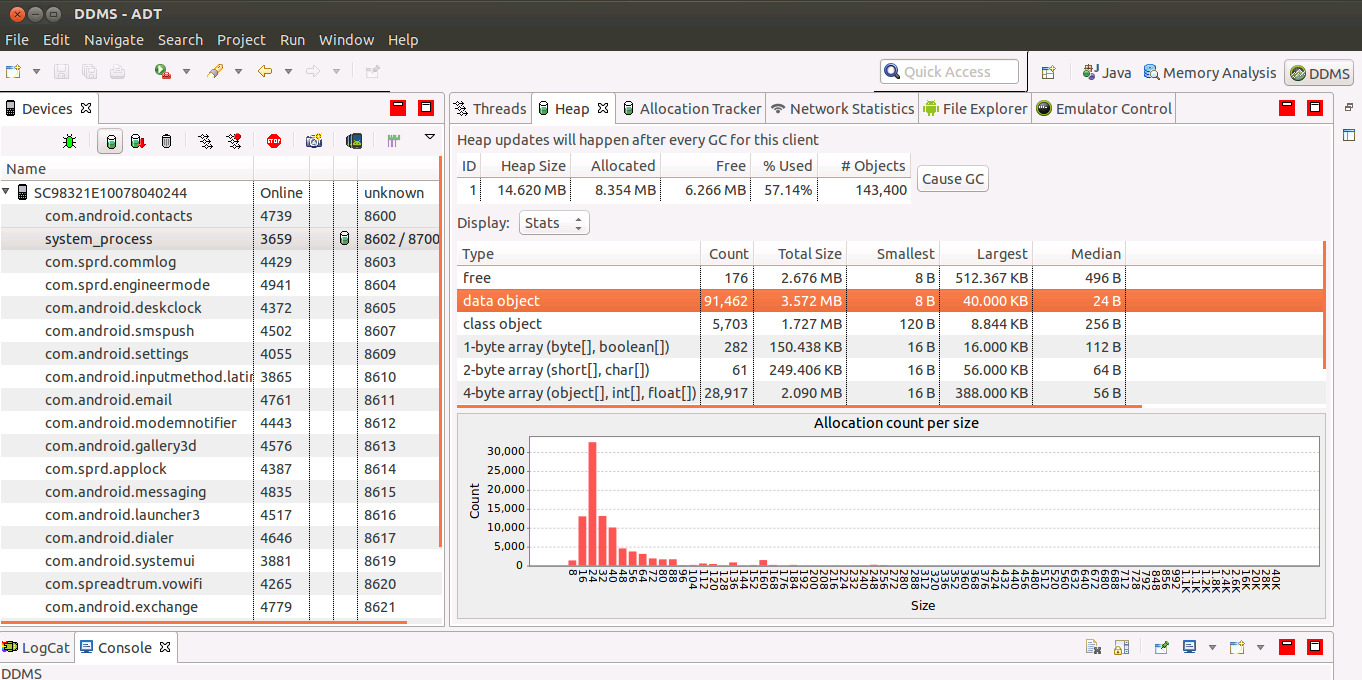
（1）选择进程：com.xxx.xx

（2）右侧面板，选择Heap

（3）点击update heap按钮（左侧绿色柱子按钮）

（4）查看Heap中allocated分配情况

（5）当发现内存抖动/持续增长/卡顿时，点击Cause GC ，释放内存（可操作几次后，继续点击Cause GC）



3、生成Hprof 文件

（1）Cause GC后，再操作几秒（可多次点击Cause GC）

（2）点击Dump HPROF file 按钮，生成hprof文件

### 通过命令行抓取hprof

我们可以使用 dumpheap 抓取进程的java层的 hprof栈.

|  |
| --- |
| # am dumpheap -g <pid> /data/local/tmp/hprof\_01.hprof |

其中:

* Pid：是被抓取的进程的 进程ID
* -g : 参数是在抓取前先进行一次GC
* 生成文件建议放在 /data/local/tmp 下

对于泄漏不明显的进程，可以写一脚本，每隔一段时间自动抓取hprof文件，然后对比其增长情况。

## hprof文件的分析转换

通过上面方法获取的进程的hprof文件是不能直接被MAT打开，需要转换。

### 方法

用DDMS生成的.hprof文件需要转换后才可以用MAT打开，转换步骤如下：

* 把生成的.hprof文件复制到hprof-conv所在目录

（在SDK安装目录下的platform-tools文件夹下）。

* 打开CMD命令行，输入hprof-conv xxxxx.hprof yyyyy.hprof

（其中xxxxx.hprof为原先生成的hprof文件，yyyyy.hprof为转换之后hprof）。

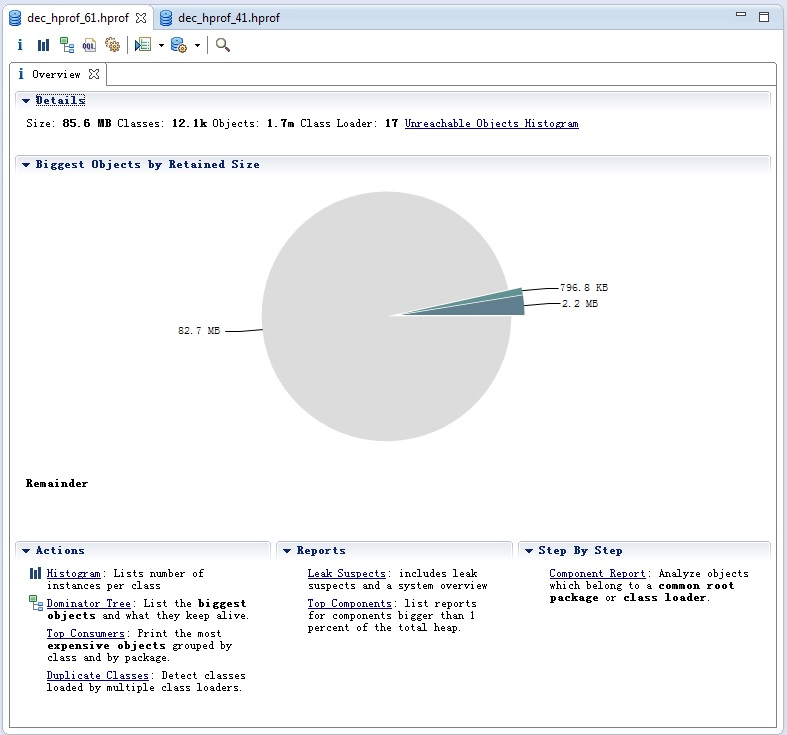
### 举例：

将源文件 hprof\_01.hprof 解码， 解码后生成：dec\_hprof\_01.hprof

$ /android-sdk/platform-tools/hprof-conv hprof\_01.hprof dec\_hprof\_01.hprof

# Mat工具使用简介

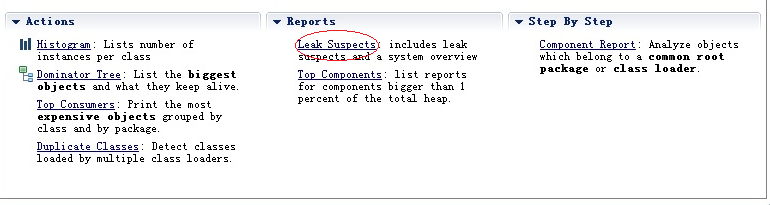
MAT是MemoryAnalysisTool的缩写，我们打开转换后的hprof文件后一般界面如下。



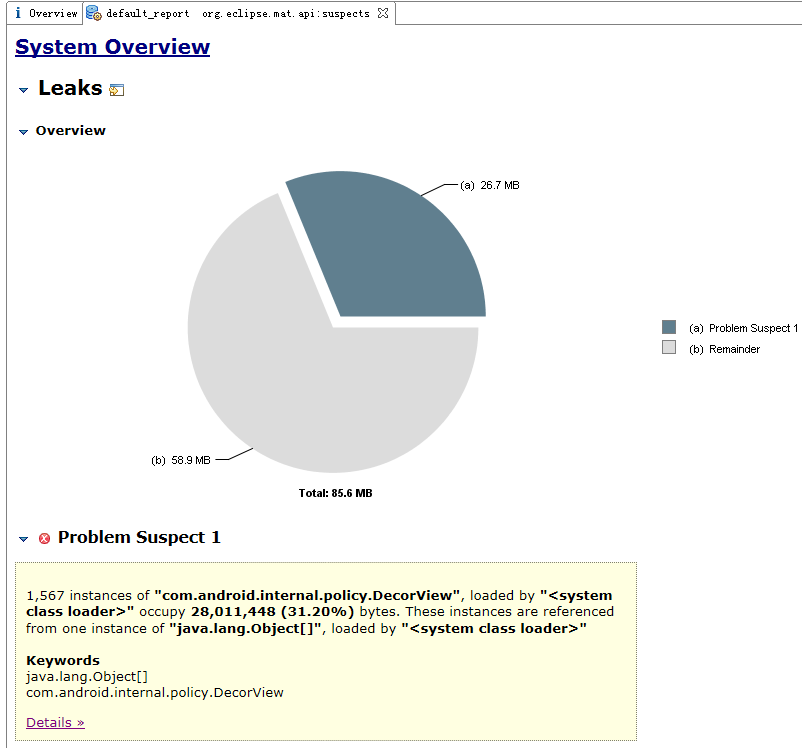
对于内存泄漏来说，我们常用的功能是 LeakSuspects ,histogram等

## LeakSuspects

导 入.hprof文件以后，MAT会自动解析并生成报告，我们也可以通过点击LeakSuspects来生成这个报告

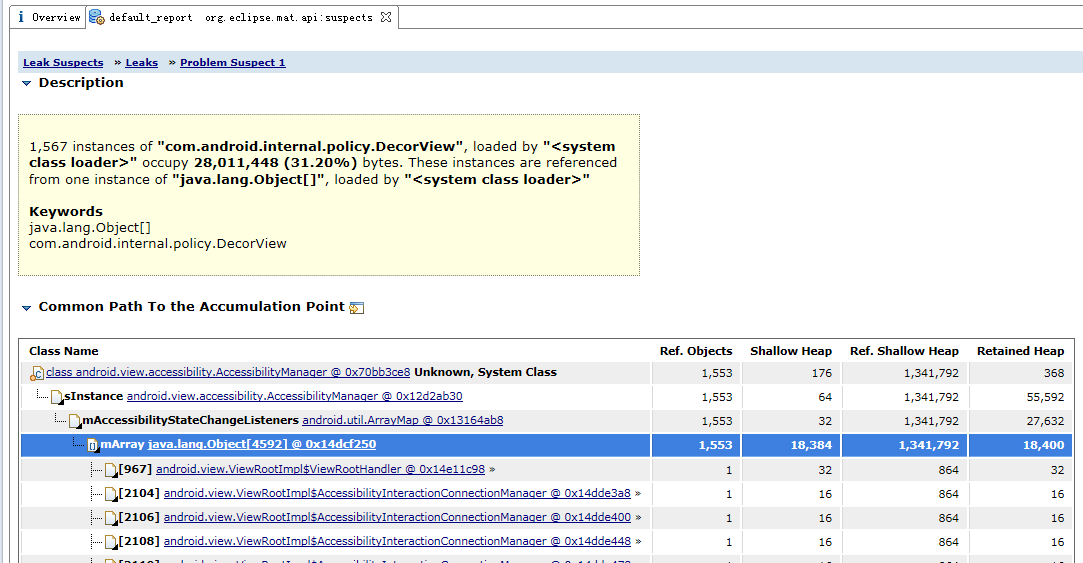


LeakSuspects 生成报告如图：



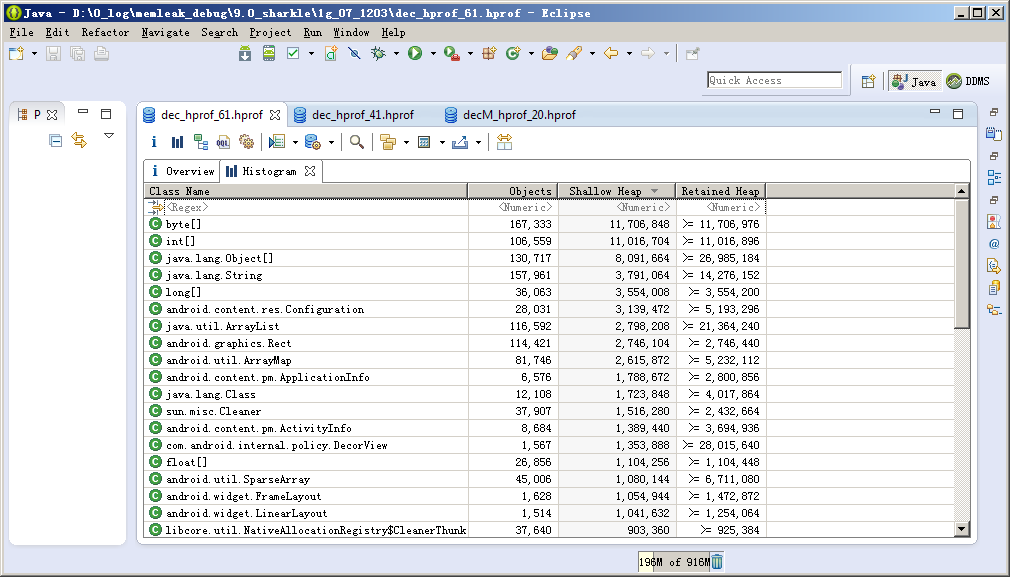
图中给出了疑似泄漏的类

也给出了该类的对象。

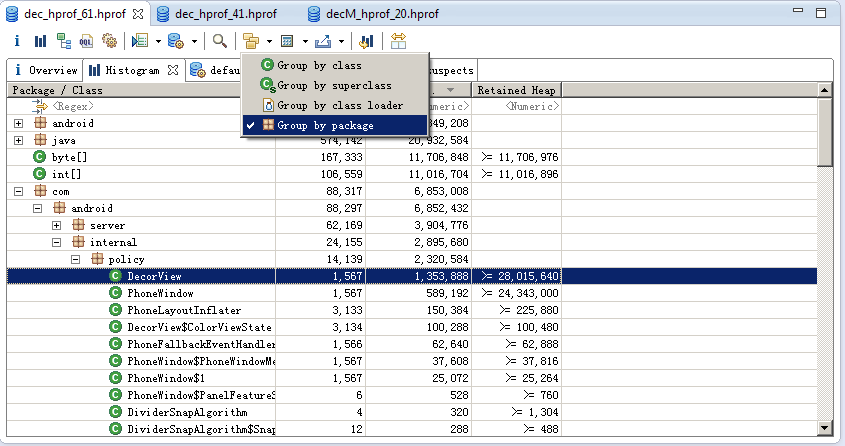


## histogram

histogram列出了各个类的对象的数量及内存占用

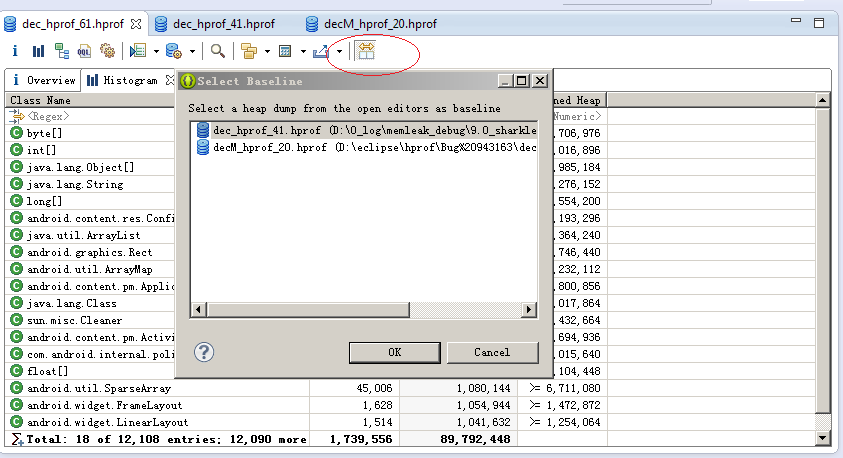


我们也可以选择按package来显示

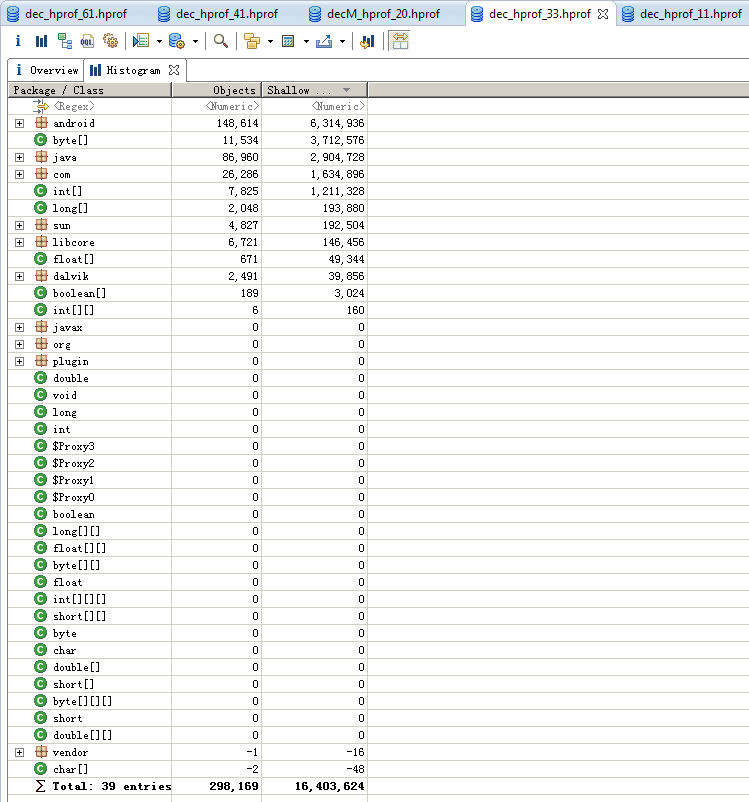


### 差异比较

我们可以比较两次hprof的差异：点击下图按钮在弹出的窗口中选择已打开的hprof文件即可

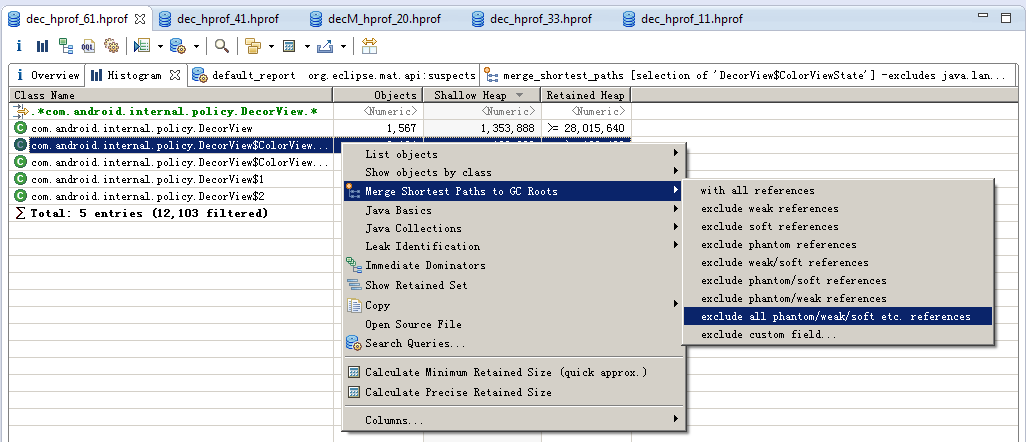


下面这张图我们可以看到，两个时间点对比的结果有增加，有减少



### 其他

顶行可以进行过滤，菜单可以查找显示其引用



# 问题分析中的应用

本章是对相关bug进行分析定位的举例。

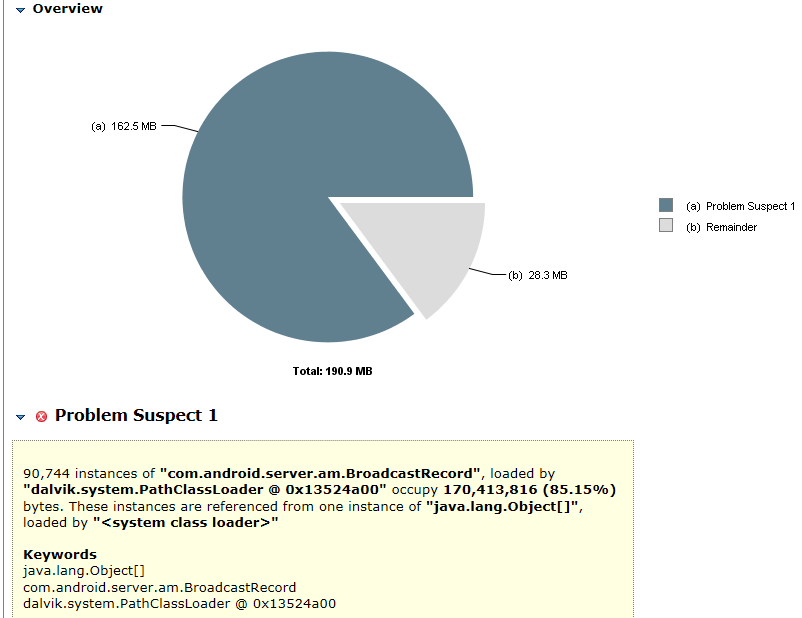
## 例一

* Bug 943163- [PSST][Android9.0][SharkLE][Go1G] [GMS] [PTR2Block] 进行CTS\_Keystore+Hotplug测试, #1401测试20小时出现多次Android重启

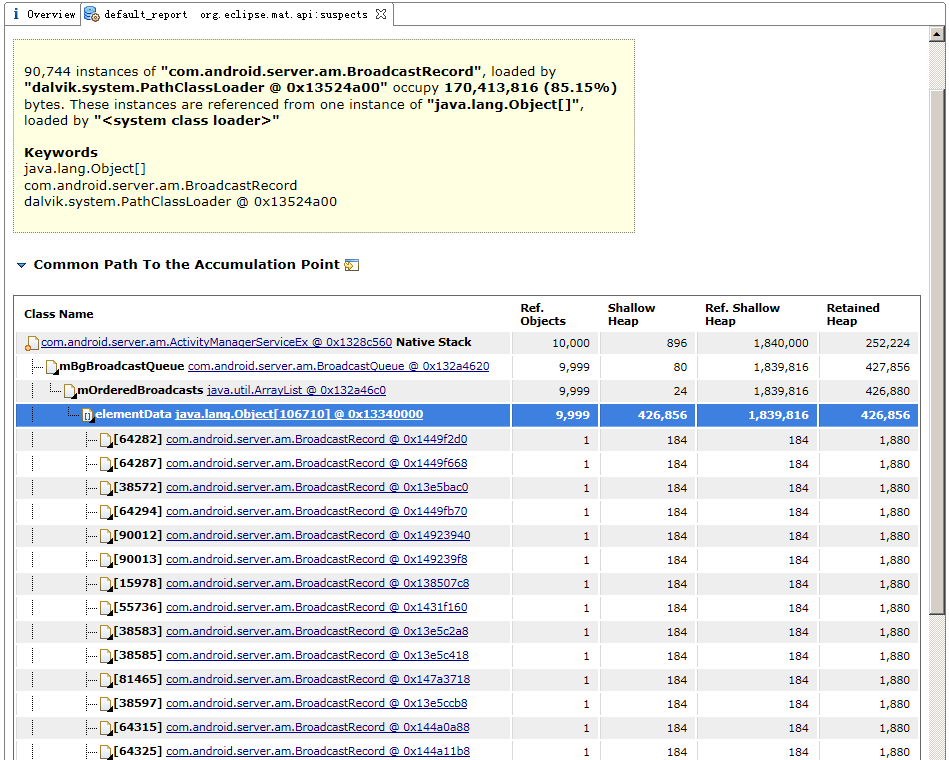
Log看system\_server 堆占用超过200M

|  |
| --- |
| M2B991A 09-25 10:38:04.555 723 733 I system\_server: Background concurrent copying GC freed 604510(37MB) AllocSpace objects, 0(0B) LOS objects, 3% free, 218MB/226MB, paused 255us total 3.000s  M2BAE39 09-25 10:38:09.080 723 733 I system\_server: Background concurrent copying GC freed 379781(23MB) AllocSpace objects, 1(20KB) LOS objects, 3% free, 232MB/240MB, paused 234.130ms total 4.145s  M2BC0D4 09-25 10:38:19.107 723 733 I system\_server: Background concurrent copying GC freed 469441(28MB) AllocSpace objects, 11(220KB) LOS objects, 3% free, 226MB/234MB, paused 1.182ms total 1.119s |

每半小时抓取hprof文件，使用MAT工具进行分析



发现存在过多多广播消息



加Log复测发现是大量DROPBOX\_ENTRY\_ADDED 消息出现堆积

|  |
| --- |
| S6F9F04 09-25 11:54:59.669 726 762 D BroadcastQueue: Add broadcast <BroadcastRecord{2b97455 u0 android.intent.action.DROPBOX\_ENTRY\_ADDED}> into [ordered | background], pending size 93441  S6F9F05 09-25 11:54:59.672 726 762 D BroadcastQueue: Add broadcast <BroadcastRecord{9269e6a u0 android.intent.action.DROPBOX\_ENTRY\_ADDED}> into [ordered | background], pending size 93442 |

Dropbox目录下大量的keymaster@1539146991850.dat 的文件

## 例二

* Bug 976929 [Sprdroid9.0\_SharklE] [FW-Stability][] New: [BJ\_RD] [android9.0\_sharkle\_GMS][9832e][StabilityTest][内存扫描]1257#机进行memleak测试，system\_server疑似内存泄漏

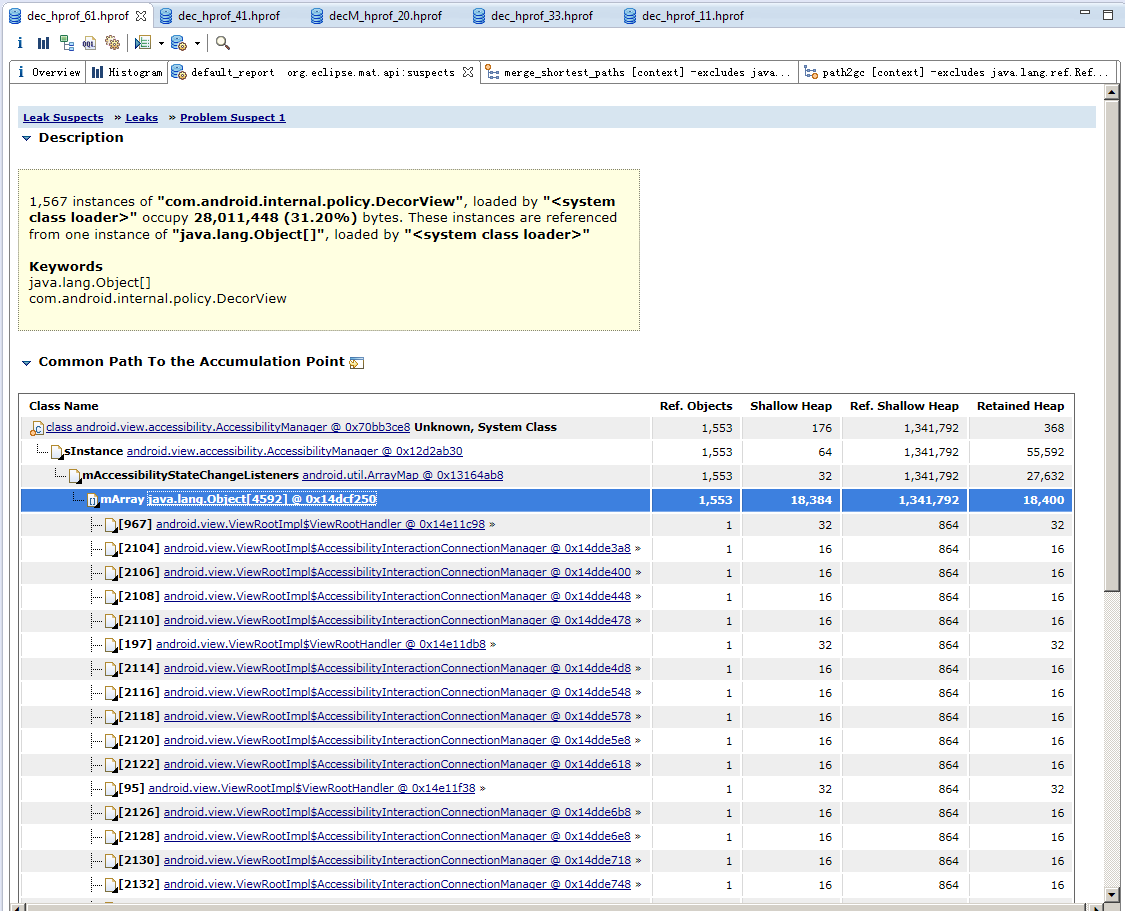
Memkeak工具分析疑似内存泄漏，于是复测抓取native栈进行分析，发现native层内存调用栈基本合理，java层内存有增加但幅度不大，

于是monkey 24小时复测，每半小时抓取hprof文件分析，看不出异常。

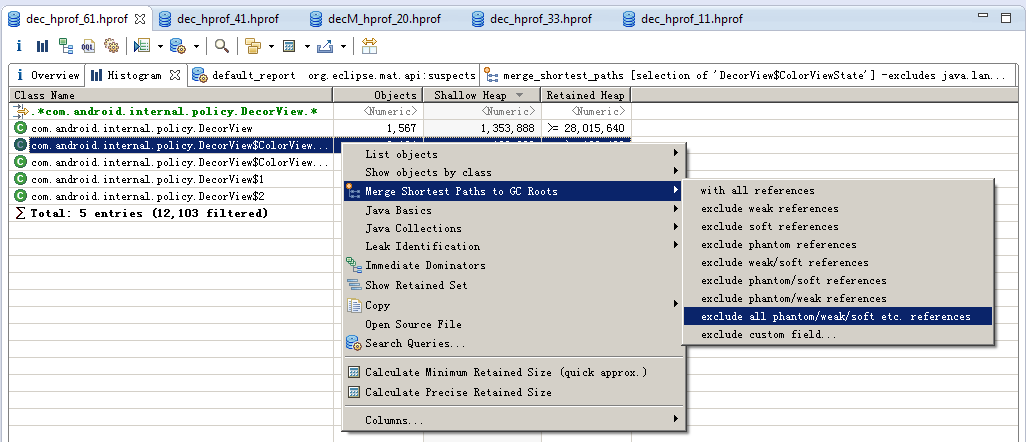
解决了一些系统重启的问题之后monkey测试可以跑到60小时+，于是每小时抓取hprof脚本进行分析。



可以看到系统中存在大量



Histogram 过滤 com.android.internal.policy.DecorView 并查找其引用



可以看到大量的 android.view.SprdViewRootImpl 的引用

