进程分析之内存

在服务端编程，对进程的内存占用情况的了解和日常监控显得尤为重要，特别是内存存储密集型进程，比如redis，namenode等，疏忽对内存的分析，肯定会酿成大的线上事故；

入门篇之free命令解析

free命令可以了解当前操作系统整体可用内存的使用情况，是分析的第一步：

```
             total       used       free     shared    buffers     cached
Mem:      49406140   48352948    1053192          0      96524   35946452
-/+ buffers/cache:   12309972   37096168
Swap:            0          0          0
```

在free命令可以看出我们当前物理内存的使用情况，其中total：49406140即为总的物理内存大小，used：48352948为当前被**系统调度**的内存大小，而free：1053192即当前未被**系统调度**的内存大小，注意，这里用的词是被**系统调度**，而不是被系统进程物理占用的内存大小；

怎么理解呢？对于上面的case，free只有100M，如果used理解为被物理占用的内存，那么就只有100M可以留给我新的进程使用？这个理解是不正确的；used只是被系统调度的内存，系统可以从used里面划分出有效的内存来调度给新进程使用，只有在可调度内存不够的情况下，才会从free中挪用新内存进行调度；

used内存会被系统用来做哪些调度？

1. 实实在在被进程占用，比如进程申请的栈内存，**被分配并被使用**的堆内存（这里强调一下分配并被使用，通过new/malloc分配的进程内存空间不一定实实在在占用了物理内存，详情见后面的分析），在进程存活并未主动释放的case下，这块内存是无法被系统调度给其他进程使用，即第三行used=12309972部分；
2. 用于写buffer，即上面buffers:96524,对于文件写操作需要经过写buffer的过程再落盘到磁盘上，而buffer这块内存是可以被系统回收并反复被调度的内存；
3. 用于读cached，即上面的cached：35946452，系统会将一大部分内存用于cache，比如用于文件预读等，与buffer相同，这部分内存可以被系统重复调度给信息的进程；

因此，一个系统被调度内存大小=buffers+cached+第三行的物理used；可用于新进程的内存大小=free+buffers+cached；而这个数字即为上面第三行free=37096168；第三行表示当前物理内存被系统物理占用的内存，used=12309972，以及可以被调度的内存大小free；而这里free+used即为total物理内存大小；在日常运维过程中，可以通过free命令的第三行的free部分来快速获取机器物理内存被实际占用情况

free,buffer，cached这种值也可以通过vmstat命令来动态获取它的变化值，比如每隔一秒采集一次，总共采集十次memory的变化；

```
[work@# ~]$ vmstat 1 10
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 463664  98752 36521052    0    0   157   272    1    1  4  6 90  0  0
 2  0      0 462780  98752 36521060    0    0     0    12 21790 17344  3  6 90  0  0
 2  0      0 458672  98752 36521076    0    0     0    20 22598 18355  3  6 90  0  0
 2  0      0 439960  98752 36521076    0    0     0    84 22457 18133  4  8 88  0  0
 2  0      0 447608  98752 36521084    0    0     0    28 22618 17794  4  6 90  0  0
 2  0      0 443508  98752 36521100    0    0     0    24 22736 41260  6  8 87  0  0
 1  0      0 444012  98752 36521064    0    0     0    52 22733 18433  3  7 90  0  0
 2  0      0 450972  98752 36521064    0    0     0     0 22269 17790  3  6 91  0  0
 4  0      0 452824  98752 36521076    0    0     0    52 22661 18445  3  6 91  0  0
 3  0      0 453120  98752 36521092    0    0     0     8 21966 17457  4  6 90  0  0
```

再推荐一个命令：watch，与free命令配合，也可以获取到动态的内存变化值，case：watch -n 1 -d free

> 总结：从上面，我们可以看出当前系统物理被占用12G，可用与新的进程的内存高达37G；

入门篇之top命令解析

top不仅仅用于内存分析，还用于进程分析，今天只写关于内存部分；

```
top - 21:41:22 up 38 days,  8:38,  1 user,  load average: 0.14, 0.20, 0.25
Tasks: 320 total,   1 running, 314 sleeping,   5 stopped,   0 zombie
Cpu(s):  3.2%us,  6.6%sy,  0.6%ni, 89.1%id,  0.0%wa,  0.0%hi,  0.4%si,  0.0%st
Mem:  49406140k total, 48392788k used,  1013352k free,    96584k buffers
Swap:        0k total,        0k used,        0k free, 35985072k cached
PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20811 work      20   0 18.0g 1.3g 6992 S 13.9  2.8   2318:28 scribed
```

其中mem部分的total，used，free，buffers，cached与free命令显示的含义是一致的，他们都反应了整个系统的内存消耗情况；

下面我们需要从top命令开始针对单进程的内存占用情况进行分析，如pid=20811

我们知道，在linux操作系统，每个进程都独立占用一个完整的“虚拟逻辑内存”大小，对于32位系统，那么每个进程可以使用的虚拟内存大小为2^32=4G内存；注意，这是虚拟内存，而不是物理内存；

> 在linux c编程时，通过malloc分配一块内存，并返回一个逻辑地址，这里分配的内存即为虚拟内存，它的大小可以超过物理内存大小；只有我们对分配的内存进行写（memset等）才会真正触发物理内存的分配；

上面的VIRT=18g，即为虚拟内存占用情况，很大吧！其实不用担心，RES=1.3g才是物理分配的内存大小，scribed进程我们没有合理设置相应的参数，导致VIRT和RES相差较大；%MEM=2.8就需要关心点了，它表示当前进程占用的物理内存比例，即（1.3G/46G=2.8%）；

SHR=6992共享内存区大小；在linux系统中进程之间是可以做内存共享，比如一些系统级别的so，只需要加载一次，就可以被所有的进程共享，在不涉及大的进程内存共享的应用程序，基本不需要关注这部分内存的大小；

> free，vmstat，top等命令可以获取当前服务器当前swap的大小；swap空间是一部分磁盘空间，相比内存来说，性能要差很多，但是在机器内存不够的情况下，swap是保证系统正常运行的最后一段关卡了；在系统里，是通过一个swappiness的值来控制对swap使用。如果swappiness=0的时候表示最大限度使用物理内存，才物理内存为0的时间才开始使用swap空间，swappiness＝100的时候表示积极的使用swap分区，并且把内存上的数据及时搬运到swap空间中。linux的基本默认设置为60（/proc/sys/vm/swappiness=60）；也就是说，你的内存在使用到100-60=40%的时候，就开始出现有swap的使用。大家知道，内存的速度会比磁盘快很多，这样子会加大系统io，同时造的成大量页的换进换出，严重影响系统的性能，所以我们在操作系统层面，要尽可能使用内存，对该参数进行调整。
>
> 目前在使用的服务器上基本都是关闭swap，通过人工运维来合理分配每个机器上最大可使用内存，避免swap对服务的性能造成影响；
>
> 如果swap没有关闭，通过vmstat看到大量的swap in/out就代表系统已经发生swap，服务的性能肯定已经受到影响

中间篇之系统历史内存使用情况解析sar命令

free，vmstat，top等命令都只能获取到系统当前的内存使用情况，但是在一些线上问题跟踪过程中，需要对历史的系统压力等case进行跟进，这个时候就需要去获取机器历史内存/IO/CPU等使用情况，这里我们就针对sar命令对历史内存使用进行分析；

sar -r 参数就可以获取到以10分钟为间隔报告自午夜起当天的内存使用情况

```
[work@# ~]$ sar -r
Linux 2.6.32_1-16-0-0 (nj01-pcsdata-b05.nj01.baidu.com)         11/28/2015      _x86_64_
(12 CPU)
12:00:01 AM kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit
12:10:01 AM    915200  48490940     98.15     96800  36076476  16877612     34.16
12:20:01 AM    895300  48510840     98.19     96816  36085924  16876592     34.16
12:30:01 AM    943304  48462836     98.09     97092  36048648  16877212     34.16
12:40:01 AM    932804  48473336     98.11     97100  36056224  16876204     34.16
12:50:01 AM    928852  48477288     98.12     97116  36059544  16876184     34.16
01:00:01 AM    918804  48487336     98.14     97116  36066656  16875120     34.16
01:10:01 AM    893916  48512224     98.19     97132  36073808  16876188     34.16
01:20:01 AM    865428  48540712     98.25     97148  36089332  16876404     34.16
```

kbmemfree kbmemused %memused kbbuffers kbcached这几个参数都比较容易理解，和上面free等命令获取的含义一直，但是这里有一个新的名词，kbcommit和%commit分别表示应用程序当前使用的内存大小和使用百分比。其实和free命令里面的第三行的used含义差不多，只是计算的方式不同而已；

中级篇之进程status内存相关分析

通过top -p 进程号可以获取到进程的当前内存使用情况，但是比较抽象，对进程内存使用情况最完美的解析就是直接去阅读进程status内存，比如，下面带中文注释的即为内存相关的数据信息；

```
[work@# ~]$ cat /proc/20811/status
Name:   scribed
State:  S (sleeping)
Tgid:   20811
Pid:    20811
PPid:   1
TracerPid:      0
Uid:    500     500     500     500
Gid:    501     501     501     501
FDSize: 8192
Groups: 501
VmPeak: 20531460 kB  //进程分配的虚拟内存峰值
VmSize: 18875052 kB  //进程当前分配的虚拟内存大小
VmLck:         0 kB  //进程当前加锁的内存大小，参考linux mlock
VmHWM:   1943192 kB  //进程使用的物理内存峰值
VmRSS:   1357056 kB  //进程当前使用的物理内存大小
VmAnon:  1350016 kB  //匿名page大小，有些内存比如so占用内存都是有名称，参考meminfo的功能
VmFile:     7040 kB  //so等文件占用内存大小，VmAnon+VmFile=VmRSS
VmData: 18711392 kB  //进行数据段占用的虚拟内存大小
VmStk:       120 kB  //进程堆栈段的虚拟内存大小
VmExe:      3652 kB  //进程代码段的虚拟内存大小
VmLib:     12968 kB  //进程所使用LIB库的虚拟内存大小
VmPTE:      4584 kB  //占用的页表的大小.
VmSwap:        0 kB  //交换分区大小
Threads:        56
SigQ:   0/385964
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000003
SigCgt: 1000000181005ccc
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed:   fff
Cpus_allowed_list:      0-11
Mems_allowed:   00000000,00000003 //
Mems_allowed_list:      0-1 //表示进程可以使用的内存段，有点高端，就不解析含义了
voluntary_ctxt_switches:        3264080472
nonvoluntary_ctxt_switches:     794416
```

中级篇之系统meminfo内存相关分析

和进程status文件一样，对系统内存的了解除了通过命令来获取以外，还可以直接对系统的meminfo文件进行阅读来了解当前系统的内存使用情况，哈哈，虽然意义不是特别大。

```
[work@# ~]$ cat /proc/meminfo
MemTotal:       49406140 kB //直译
MemFree:          410128 kB //直译
Buffers:           98848 kB //直译
Cached:         36568808 kB //直译
SwapCached:            0 kB //直译
Active:         26052152 kB //最近经常被使用的内存，除非非常必要否则不会被移作他用.
Inactive:       21128908 kB //最近不经常被使用的内存，非常用可能被用于其他途径.
Active(anon):   10577508 kB //匿名活动page
Inactive(anon):   396316 kB //匿名非活动page
Active(file):   15474644 kB //文件活动page，Active= Active(anon) + Active(file)
Inactive(file): 20732592 kB //文件非文件page，Inactive=Inactive(anon) + Inactive(file)
Unevictable:        1892 kB
Mlocked:               0 kB
SwapTotal:             0 kB //直译
SwapFree:              0 kB //直译
Dirty:               584 kB //脏页内存大小，页更新但是没有刷到文件的内存大小
Readahead:             0 kB
Writeback:             0 kB
AnonPages:      10515528 kB
Mapped:           114204 kB //文件map映射占用内存大小
Shmem:            458524 kB //共享内存占用的内存大小
Slab:            1214064 kB
SReclaimable:    1163692 kB
SUnreclaim:        50372 kB
KernelStack:        8792 kB
PageTables:        37736 kB //页表大小
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    24703068 kB
Committed_AS:   16887880 kB
VmallocTotal:   34359738367 kB
VmallocUsed:      380668 kB
VmallocChunk:   34332399224 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:        6152 kB
DirectMap2M:    50305024 kB
```

> 关于匿名内存和文件内存的解析：
>
> 匿名内存一般是程序内存通过malloc等分配的没有名称的内存，而文件内存>> 即为so，jar等库文件占用的内存，通过pmap 进程号 可以直接看出每个进程的匿名内存和文件内存
> 00007f34243f9000 4K rw--- /lib64/libnss_files-2.12.so 00007f34243fa000 12K ----- [anon]