Low Memory Issue Debug

Agenda

- 如何判定有low memory
- Android Memory Layout
- User Space Low Memory handling
- Linux Kernel Low memory handling
- Java Heap OOM handling
- Summary

如何判定low memory

菜单

Setting->apps->running app

计算方法请参考:
RunningProcessesView.java(/packages/apps/settings/src/com/android/settings/applications)中的void **refreshUi**(boolean dataChanged)

可用 内存已经很低》》》

如何判定low memory

有进程被low memory killer 杀掉

Kernel log 里面搜索：candidate 或者sigkill

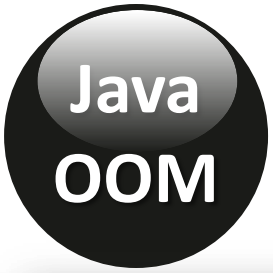
```
                                                    pid  process_name   adj     use memory
Line 7160: <4>[  229.892278] (1)[26:kswapd0]Candidate 1 (init), adj -16, rss 128, to kill
Line 7161: <4>[  229.893124] (1)[26:kswapd0]Candidate 84 (ueventd), adj -16, rss 59, to kill
Line 7163: <4>[  229.894001] (1)[26:kswapd0]Candidate 121 (servicemanager), adj -16, rss 71, to kill
Line 7165: <4>[  229.894962] (1)[26:kswapd0]Candidate 122 (vold), adj -16, rss 165, to kill
Line 7166: <4>[  229.895820] (1)[26:kswapd0]Candidate 123 (logwrapper), adj -16, rss 74, to kill
Line 7167: <4>[  229.896801] (1)[26:kswapd0]Candidate 124 (ccci_fsd), adj -16, rss 95, to kill
Line 7168: <4>[  229.897721] (1)[26:kswapd0]Candidate 126 (ccci_mdinit), adj -16, rss 59, to kill
```

```
Line 7424: <4>[  230.120623] (1)[26:kswapd0]send sigkill to 2319 (o.leos.appstore), adj 9, size 5440
```

VSS/**RSS**/PSS/USS 意义？→请问baidu|google

adj 的定义:processlist.java, HIDDEN_APP_MIN_ADJ = 9

如何判定low memory



Java Heap Out of memory

Keyword:
Out of memory or
OutOfMemoryError in
main_log.

```
07-20 11:29:00.428 11074 11748 D dalvikvm: GC_BEFORE_OOM freed 0K, 1% free 65076K/65527K, paused 78ms
07-20 11:29:00.428 11074 11748 E dalvikvm-heap: Out of memory on a 499408-byte allocation.
07-20 11:29:00.428 11074 11748 I dalvikvm: "Thread-652" prio=5 tid=16 RUNNABLE
07-20 11:29:00.428 11074 11748 I dalvikvm:   | group="main" sCount=0 dsCount=0 obj=0x4339bba0 self=0x161aad0
07-20 11:29:00.428 11074 11748 I dalvikvm:   | sysTid=11748 nice=0 sched=0/0 cgrp=default handle=23172928
07-20 11:29:00.428 11074 11748 I dalvikvm:   | schedstat=( 375203927 155522461 398 ) utm=34 stm=3 core=0
07-20 11:29:00.428 11074 11748 I dalvikvm:   at android.graphics.BitmapFactory.nativeDecodeFileDescriptor(Native Method)
07-20 11:29:00.428 11074 11748 I dalvikvm:   at android.graphics.BitmapFactory.decodeFileDescriptor(BitmapFactory.java:587)
07-20 11:29:00.428 11074 11748 I dalvikvm:   at android.provider.MediaStore$InternalThumbnails.getMiniThumbFromFile(MediaStore.
07-20 11:29:00.428 11074 11748 I dalvikvm:   at android.provider.MediaStore$InternalThumbnails.getThumbnail(MediaStore.java:631
07-20 11:29:00.428 11074 11748 I dalvikvm:   at android.provider.MediaStore$Images$Thumbnails.getThumbnail(MediaStore.java:1044
07-20 11:29:00.428 11074 11748 I dalvikvm:   at com.android.camera.Thumbnail.getLastThumbnail(Thumbnail.java:177)
07-20 11:29:00.428 11074 11748 I dalvikvm:   at com.android.camera.VideoCamera$9.run(VideoCamera.java:2301)
07-20 11:29:00.428 11074 11748 I dalvikvm:
07-20 11:29:00.429 11074 11748 W dalvikvm: Exception thrown (Ljava/lang/OutOfMemoryError;) while throwing internal exception (L
07-20 11:29:00.429 11074 11748 W skia    : libjpeg error 105 <  Ss=%d, Se=%d, Ah=%d, Al=%d> from allocPixelRef [408 306]
```

```
07-20 11:29:00.593 11074 11748 I dalvikvm-heap: Clamp target GC heap from 65.558MB to 64.000MB
07-20 11:29:00.593 11074 11748 E dalvikvm-heap: Out of memory on a 65552-byte allocation.
07-20 11:29:00.684 11074 11753 I dalvikvm-heap: Clamp target GC heap from 65.560MB to 64.000MB
```


Main_log.1

[dalvik.vm.heapgrowthlimit]: [64m]
[dalvik.vm.heapsize]: [128m]

如何判定low memory

adb commands

adb shell cat /proc/meminfo
adb shell dumpsys meminfo

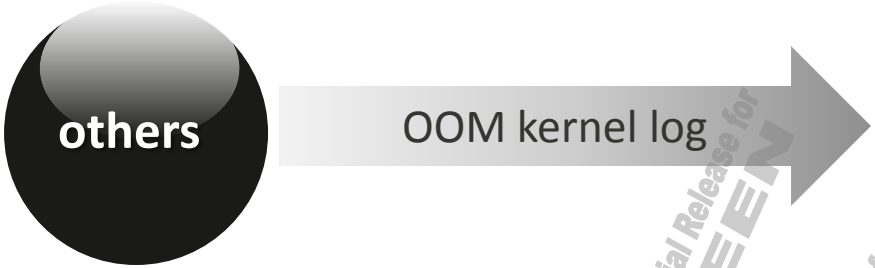
```
MemTotal:        2981164 kB
MemFree:         1202424 kB
Buffers:           13504 kB
Cached:           366620 kB
SwapCached:            0 kB
```

Memfree + Cached 偏低？？

Free 偏低？？

```
Total RAM: 2981164 kB
 Free RAM: 2485131 kB (89095 cached pss + 334388 cached + 2061648 free)
 Used RAM: 350841 kB (310445 used pss + 7888 buffers + 428 shmem + 32080 slab)
 Lost RAM: 145192 kB
```

如何判定low memory



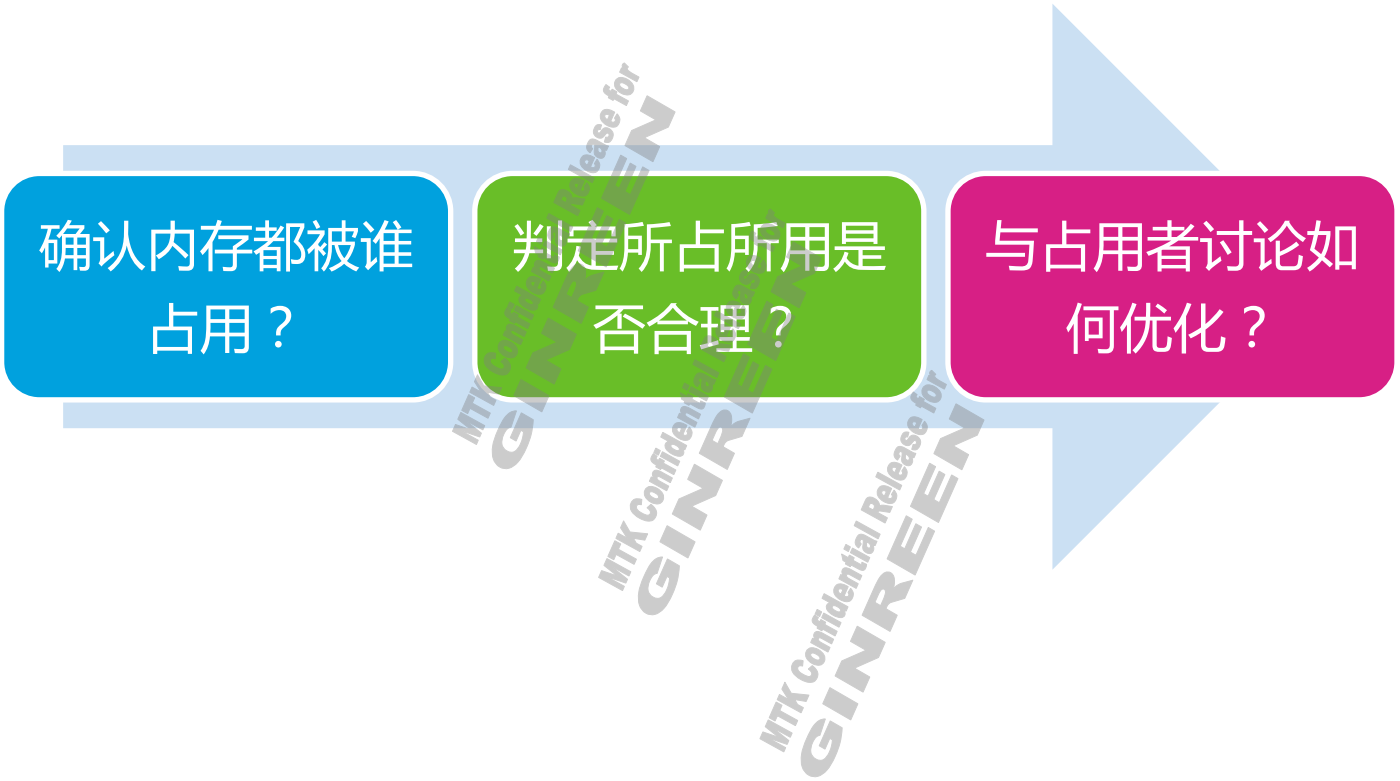
Out of memory: Kill process 9293 (rmanager:remote) score 789 or sacrifice child

```
(2)[9502:android.gms:ca]Out of memory: Kill process 9293 (rmanager:remote) score 789 or sacrifice child
(2)[9502:android.gms:ca]Killed process 9293 (rmanager:remote) total-vm:1381480kB, anon-rss:25220kB, file-rss:23304kB
-(2)[9502:android.gms:ca][9502:android.gms:ca] sig 9 to [9293:rmanager:remote] stat=S
(3)[9502:android.gms:ca]android.gms:ca invoked oom-killer: gfp_mask=0x3000d0, order=2, oom_score_adj=-1000
(3)[9502:android.gms:ca]CPU: 3 PID: 9502 Comm: android.gms:ca Tainted: P        W  O 3.10.65-ge0ca7b9 #1
(3)[9502:android.gms:ca]Call trace:
(3)[9502:android.gms:ca][<ffffffc000089144>] dump_backtrace+0x0/0x16c
(3)[9502:android.gms:ca][<ffffffc0000892c0>] show_stack+0x10/0x1c
(3)[9502:android.gms:ca][<ffffffc000a1a654>] dump_stack+0x1c/0x28
(3)[9502:android.gms:ca][<ffffffc000157934>] dump_header.isra.14+0x6c/0x1b0
(3)[9502:android.gms:ca][<ffffffc000157f10>] oom_kill_process+0x2c0/0x444
(3)[9502:android.gms:ca][<ffffffc000158570>] out_of_memory+0x2bc/0x2f8
(3)[9502:android.gms:ca][<ffffffc00015d240>] __alloc_pages_nodemask+0x7dc/0x7f0
(3)[9502:android.gms:ca][<ffffffc000098ee4>] copy_process+0x114/0x1114
(3)[9502:android.gms:ca][<ffffffc000099f94>] do_fork+0x7c/0x3d0
(3)[9502:android.gms:ca][<ffffffc00009a380>] SyS_clone+0x10/0x1c
```

Low memory会带来什么问题？

- Performance issues
- App runs abnormal(LMK)
- System Crash(OOM)

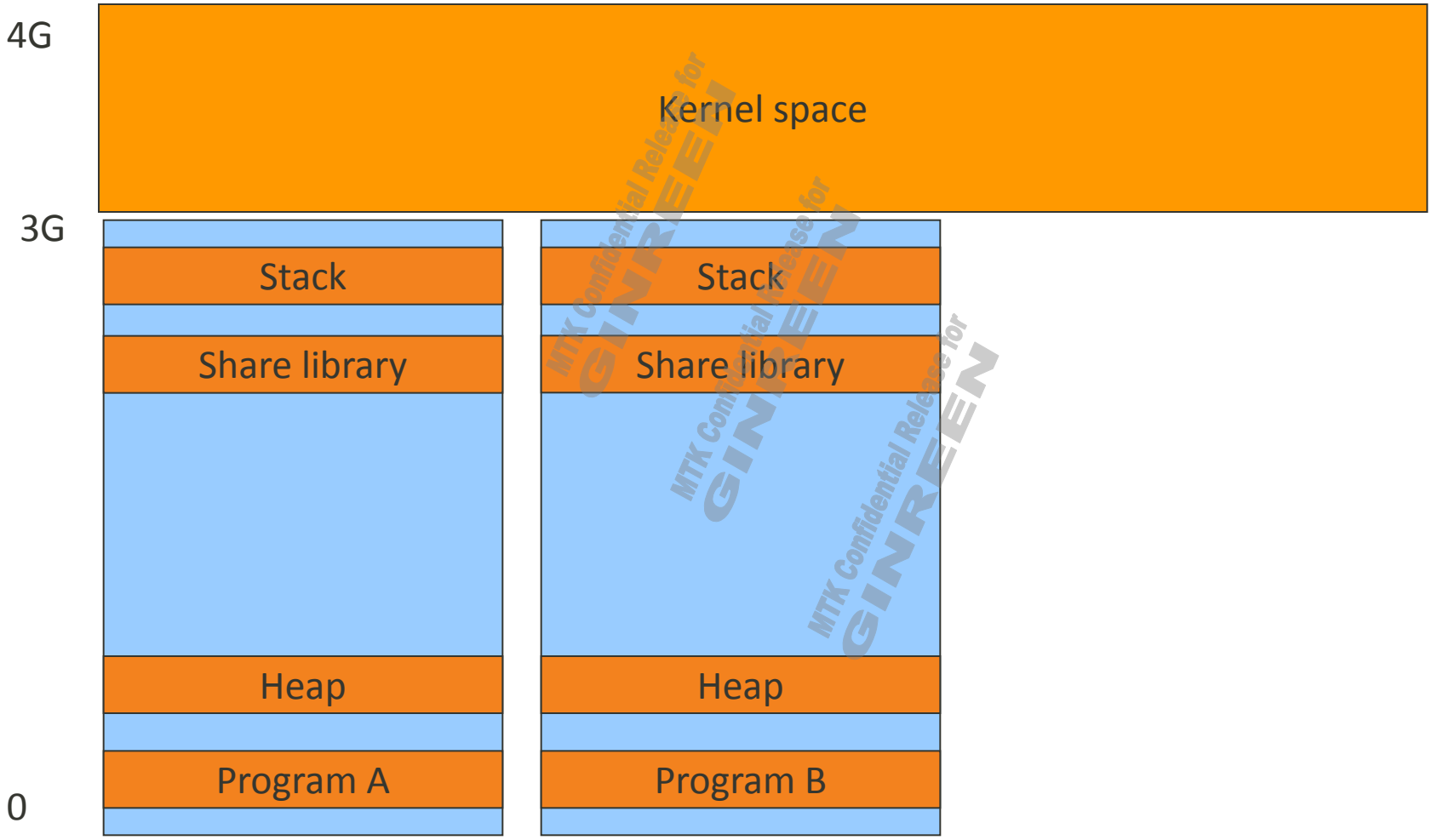
如何分析低内存问题？



Android 系统内存分布情况是怎样的？

Linux memory layout

32 位 Virtual address (0 - 4G)， 64变化巨大



如何查看memory layout

- User space process
  - adb shell cat /proc/[pid]/maps
  - adb shell cat /proc/[pid]/smaps
  - adb shell showmap pid(eng only)
- Kernel space
  - adb shell cat /proc/mtk_memconfig/memory_layout(eng load)
  - 物理地址layout
    - FAQ07759 **如何查看当前项目的physical memory layout**
  - 虚拟地址layout
    - FAQ07760 **如何查看当前项目的virtual memory layout**

Memory layout example

- 32

```
root@mt6592_phone_720p:/proc/mtk_memcfg # cat memory_layout
<1>[PHY layout]avaiable DRAM size (lk) = 0x40000000
   [PHY layout]avaiable DRAM size = 0x40000000
   [PHY layout]FB       :   0xbeb00000 - 0xbfffffff   (0x1500000)
<1>[PHY layout]mtk_wcn_consys_memory_reserve   :   0xbea00000 - 0xbeafffff (0x00100000)
<1>[PHY layout]ccci_md_mem_reserve   :   0xbc000000 - 0xbd7fffff (0x01800000)
<1>[PHY layout]kernel   :   0x80000000 - 0x9feffffff (0x1ff00000)
<1>[PHY layout]kernel   :   0xa0100000 - 0xbbffffff (0x1bf00000)
<5>Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    vmalloc : 0xe0800000 - 0xff000000   ( 488 MB)
    lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
    pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
    modules : 0xbf000000 - 0xbfe00000   (  14 MB)
      .text : 0xc0008000 - 0xc0b6c000   (11664 kB)
      .init : 0xc0b6c000 - 0xc0bcdd00   ( 392 kB)
      .data : 0xc0bce000 - 0xc0ca0e20   ( 844 kB)
       .bss : 0xc0ca0e44 - 0xc174af60   (10921 kB)
```

Memory layout example

```
cat /proc/mtk_memcfg/memory_layout
SOH1[memblock]mrdump_mini_reserve_memory: 0x43ff0000 - 0x43ffafff (0x0000b000)
SOH1[PHY layout]reserve-memory-ccci_md1    :   0xba000000 - 0xbd80ffff (0x3810000)
SOH1[PHY layout]consys-reserve-memory    :   0xbe000000 - 0xbe1fffff (0x200000)
SOH1[PHY layout]reserve-memory-ccci_md3    :   0xb8000000 - 0xb8ffffff (0x1000000)
SOH1modem_sdio_reserve_mem_of_init,uname:reserve-memory-ccci_md3,base:0xb8000000,size:0x1000000
SOH1[PHY layout]kernel    :   0x40000000 - 0x42ffffff (0x3000000)
SOH1[PHY layout]kernel    :   0x43030000 - 0xb7ffffff (0x74fd0000)
SOH1[PHY layout]kernel    :   0xb9000000 - 0xb9ffffff (0x1000000)
SOH1[PHY layout]kernel    :   0xbd810000 - 0xbdffffff (0x27f0000)
SOH1[PHY layout]kernel    :   0xbe200000 - 0xbe3dffff (0x1e0000)
SOH5Virtual kernel memory layout:
    vmalloc : 0xffffff8000000000 - 0xffffffbbffff0000   (245759 MB)
    vmemmap : 0xffffffbc00e00000 - 0xffffffbc0299d980   (    27 MB)
    modules : 0xffffffbffc000000 - 0xffffffc000000000   (    64 MB)
    memory  : 0xffffffc000000000 - 0xffffffc07e3e0000   (  2019 MB)
      .init : 0xffffffc000e73000 - 0xffffffc000ed8100   (   405 kB)
      .text : 0xffffffc000080000 - 0xffffffc000e73000   ( 14284 kB)
      .data : 0xffffffc000edc000 - 0xffffffc00102f0b8   (  1357 kB)
```

整体把握 — 谁大？
Kernel space
or
User space ?

查看内存使用的常用command

- 参考FAQ04354 内存不足时查看内存使用情况的一些adb command
- 整体把握的cmds：
  - adb shell cat /proc/meminfo
  - adb shell dumpsys meminfo
  - adb shell procrank -s (eng/userdebug)

/proc/meminfo

- MemTotal：扣掉开机预留后kernel可用
- Free：物理剩余内部
- Cached：Read file 缓存，可被及时回收
- Buffers： write buffer
- AnonPages + Mapped: user space 未压缩占用之空间
- SwapTotal-SwapFree: 已经压缩出去user space占用空间
- Slab/KernelStack/pageTables/VmallocUsed均属kernel占用(仅一部分)
- 所以，此meminfo可初步判定user space占用情况： AnonPages + Mapped+ SwapTotal-SwapFree

```
MemTotal:         422472 kB
MemFree:            6804 kB
Buffers:            1048 kB
Cached:            94336 kB
SwapCached:         2828 kB
Active:           133832 kB
Inactive:         136092 kB
Active(anon):      85520 kB
Inactive(anon):    90836 kB
Active(file):      48312 kB
Inactive(file):    45256 kB
Unevictable:        1404 kB
Mlocked:               0 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         422472 kB
LowFree:            6804 kB
SwapTotal:        319484 kB
SwapFree:         241684 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        175596 kB
Mapped:            53772 kB
Shmem:               412 kB
Slab:              26556 kB
SReclaimable:       9272 kB
SUnreclaim:        17284 kB
KernelStack:        9672 kB
PageTables:        15528 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      530720 kB
Committed_AS:   23271708 kB
VmallocTotal:     507904 kB
VmallocUsed:       73256 kB
VmallocChunk:     143744 kB
Committed_AS:   23271708 kB
```

dumpsys meminfo

```
Total RAM: 966112 kB
 Free RAM: 556216 kB (44044 cached pss + 75820 cached + 436352 free)
 Used RAM: 286649 kB (213433 used pss + 444 buffers + 232 shmem + 72540 slab)
 Lost RAM: 123247 kB
     ZRAM: 1296 kB physical used for 2300 kB in swap (524284 kB total swap)
   Tuning: 128 (large 256), oom 122880 kB, restore limit 40960 kB (high-end-gfx)
```

- Cached pss + used pss : user space 未swap出去内存
- Lost RAM： Android 上层统计不到的kernel 占用memory，主要指某些 kernel driver 占用部分。目前主要为GPU，ION，ZRAM， 如果Lost RAM偏大，需要确认这三者的占用情况，参考下面commands
  - **GPU**： /proc/mali/memory_usage (若非mali GPU请咨询MTK)
  - **ION**： /sys/kernel/debug/ion/ion_mm_heap (不同版本可能有差异，具体可咨询MTK)
  - **ZRAM**：/proc/zraminfo --》memused
  - 如果上面三者占据不多，那有其它driver泄露的可能，比较常见的是TP driver 直接调用dmam_alloc_coherent()后没有释放，可优先检查。如果贵司无法确认，再提交ES请MTK协助

Procrank –s

被zram swap出去size

PID	Uss	Rss	Pss	Uss	Swap	PSwap	cmdline
153	51640K	3960K	925K	624K	640K	640K	/system/bin/atci_service
156	35652K	844K	40K	12K	440K	440K	/system/bin/drvbd
154	34276K	424K	48K	40K	352K	352K	/system/bin/atcid
1	928K	568K	419K	316K	216K	216K	/init
110	776K	312K	206K	104K	172K	172K	/sbin/ueventd
163	34156K	484K	138K	132K	168K	168K	/system/bin/servicemanager
181	562812K	48888K	15807K	9564K	112K	74K	zygote
1020	619880K	25248K	5399K	3956K	44K	12K	com.mediatek.voicecommand

某个进程占用：
PSS + Swap

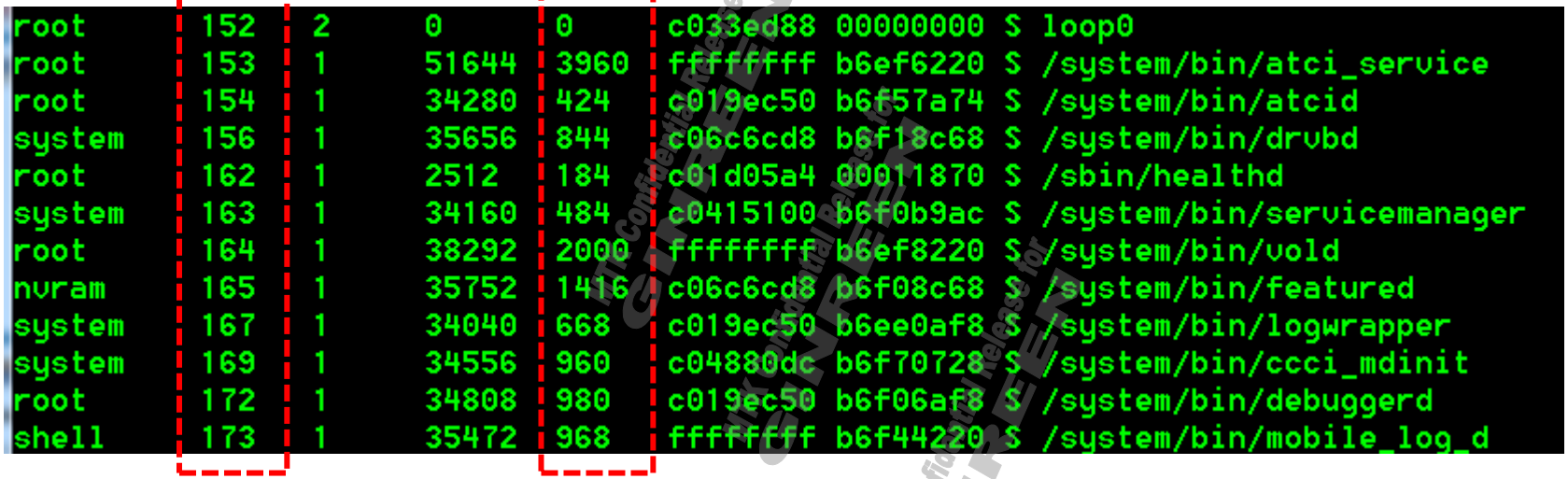
User Space占用异常时如何分析？

User space占用异常时如何分析？

- 整体把握– 判定哪些进程占用较多：
  - adb shell ps
  - adb shell dumpsys meminfo
  - adb shell procrank -s (eng/userdebug)
- 各个击破 – 寻找占用内存较多进程的具体信息：
  - adb shell cat /proc/pid/smaps
  - adb shell showmap pid (eng)

Ps 查看userspace进程占据情况

PID RSS



```
root      152   2     0       0     c033ed88 00000000 S loop0
root      153   1     51644   3960  ffffffff b6ef6220 S /system/bin/atci_service
root      154   1     34280   424   c019ec50 b6f57a74 S /system/bin/atcid
system    156   1     35656   844   c06c6cd8 b6f18c68 S /system/bin/drvbd
root      162   1     2512    184   c01d05a4 00011870 S /sbin/healthd
system    163   1     34160   484   c0415100 b6f0b9ac S /system/bin/servicemanager
root      164   1     38292   2000  ffffffff b6ef8220 S /system/bin/vold
nvram     165   1     35752   1416  c06c6cd8 b6f08c68 S /system/bin/featured
system    167   1     34040   668   c019ec50 b6ee0af8 S /system/bin/logwrapper
system    169   1     34556   960   c04880dc b6f70728 S /system/bin/ccci_mdinit
root      172   1     34808   980   c019ec50 b6f06af8 S /system/bin/debuggerd
shell     173   1     35472   968   ffffffff b6f44220 S /system/bin/mobile_log_d
```

Dumpsys meminfo 查看userspace进程占据情况

```
Applications Memory Usage (kB):
Uptime: 99931 Realtime: 99931

Total PSS by process:
    47410 kB: com.android.keyguard (pid 819)
    45147 kB: system (pid 730)
    42443 kB: com.bbk.launcher2 (pid 1005 / activities)
    30234 kB: com.android.systemui (pid 839)
    26287 kB: android.process.acore (pid 1202)
    19129 kB: com.android.phone (pid 1371)
    13480 kB: zygote (pid 174)
    13428 kB: mediaserver (pid 176)
    13135 kB: com.iqoo.secure (pid 1037)
     7684 kB: com.baidu.input_bbk.service (pid 915)
     5880 kB: com.vivo.daemonService (pid 950)
     5835 kB: com.android.bbkmusic (pid 1302)
     5702 kB: com.baidu.map.location (pid 938)
```

Procrank –s 查看userspace进程占用状况

PID	Vss	Rss	Pss	Uss	Swap	PSwap	cmdline
153	51640K	3960K	925K	624K	640K	640K	/system/bin/atci_service
156	35652K	844K	40K	12K	440K	440K	/system/bin/drvbd
154	34276K	424K	48K	40K	352K	352K	/system/bin/atcid
1	928K	568K	419K	316K	216K	216K	/init
110	776K	312K	206K	104K	172K	172K	/sbin/ueventd
163	34156K	484K	138K	132K	168K	168K	/system/bin/servicemanager
181	562812K	48888K	15807K	9564K	112K	74K	zygote
1020	619880K	25248K	5399K	3956K	44K	12K	com.mediatek.voicecommand

确认PSS + Swap是否有异常者

/proc/pid/smaps 查看某个进程占用内存的具体分布

40048000-400b5000 r-xp 00000000 b3:05 919 /system/lib/libc.so

Size: 436 kB

Rss: 328 kB

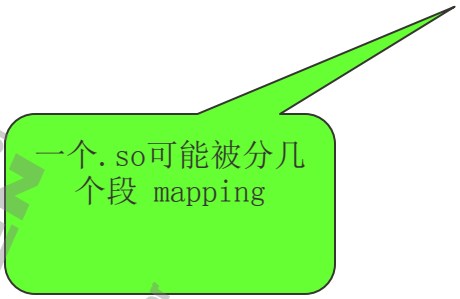
Pss: 12 kB

Shared_Clean: 328 kB

Shared_Dirty: 0 kB

Private_Clean: 0 kB

Private_Dirty: 0 kB



showmap pid 查看某个进程占用内存的具体分布

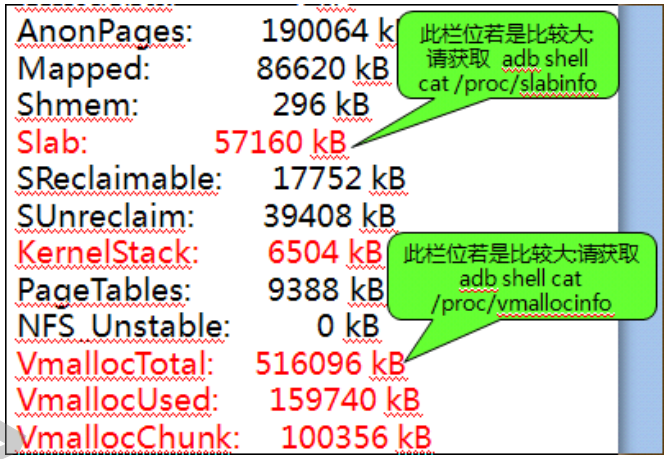
virtual size	RSS	PSS	clean	dirty	shared clean	shared dirty	private #	object
52	32	0	0	32	0	0	1	/dev/__properties__ (deleted)
2040	4	4	0	0	4	0	1	/dev/binder
4	4	0	4	0	0	0	1	/dev/xLog
72	56	1	56	0	0	0	3	/system/bin/linker
16	8	8	0	0	4	4	3	/system/bin/rild
144	124	14	116	0	0	8	3	/system/lib/libbinder.so
500	276	20	260	0	0	16	5	/system/lib/libc.so
.....	....	...	....	...	...	...	...	.....
1020	0	0	0	0	0	0	1	[stack:534]
1020	0	0	0	0	0	0	1	[stack:535]
1020	0	0	0	0	0	0	1	[stack:536]
136	4	4	0	0	0	4	1	[stack]
4	0	0	0	0	0	0	1	[vectors]
virtual size	RSS	PSS	clean	dirty	shared clean	shared dirty	private #	object
30608	964	335	612	32	136	184	122	TOTAL

找到哪一块的RSS比较异常大

Kernel Space占用异常时如何分析？

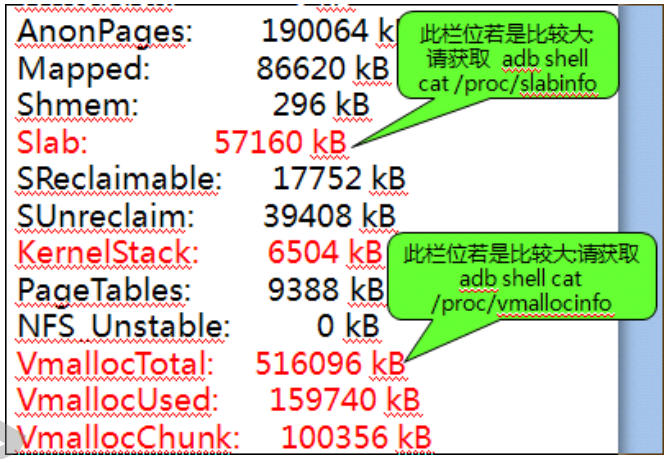
Linux Kernel Low memory handling

- adb shell cat /proc/slabinfo
- adb shell cat /proc/vmallocinfo



Linux Kernel Low memory handling

- Slab 异常：cat slabinfo，查看是哪个pool用的异常，以num_objs为主，再来决定下一步的debug方法：加log？Or review code？？




```
slabinfo - version: 2.1

# name            <active_objs> <num_objs> <objsize> <objpers
<sharedavail>

ext4_groupinfo_4k      17      24     336   12    1 : tunables

bridge_fdb_cache        0       0     256   16    1 : tunables

...
buffer_head        148351  148359     240   17    1 : tunables
```

- Vmalloc异常：cat vmallocinfo，参考 pages 栏位

```
0xbf000000-0xbf04b000   307200 module_alloc_update_bounds+0x14/0x68 pages=74 vmalloc

0xbf05b000-0xbf063000    32768 module_alloc_update_bounds+0x14/0x68 pages=7 vmalloc

0xbf066000-0xbf073000    53248 module_alloc_update_bounds+0x14/0x68 pages=12 vmalloc
```

Linux Kernel Low memory handling

- Kmemleak? May cause low memory

kmemleak如何使用

HW driver used too much? – Lost RAM

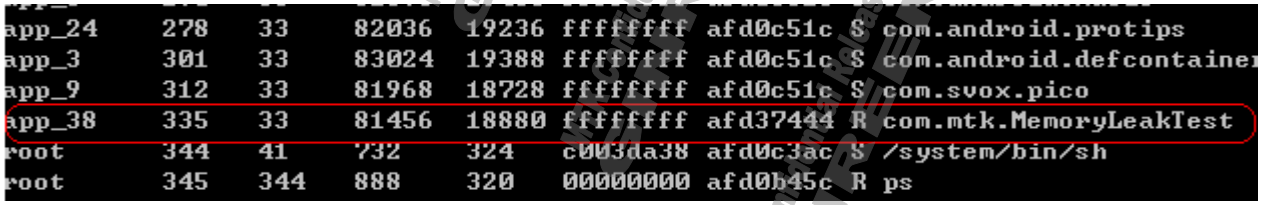
```
Total RAM: 966112 kB
 Free RAM: 556216 kB (44044 cached pss + 75820 cached + 436352 free)
 Used RAM: 286649 kB (213433 used pss + 444 buffers + 232 shmem + 72540 slab)
 Lost RAM: 123247 kB
     ZRAM: 1296 kB physical used for 2300 kB in swap (524284 kB total swap)
   Tuning: 128 (large 256), oom 122880 kB, restore limit 40960 kB (high-end-gfx)
```

- Lost RAM： Android 上层统计不到的kernel 占用memory，主要指某些 kernel driver 占用部分。目前主要为GPU，ION，ZRAM， 如果Lost RAM偏大，需要确认这三者的占用情况，参考下面commands
  - **GPU**： /proc/mali/memory_usage (若非mali GPU请咨询MTK)
  - **ION**： /sys/kernel/debug/ion/ion_mm_heap (不同版本可能有差异，具体可咨询MTK)
  - **ZRAM**：/proc/zraminfo --》memused
  - 如果上面三者占据不多，那有其它driver泄露的可能，比较常见的是TP driver 直接调用dmam_alloc_coherent()后没有释放，可优先检查。如果贵司无法确认，再提交ES请MTK协助

JAVA Heap OOM时如何分析？

Java Heap OOM - Hprof

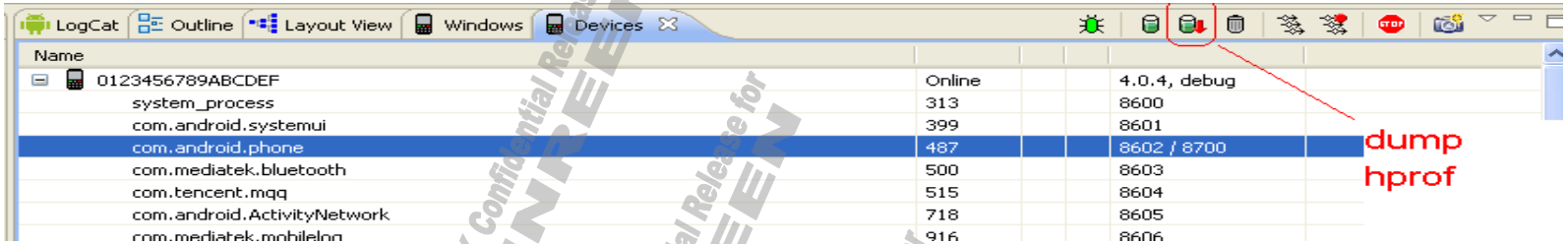
- Hprof file is the DVM heap dump file 。
- Product hprof file step:
  - adb shell ps get process pid，then adb shell kill -10 pid
  - Note: data/misc folder 如果没有写的权限，建议先adb shell chmod 777 /data/misc
  - Example： 产生com.mtk.MemoryLeakTest process 的hprof ，
    1. 使用 adb shell ps 得到pid 为 335



    2. 使用adb kill -10 335 产生hprof ，产生的文件在/data/misc目录下。

Java Heap OOM - Hprof

- Product hprof file step:
  - 使用eclipse 上的device 带的dump hprof 功能来dump。



  - 在mediatek/source/dalvik/vm/SignalCatcher.cpp

```
static void handleSigUsr1()
{
#if 1
    char buf[128];

     if (property_get("dalvik.vm.hprof", buf, "1")) {
        if (buf[0] == '0') {
            LOGD("SIGUSR1 forcing GC (no HPROF)\n");
            dvmCollectGarbage();
            return;
        }
    }

    LOGD("SIGUSR1 forcing GC and HPROF dump\n");

    sprintf(buf, "/data/misc/heap-dump-tm%d-pid%d.hprof",
                 (int) time(NULL), (int) getpid());
    hprofDumpHeap(buf, -1, false);

#else
    LOGD("SIGUSR1 forcing GC (no HPROF)");
    dvmCollectGarbage();
#endif
} ? end handleSigUsr1 ?
```

如果发kill -10 signal，打印这个log，那么这个dvm 不支持

Java Heap OOM - MAT Tool

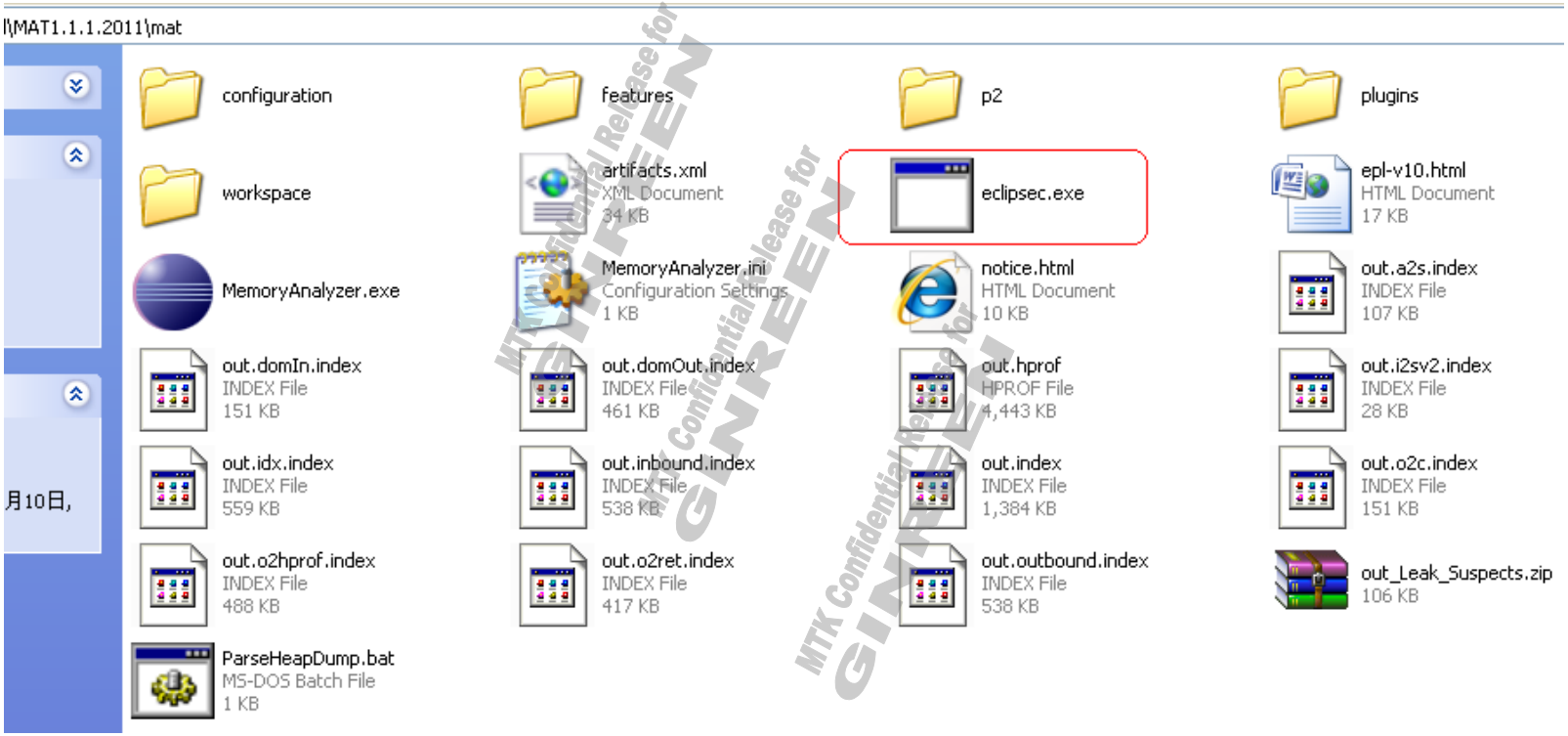
- The Eclipse Memory Analyzer is a fast and feature-rich **Java heap analyzer** that helps you find memory leaks and reduce memory consumption.
- MAT path:
  - http://www.eclipse.org/mat/downloads.php
  - http://teams.mediatek.inc/WCP/MSZ_WCP_SW/AF2/System%20Group/Useful%20Tools/MemoryAnalyzer-1.1.1.20110824-win32.win32.x86.zip

Java Heap OOM - Convet Hprof file

- hprof-conv.exe tool
  - Google 提供了该tool，在 android-sdk-windows\tools 下。
- 在命令行中输入command，就可以生成out.hprof：

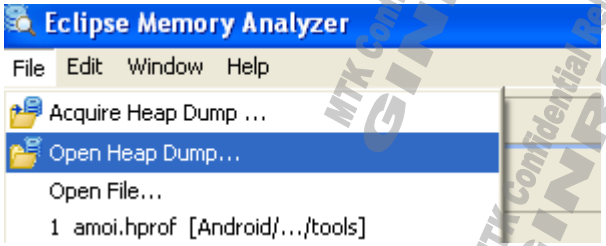
```
E:\Android\android-sdk-windows\tools>hprof-conv.exe heap-dump-tm1262305703-pid13
07.hprof out.hprof
```

Java Heap OOM - Start MAT



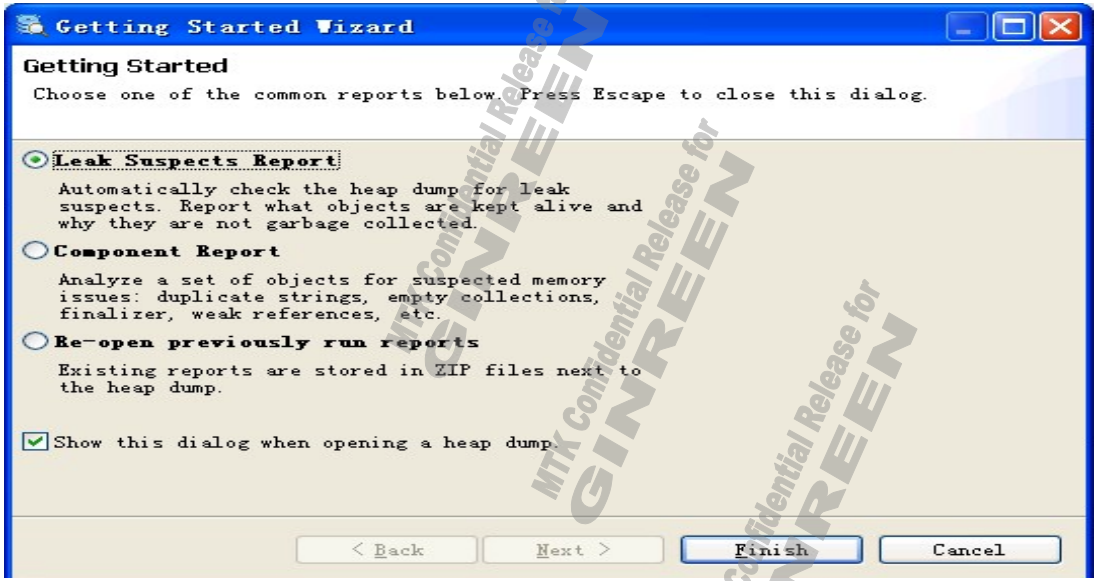
Java Heap OOM - Start MAT

- 启动mat ,选择File->Open Heap Dump 选择你的hprof dump文件。



Java Heap OOM - Start MAT

- 查看内存泄漏分析报表。mat解析完成以后会出现如下图的提示：

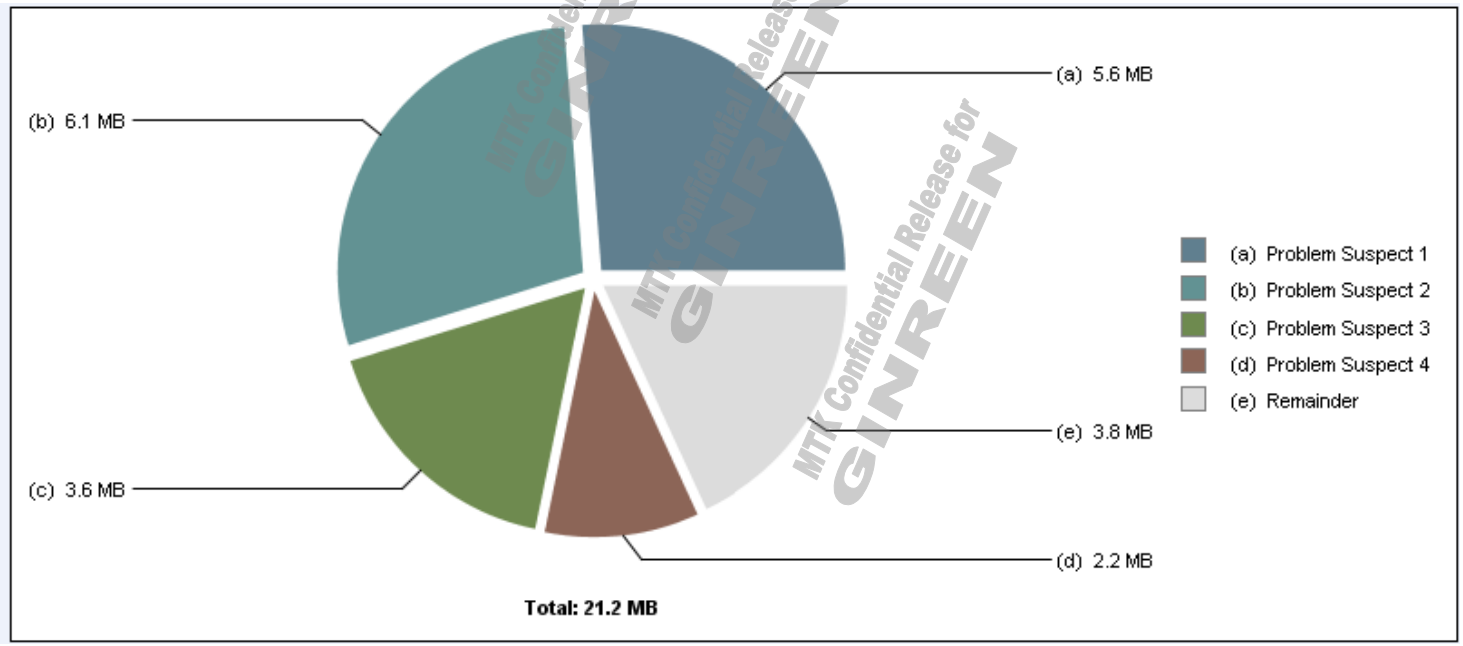


- 保持默认选项直接点“Finish”就可以。

Java Heap OOM - Start MAT

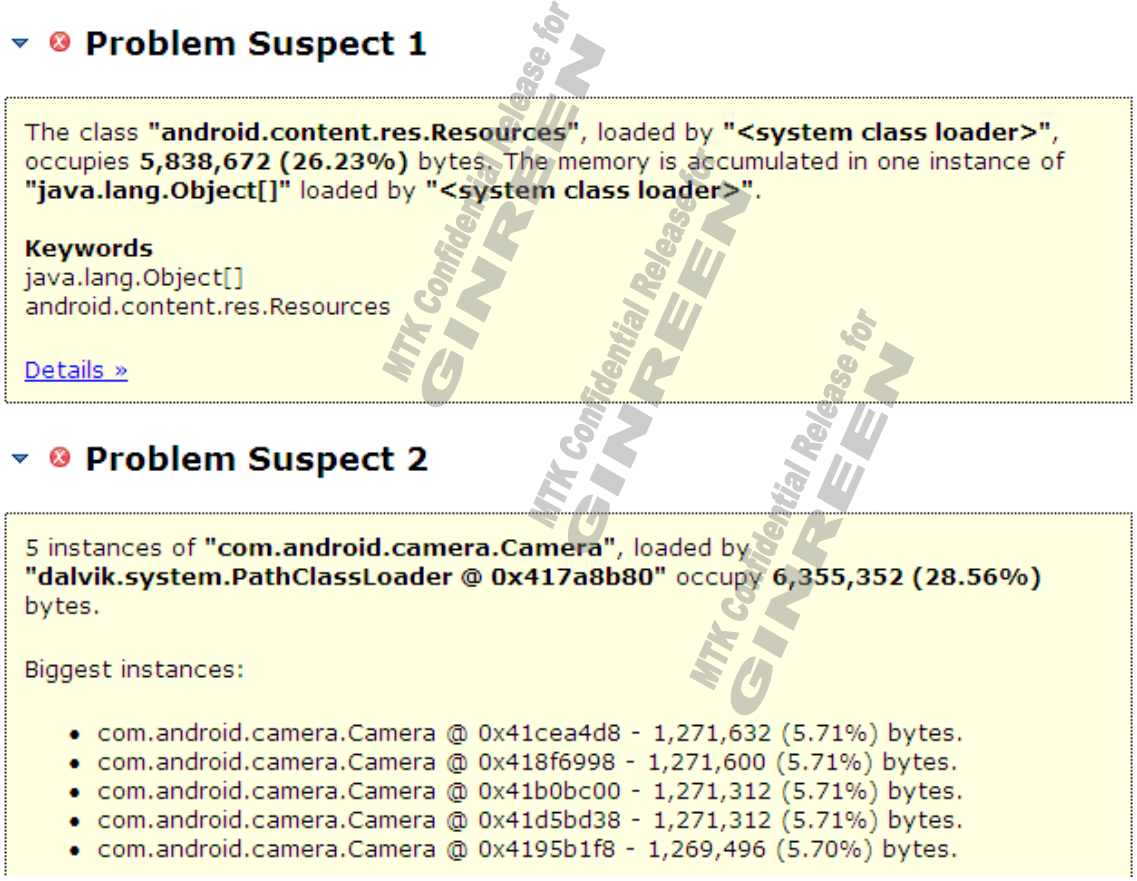
- 点Finish 后会出现饼状直观图：

Leak Suspects
System Overview
Leaks
Overview



Java Heap OOM - Start MAT

- 列出内存占用最大的几个对象：



Problem Suspect 1

The class **"android.content.res.Resources"**, loaded by **"<system class loader>"**, occupies **5,838,672 (26.23%)** bytes. The memory is accumulated in one instance of **"java.lang.Object[]"** loaded by **"<system class loader>"**.

Keywords
java.lang.Object[]
android.content.res.Resources

Details »

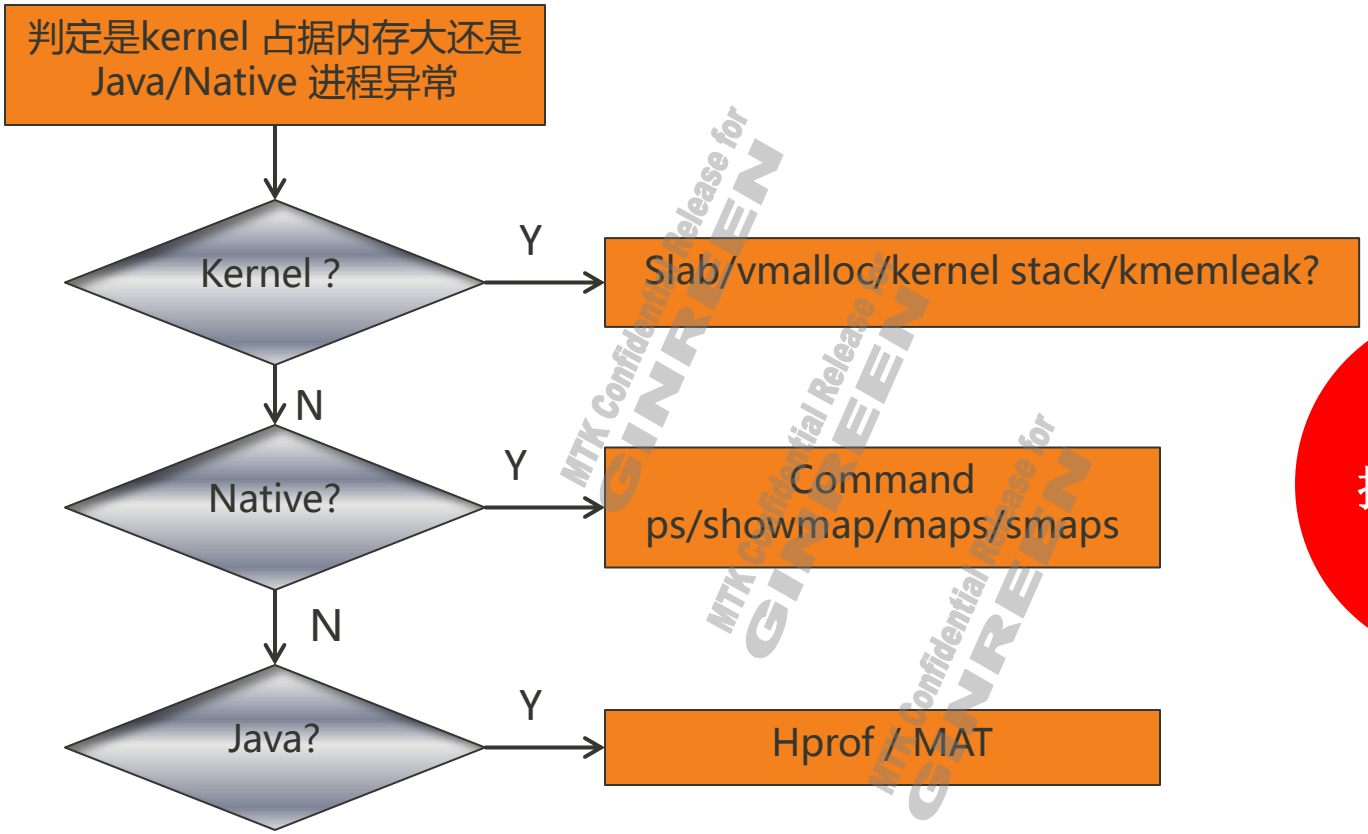
Problem Suspect 2

5 instances of **"com.android.camera.Camera"**, loaded by **"dalvik.system.PathClassLoader @ 0x417a8b80"** occupy **6,355,352 (28.56%)** bytes.

Biggest instances:

- com.android.camera.Camera @ 0x41cea4d8 - 1,271,632 (5.71%) bytes.
- com.android.camera.Camera @ 0x418f6998 - 1,271,600 (5.71%) bytes.
- com.android.camera.Camera @ 0x41b0bc00 - 1,271,312 (5.71%) bytes.
- com.android.camera.Camera @ 0x41d5bd38 - 1,271,312 (5.71%) bytes.
- com.android.camera.Camera @ 0x4195b1f8 - 1,269,496 (5.70%) bytes.

Summary



希望：
提供正确的
信息到MTK

Summary

- 请将下面的命令打包成.bat 抓取信息，user版本上面有些cmd抓取会fail，可先忽略

```
adb shell cat /proc/meminfo > meminfo.txt
adb shell dumpsys meminfo > dumpsysMemoryinfo.txt
adb shell cat /proc/zraminfo > zraminfo.txt
adb shell cat /proc/zoneinfo  > zoneinfo.txt
adb shell ps -t > ps.txt
adb shell procrank -s > procrank.txt
adb shell cat /sys/kernel/debug/ion/ion_mm_heap > ion_mm_heap.txt
adb shell cat /proc/mali/memory_usage > mali_memory_usage.txt
adb shell cat /d/pvr/driver_stats > pvr_gpu_status.txt
adb shell ls -l /d/pvr/pid > pvr_pid.txt
adb shell cat /d/pvr/pid/*/process_stats > pvr_pid_stats.txt
adb shell cat /proc/mtk_memcfg/memory_layout  > memory_layout.txt
adb shell cat /proc/vmallocinfo > vmallocinfo.txt
adb shell cat /proc/slabinfo > slabinfo.txt
adb shell cat /proc/buddyinfo > buddyinfo.txt
adb shell cat /sys/module/lowmemorykiller/parameters/adj > adj.txt
adb shell cat /sys/module/lowmemorykiller/parameters/minfree > minfree.txt
```

Background knowledge reference

- 《深入理解linux内核中文第三版》第八章、第九章、第十五章、第十七章