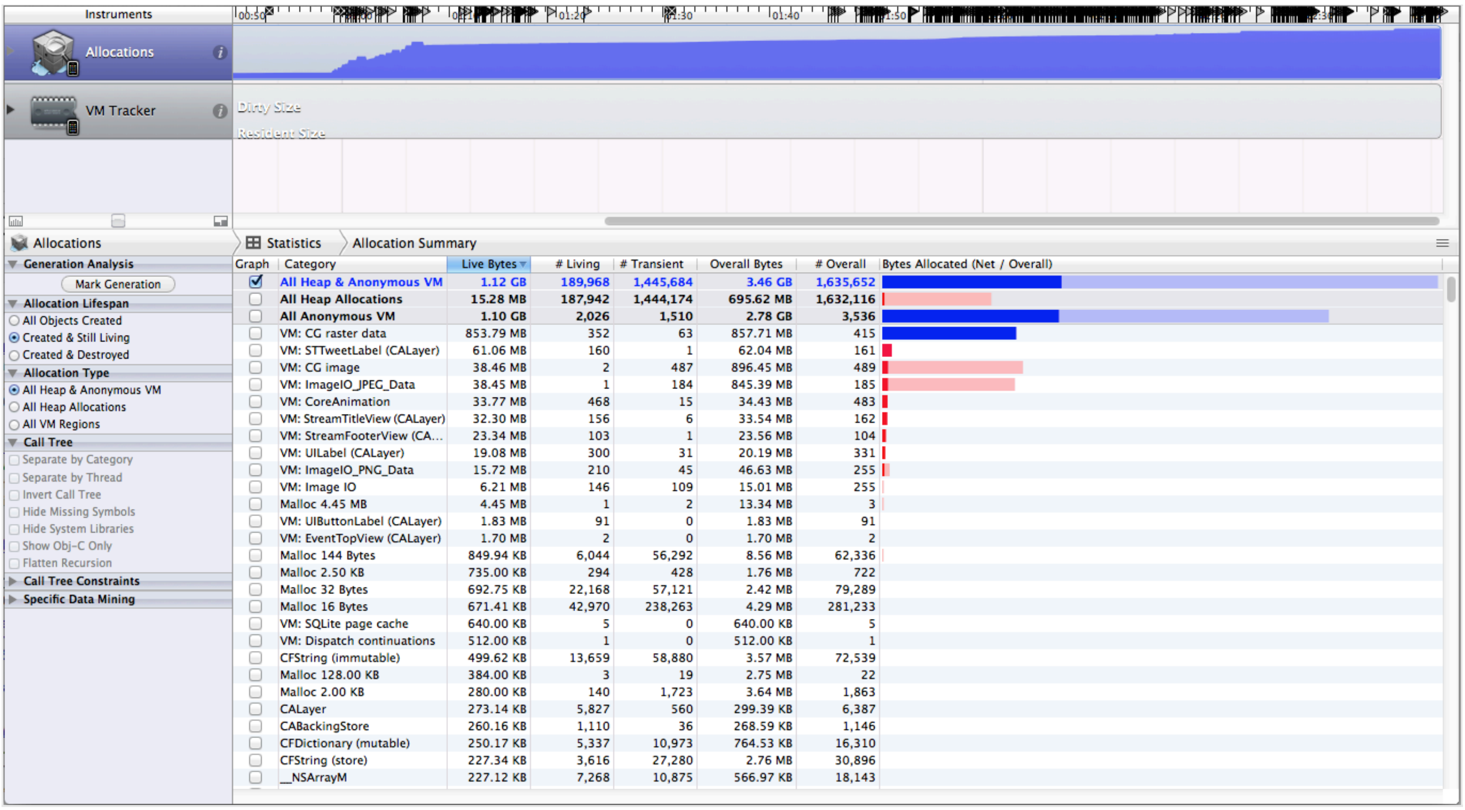
关于SDWebImage中的decodedimagewithimage引发的内存大量消耗的解决办法

Published by 江南的悲伤 on 一月 7, 2015

问题来源于SDWebImage这个库。使用这个库加载了网络图片之后，会将图片存到NSCache中去，然后再显示出来。但是在使用中，出现了一个最简单却又最粗暴的问题，内存的爆炸。在用它加载出图片了之后，内存会发生惊人的爆炸。通过Instrument的leaks可以看到在特定的某个图片的加载中内存爆炸了，这块空间的名字叫VM:CG raster data。百度了一下这个名字，只有一条记录，简直是逼我再也不用百度的节奏。后来在谷歌上一找就是源源不断的stackoverflow……
但是这一条记录就是最重要的了。
这就是SDWebImage的GitHub：
https://github.com/rs/SDWebImage/issues/538

这个项目本身是一个开源项目，所以当大多数人都遇到同一个问题的时候就会比较好解决一点。
在这里我看到了无数和我遇到了相同问题的同志们，有些甚至更加的夸张。有大部分使用这个库用来从加载GIF图，加载了两个GIF图就能达到1G左右的内存，当然后面我就知道这个bug（也可以不算是个bug）的厉害了……



Generation Analysis
Mark Generation

Allocation Lifespan
- ○ All Objects Created
- ● Created & Still Living
- ○ Created & Destroyed

Allocation Type
- ● All Heap & Anonymous VM
- ○ All Heap Allocations
- ○ All VM Regions

Call Tree
- ☐ Separate by Category
- ☐ Separate by Thread
- ☐ Invert Call Tree
- ☐ Hide Missing Symbols
- ☐ Hide System Libraries
- ☐ Show Obj-C Only
- ☐ Flatten Recursion

Call Tree Constraints

Specific Data Mining

Graph	Category	Live Bytes	# Living	# Transient	Overall Bytes	# Overall	Bytes Allocated (Net / Overall)
☑	**All Heap & Anonymous VM**	1.12 GB	189,968	1,445,684	3.46 GB	1,635,652	
☐	**All Heap Allocations**	15.28 MB	187,942	1,444,174	695.62 MB	1,632,116	
☐	**All Anonymous VM**	1.10 GB	2,026	1,510	2.78 GB	3,536	
☐	VM: CG raster data	853.79 MB	352	63	857.71 MB	415	
☐	VM: STTweetLabel (CALayer)	61.06 MB	160	1	62.04 MB	161	
☐	VM: CG image	38.46 MB	2	487	896.45 MB	489	
☐	VM: ImageIO_JPEG_Data	38.45 MB	1	184	845.39 MB	185	
☐	VM: CoreAnimation	33.77 MB	468	15	34.43 MB	483	
☐	VM: StreamTitleView (CALayer)	32.30 MB	156	6	33.54 MB	162	
☐	VM: StreamFooterView (CA…	23.34 MB	103	1	23.56 MB	104	
☐	VM: UILabel (CALayer)	19.08 MB	300	31	20.19 MB	331	
☐	VM: ImageIO_PNG_Data	15.72 MB	210	45	46.63 MB	255	
☐	VM: Image IO	6.21 MB	146	109	15.01 MB	255	
☐	Malloc 4.45 MB	4.45 MB	1	2	13.34 MB	3	
☐	VM: UIButtonLabel (CALayer)	1.83 MB	91	0	1.83 MB	91	
☐	VM: EventTopView (CALayer)	1.70 MB	2	0	1.70 MB	2	
☐	Malloc 144 Bytes	849.94 KB	6,044	56,292	8.56 MB	62,336	
☐	Malloc 2.50 KB	735.00 KB	294	428	1.76 MB	722	
☐	Malloc 32 Bytes	692.75 KB	22,168	57,121	2.42 MB	79,289	
☐	Malloc 16 Bytes	671.41 KB	42,970	238,263	4.29 MB	281,233	
☐	VM: SQLite page cache	640.00 KB	5	0	640.00 KB	5	
☐	VM: Dispatch continuations	512.00 KB	1	0	512.00 KB	1	
☐	CFString (immutable)	499.62 KB	13,659	58,880	3.57 MB	72,539	
☐	Malloc 128.00 KB	384.00 KB	3	19	2.75 MB	22	
☐	Malloc 2.00 KB	280.00 KB	140	1,723	3.64 MB	1,863	
☐	CALayer	273.14 KB	5,827	560	299.39 KB	6,387	
☐	CABackingStore	260.16 KB	1,110	36	268.59 KB	1,146	
☐	CFDictionary (mutable)	250.17 KB	5,337	10,973	764.53 KB	16,310	
☐	CFString (store)	227.34 KB	3,616	27,280	2.76 MB	30,896	
☐	__NSArrayM	227.12 KB	7,268	10,875	566.97 KB	18,143	

在最后的大神站出来解决问题之前，还有一个哥们的回复我觉得很有记录下来的必要：

Perhaps other people with the same problem could try what I tried. Keep in mind that I was using ARC.

Run the profiler with the Leaks template and check for the allocation of your own classes in the Allocation Summary.
Dive into them and check how they are allocated, if there are leaks. Mind that a leak doesn't show up in the Leaks instrument because you are using ARC. So Instruments could think everything is going oké but there still could be a leak somewhere. By diving in to the allocation of your own classes you could figure out what is going wrong.
Keep in mind that the retain/release count information is only provided when using the Leaks template and not while using the Allocations template.
My problem was that I was referencing instance variables and self directly from within blocks without reassigning them to __weak variables. When self is used within a block it will automatically be retained by ARC and sometimes never released. A weak reference prevents that from happening.

For example, this is wrong:

```
[[NSNotificationCenter defaultCenter]
addObserverForName:UIKeyboardDidShowNotification
usingBlock:^(NSNotification *note) {
[self.view setContentOffset:CGPointMake(0.0, kKeyboardOffset)
animated:YES];
```

You should call self using a __weak reference like this:

```
__weak YourViewControllerClass *weakSelf = self;
[[NSNotificationCenter defaultCenter]
addObserverForName:UIKeyboardDidShowNotification
usingBlock:^(NSNotification *note) {
    [weakSelf.view setContentOffset:CGPointMake(0.0,
kKeyboardOffset) animated:YES];
```

Since my app uses a lot of block's I had a ton of leaks the Leaks instrument could not detect. When I fixed them the memory problem was gone.

以上对于块的循环引用已经不新鲜了，主要是他说如果你正在使用ARC并且希望通过Leaks来找到泄露的地方，程序会认为一切泄露都是正常处理的。这也解释了灿哥之前用instrument为什么一直看不到泄露的原因。当然这可能只是一部分原因吧，关于**instrument的使用我会在之后写出详细的学习日志，阅读完官方的instrument guide document，那时候再找出真正的原因。**

最后，大神终于站出来拯救世界了，（虽然我不明白为什么这么久了才有大神站出来）

harishkashyap commented 16 days ago Its the memory issue again. decodedImageWithImage takes up huge memory and causes the app to crash. I have added an option to put this off in the library but defaulting to YES so there aren't any breaking changes. If you put off the decodeImageWithImage method in both image cache and image downloader then you shouldn't be seeing the VM: CG Raster data on the top consuming lots of memory

decodeImageWithImage is supposed to decompress images and cache them so the loading on tableviews/collectionviews become better. However, with large set of images being loaded, the experience worsened and the memory of uncompressed images even with thumbnails can consume GBs of memory. Putting this off only improved performance.

```
[[SDImageCache sharedImageCache] setShouldDecompressImages:NO];
[[SDWebImageDownloader sharedDownloader]
setShouldDecompressImages:NO];
```

首先二话不说，我把所有用到了decodedImageWithImage得地方都注释掉了，跑起来一看，果然VM：CG raster data那块已经不见了，内存的增长又回到了比较舒服的程度了。但是我还是觉得奇怪呀，这个函数不可能就这么废却还要存在呀，他说的意思大概是减压缩图片，并将图片存到cache使得之后的加载更加快，效果更加好。但是问题就在于去压缩这个操作，如果传进的图片分辨率特别的高，它的减压缩会消耗大量的内存，按照帖子其他回复的综合可以大概得出结论就是他会将图片的每一个像素点都有一个空间存下它的各个通道值，虽然这个内存空间是由于CG重绘的时候alloc出来的，所以是存在于VM里的空间。因此这样的处理会导致一个拇指大小的图片都可能消耗上GB得内存。我也是醉了。
几百K的图片，分辨率达到3000+*2000+，就多消耗了40M内存，如果是GIF图，又是帧数比较高，分辨率比较高的，就会出现一个GIF图500M甚至上GB得奇葩现象==（我忽然觉得可以在处理之前处理一下分辨率的问题，但是我又觉得这个方法只是对于显示有帮助，缓存同样还是缓存下来了，貌似是只有在retina屏幕的设备上才能看出这个的区别……）
另外中间有个小插曲，在一开始的profile过程中，灿哥那里用AM看内存消耗大概是60M左右（爆炸后），用Leaks看内存就到100M了，这两者显示的内存不同，也是一个问题。

http://stackoverflow.com/questions/4596037/why-is-there-a-difference-between-the-reported-memory-usage-of-an-app-by-activit”

> Activity Monitor is useless for development/debugging purposes. AM is only useful if when you don’t have Instruments running already & you see the RPRVT growing over time significantly. Even then, it is just a symptom and may not indicate a real problem.

> AM is sort of summarizing a set of different memory related numbers. It is a very rough number. The Allocations instrument is summarizing exactly the set of allocations in your application (which, under Mac OS X, could include both GC and non-GC allocations). Reduce the allocations and the overall memory use will generally go down.
>
> Note that a system not under memory pressure will often not request that applications give back memory. That is, you may not see a drop in Activity Monitors numbers.

http://www.cocoachina.com/bbs/read.php?tid=266974

> activity monitor 是程序在手机运行真正占用的内存大小，达到系统内存的一半左右，就很可能被系统杀掉。建议检查程序中耗用内存大的地方。allocations 个人认为是alloc/new所产生的内存大小。

这段话的意思大致是AM就是不准确的，只是用来看大概监控的，也没法定位内存问题，只有allocation或者leaks可以看到程序中每一块内存空间的实时情况，另外也包括了Virtual Memory，所以可能会比AM里看到的值更大。关于Virtual Mem和Real Mem的问题还会继续跟进吧，其实真正运行时的内存消耗应当是以AM里的为准，VM很多并不是在内存中开辟出来的空间，是系统在存储空间里为内存开辟出来的空间，类似PC的虚拟内存。

Plus.（待研究）记录一个小bug得解决办法：
Xcode6.1运行程序后，左侧Debug区域的Memory显示空白
解决办法：打开左侧暂停符号右边的那个下拉列表里面的editSCheme 里面有个选项叫叫做enable zoombie Objects 取消选中 ok
（关掉僵尸的其他麻烦就没办法了，不过我觉得在这里看到内存的监控也不是特别重要）