

## Linux Used内存到底哪里去了?

January 19th, 2013

Yu Feng

[Go to comments](#)[Leave a comment](#)

原创文章，转载请注明： 转载自[系统技术非业余研究](#)

本文链接地址: [Linux Used内存到底哪里去了?](#)

前几天 纯上 同学问了一个问题:

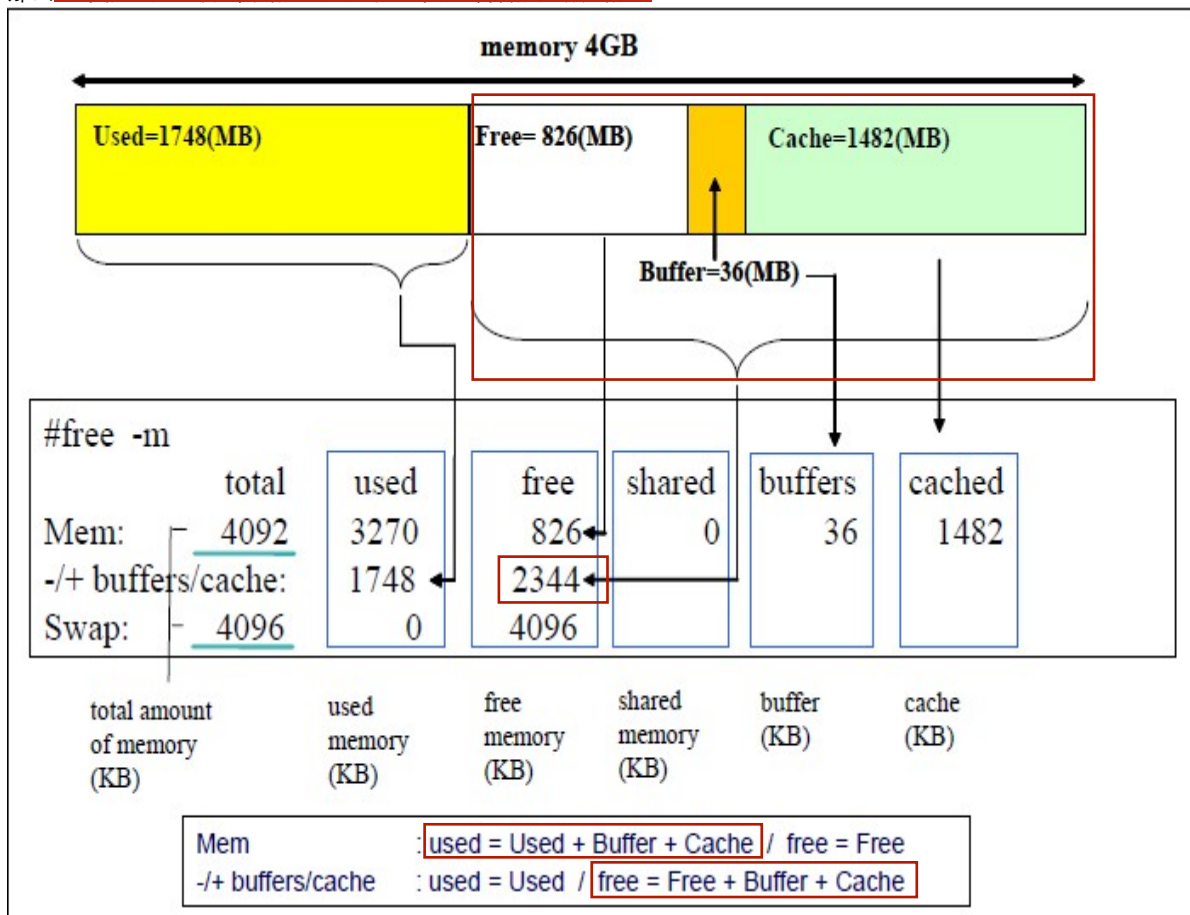
我ps aux看到的RSS内存只有不到30M, 但是free看到内存却已经使用了7.8G了, 已经开始swap了, 请问ps aux的实际物理内存统计是不是漏了哪些内存没算? 我有什么办法确定free中used的内存都去哪儿了呢?

这个问题不止一个同学遇到过了, 之前子嘉同学也遇到这个问题, 内存的计算总是一个迷糊账。我们今天来把它算个清楚下!

通常我们是这样看内存的剩余情况的:

```
$ free -m
              total        used        free      shared    buffers     cached
Mem:           48262         7913        40349           0          14        267
-/+ buffers/cache:        7631        40631
Swap:           2047           336         1711
```

那么这个信息是如何解读的呢, 以下这个图解释的挺清楚的!



补充 (不少人反映图不清晰, 请参考: <http://www.redbooks.ibm.com/redpapers/pdfs/redp4285.pdf> P46-47)

上面的情况下我们总的内存有48262M, 用掉了7913M。其中buffer+cache总共14+267=281M, 由于这种类型的内存是可以回收的, 虽然我们用掉了7913M, 但是实际上我们如果实在需要的话, 这部分buffer/cache内存是可以放出来的。

我们来演示下：

```
$ sudo sysctl vm.drop_caches=3
vm.drop_caches = 3
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	48262	7676	40586	0	3	41
-/+ buffers/cache:		7631	40631			
Swap:	2047	336	1711			

我们把buffer/cache大部分都清除干净了，只用了44M，所以我们这次used的空间是7676M。

到现在我们比较清楚几个概念：

1. 总的内存多少
2. buffer/cache内存可以释放的。
3. used的内存的概率。

即使是这样我们还是要继续追查下used的空间（7637M）到底用到哪里去了？

这里首先我们来介绍下nmon这个工具，它对内存的使用显示比较直观。

```

Total MB      RAM      High      Low      Swap      Page Size=4 KB
Free MB       40543.7    -0.0     -0.0     1711.6
Free Percent   84.0%     100.0%   100.0%   83.6%

MB                                     MB                                     MB
Buffers=       7.7    Cached=       72.1    Active= 5405.6
Dirty  =       6.8    Swapcached= 18.5    Inactive = 1175.3
Slab   = 913.8    Writeback = 0.0    Mapped  = 36.6
Commit_AS = 23015.0 PageTables= 56.3
AMS is not active
  
```

使用的内存的去向我们很自然的就想到操作系统系统上的各种进程需要消耗各种内存，我们透过top工具来看下：

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
29168	root	20	0	1597m	89m	1436	S	19.4	0.2	5021:25	beam.smp
8571	admin	20	0	4993m	1.4g	3792	S	7.7	3.0	988:32.93	beam.smp

通常我们会看进程的RES这一项，这项到底是什么意思呢？这个数字从哪里出来的呢？通过strace对top和nmon的追踪和结合源码，我们确定这个值是从/proc/PID/statm的第二个字段读取出来的。

那这个字段什么意思呢？

man proc或者<http://www.kernel.org/doc/man-pages/online/pages/man5/proc.5.html> 会详细的解释/proc/下的文件的具体意思，我们摘抄下：

```

/proc/[pid]/statm
Provides information about memory usage, measured in pages. The
columns are:

size total program size
(same as VmSize in /proc/[pid]/status)
resident resident set size
(same as VmRSS in /proc/[pid]/status)
share shared pages (from shared mappings)
text text (code)
lib library (unused in Linux 2.6)
data data + stack
dt dirty pages (unused in Linux 2.6)
  
```

resident set size 也就是每个进程用了具体的多少页的内存。由于linux系统采用的是虚拟内存，进程的代码，库，堆和栈

使用的内存都会消耗内存，但是申请出来的内存，只要没真正touch过，是不算的，因为没有真正为之分配物理页面。

我们实际进程使用的物理页面应该用resident set size来算的，遍历所有的进程，就可以知道所有的所有的进程使用的内存。

我们来实验下RSS的使用情况：

```
$ cat RSS.sh
#/bin/bash
for PROC in `ls /proc/ | grep "[0-9]"`
do
    if [ -f /proc/$PROC/statm ]; then
        TEP=`cat /proc/$PROC/statm | awk '{print ($2)}'`
        RSS=`expr $RSS + $TEP`
    fi
done
RSS=`expr $RSS \* 4`
echo $RSS"KB"
$ ./RSS.sh
7024692KB
```

从数字来看，我们的进程使用了大概7024M内存，距离7637M还有几百M内存哪里去了？哪里去了？猫吃掉了？我们再回头来仔细看下nmon的内存统计表。

	RAM	High	Low	Swap	Page Size=4 KB
Total MB	48262.7	-0.0	-0.0	2048.0	
Free MB	40194.7	-0.0	-0.0	1711.6	
Free Percent	83.3%	100.0%	100.0%	83.6%	
	MB		MB		MB
Buffers=	15.6	Cached=	400.6	Active=	5727.7
Dirty =	5.3	Swpcached=	18.5	Inactive =	1200.2
Slab =	915.7	Writeback =	0.0	Mapped =	36.6
		Commit_AS =	23018.8	PageTables=	56.8
AMS is not active					

那个该死的slab是什么呢？那个PageTables又是什么呢？

简单的说内核为了高性能每个需要重复使用的对象都会有个池，这个slab池会cache大量常用的对象，所以会消耗大量的内存。运行命令：

```
$ slabtop
```

我们可以看到：

slabtop

从图我们可以看出各种对象的大小和数目，遗憾的是没有告诉我们slab消耗了多少内存。

我们自己来算下好了：

```
$ echo `cat /proc/slabinfo | awk 'BEGIN{sum=0;}{sum=sum+$3*$4;}END{print sum/1024/1024}'`
MB
904.256 MB
```

好吧，把每个对象的数目\*大小，再累加，我们就得到了总的内存消耗量:904M

那么PageTables呢？我们万能的内核组的同学现身了：

伯瑜:

你还没有计算page tables的大小, 还有struct page也有一定的大小(每个页一个, 64bytes), 如果是2.6.32的话, 每个页还有一个page\_cgroup(32bytes), 也就是说内存大小的2.3%(96/4096)会被内核固定使用的含黛:

struct page是系统boot的时候就会根据内存大小算出来分配出去的, 18内核是1.56%左右, 32内核由于cgroup的原因会在2.3%

好吧, 知道是干嘛的啦, 管理这些物理页面的硬开销, 那么具体是多少呢?

```
$ echo `grep PageTables /proc/meminfo | awk '{print $2}'` KB
58052 KB
```

好吧, 小结下! 内存的去向主要有3个: 1. 进程消耗. 2. slab消耗 3. pagetable消耗.  
我把三种消耗汇总下和free出的结果比对下, 这个脚本的各种计算项仲同学帮忙搞定的:

```
$ cat cm.sh
#!/bin/bash
for PROC in `ls /proc/ | grep "^[0-9]"`
do
    if [ -f /proc/$PROC/statm ]; then
        TEP=`cat /proc/$PROC/statm | awk '{print ($2)}'`
        RSS=`expr $RSS + $TEP`
    fi
done
RSS=`expr $RSS \* 4`
PageTable=`grep PageTables /proc/meminfo | awk '{print $2}'`
SlabInfo=`cat /proc/slabinfo | awk 'BEGIN{sum=0;} {sum=sum+$3*$4;} END{print sum/1024 /1024}'`

echo $RSS"KB", $PageTable"KB", $SlabInfo"MB"
printf "rss+pagetable+slabinfo=%sMB\n" `echo $RSS/1024 + $PageTable/1024 + $SlabInfo|bc`
free -m

$ ./cm.sh
7003756KB, 59272KB, 904.334MB
rss+pagetable+slabinfo=7800.334MB
Mem:          total      used      free      shared    buffers    cached
-/+ buffers/cache:  7629      40633
Swap:         2047        336       1711
```

free报告说7629M, 我们的cm脚本报告说7800.3M, 我们的CM多报了171M。

damn,这又怎么回事呢?

我们重新校对下我们的计算。我们和nmon来比对下, slab和pagetable的值是吻合的。那最大的问题可能在进程的消耗计算上。

resident resident set size 包括我们使用的各种库和so等共享的模块, 在前面的计算中我们重复计算了。

```
$ pmap `pgrep bash`
...
22923:  -bash
0000000000400000    848K r-x--  /bin/bash
00000000006d3000    40K rw---  /bin/bash
00000000006dd000    20K rw---  [ anon ]
00000000008dc000    36K rw---  /bin/bash
00000000013c8000   592K rw---  [ anon ]
000000335c400000   116K r-x--  /lib64/libtinfo.so.5.7
...
0000003ec5220000     4K rw---  /lib64/ld-2.12.so
0000003ec5221000     4K rw---  [ anon ]
0000003ec5800000  1628K r-x--  /lib64/libc-2.12.so
...
0000003ec5b9c000    20K rw---  [ anon ]
00007f331b910000  96836K r----  /usr/lib/locale/locale-archive
00007f33217a1000    48K r-x--  /lib64/libnss_files-2.12.so
...
```

```
00007f33219af000      12K rw---    [ anon ]
00007f33219bf000       8K rw---    [ anon ]
00007f33219c1000     28K r--s-  /usr/lib64/gconv/gconv-modules.cache
00007f33219c8000       4K rw---    [ anon ]
00007fff5e553000     84K rw---    [ stack ]
00007fff5e5e4000       4K r-x--    [ anon ]
fffffffffff60000       4K r-x--    [ anon ]
total                108720K
```

多出的171M正是共享库重复计算的部分。  
但是由于每个进程共享的东西都不一样，我们也没法知道每个进程是如何共享的，没法做到准确的区分。  
所以只能留点小遗憾，欢迎大家来探讨。  
总结：内存方面的概念很多，需要深入挖掘！  
祝玩的开心！

[Leave a comment](#)

[Trackback](#)

Trackbacks (5)


Comments (26)


  
haofish

January 19th, 2013 at 21:54 | [#4](#) [Reply](#) | [Quote](#)

关于重复计算的部分，还有可能是共享内存占用了，这种可以在/proc/\$pid/smaps 里面可以看到（匿名的那些）

[\[Reply\]](#)

  
**Yu Feng Reply:**  
January 19th, 2013 at 9:55 pm  
多谢指点！  
[\[Reply\]](#)

  
**homer Reply:**  
January 21st, 2013 at 10:53 am  
多谢，我一直没搞明白，smaps里面没名字的是啥东东。  
[\[Reply\]](#)

  
**Yu Feng Reply:**  
January 21st, 2013 at 11:53 am  
这块应该没法算。  
[\[Reply\]](#)

  
吴洪辉

January 19th, 2013 at 23:52 | [#6](#) [Reply](#) | [Quote](#)

[@haofish](#)

冒泡一下，完全正确，smap里面这些信息可以提供进程使用的共享库的内存页面信息。不过要精确统计还是略困难的。。。再说脚本运行也有开开关关进程的，有影响。

Rss: 4 kB

Pss: 4 kB

Shared\_Clean: 0 kB

Shared\_Dirty: 0 kB

Private\_Clean: 0 kB

Private\_Dirty: 4 kB

[\[Reply\]](#)



*Yu Feng* Reply:

January 19th, 2013 at 11:53 pm

对的。

[\[Reply\]](#)