

free

显示内存的使用情况

补充说明

free命令 可以显示当前系统未使用的和已使用的内存数目，还可以显示被内核使用的内存缓冲区。

语法

```
free(选项)
```

选项

- b # 以Byte为单位显示内存使用情况；
- k # 以KB为单位显示内存使用情况；
- m # 以MB为单位显示内存使用情况；
- g # 以GB为单位显示内存使用情况。
- o # 不显示缓冲区调节列；
- s<间隔秒数> # 持续观察内存使用状况；
- t # 显示内存总和列；
- V # 显示版本信息。

实例

```
free -t      # 以总和的形式显示内存的使用信息
free -s 10    # 周期性的查询内存使用信息，每10s 执行一次命令
```

显示内存使用情况

```
free -m
```

	total	used	free	shared	buffers	cached
Mem:	2016	1973	42	0	163	1497
-/+ buffers/cache:		312	1703			
Swap:	4094	0	4094			

第一部分Mem行解释：

- total[]内存总数；
- used[]已经使用的内存数；
- free[]空闲的内存数；
- shared[]当前已经废弃不用；
- buffers Buffer[]缓存内存数；
- cached Page[]缓存内存数。

关系[]total = used + free

第二部分(-/+ buffers/cache)解释:

(-buffers/cache) used内存数：第一部分Mem行中的 used – buffers – cached

(+buffers/cache) free内存数: 第一部分Mem行中的 free + buffers + cached

可见-buffers/cache反映的是被程序实实在在吃掉的内存，而+buffers/cache反映的是可以挪用的内存总数。

第三部分是指交换分区。

输出结果的第四行是交换分区SWAP的，也就是我们通常所说的虚拟内存。区别：第二行(mem)的used/free与第三行(-/+ buffers/cache) used/free的区别。这两个的区别在于使用的角度来看，第一行是从OS的角度来看，因为对于OS buffers/cached 都是属于被使用，所以他的可用内存是2098428KB,已用内存是30841684KB,其中包括，内核 OS 使用+Application(X, oracle,etc)使用的+buffers+cached.

第三行所指的是从应用程序角度来看，对于应用程序来说 buffers/cached 是等于可用的，因为buffer/cached是为了提高文件读取的性能，当应用程序需在用到内存的时候 buffer/cached会很快地被回收。

所以从应用程序的角度来说，可用内存=系统free memory+buffers+cached 如本机情况的可用内存为：

18007156=2098428KB+4545340KB+11363424KB

接下来解释什么时候内存会被交换，以及按什么方交换。

当可用内存少于额定值的时候，就会开会进行交换。如何看额定值：

```
cat /proc/meminfo
```

```
MemTotal:      16140816 kB
MemFree:       816004 kB
MemAvailable:  2913824 kB
Buffers:       17912 kB
Cached:        2239076 kB
SwapCached:    0 kB
Active:        12774804 kB
Inactive:      1594328 kB
Active(anon):  12085544 kB
Inactive(anon): 94572 kB
Active(file):  689260 kB
Inactive(file): 1499756 kB
Unevictable:   116888 kB
Mlocked:       116888 kB
SwapTotal:     8191996 kB
SwapFree:      8191996 kB
Dirty:         56 kB
Writeback:     0 kB
AnonPages:     12229228 kB
Mapped:        117136 kB
Shmem:         58736 kB
Slab:          395568 kB
SReclaimable:  246700 kB
SUnreclaim:    148868 kB
KernelStack:   30496 kB
PageTables:    165104 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
```

```
WritebackTmp:      0 kB
CommitLimit:      16262404 kB
Committed_AS:      27698600 kB
VmallocTotal:      34359738367 kB
VmallocUsed:        311072 kB
VmallocChunk:      34350899200 kB
HardwareCorrupted:  0 kB
AnonHugePages:     3104768 kB
HugePages_Total:    0
HugePages_Free:     0
HugePages_Rsvd:     0
HugePages_Surp:     0
Hugepagesize:       2048 kB
DirectMap4k:        225536 kB
DirectMap2M:        13279232 kB
DirectMap1G:        5242880 kB
```

交换将通过三个途径来减少系统中使用的物理页面的个数：

1. 减少缓冲与页面cache的大小,
2. 将系统V类型的内存页面交换出去,
3. 换出或者丢弃页面(Application 占用的内存页, 也就是物理内存不足)。

事实上, 少量地使用swap是不是影响到系统性能的。

那buffers和cached都是缓存, 两者有什么区别呢？

为了提高磁盘存取效率, Linux做了一些精心的设计, 除了对dentry进行缓存(用于VFS,加速文件路径名到inode的转换), 还采取了两种主要Cache方式：

Buffer Cache和Page Cache前者针对磁盘块的读写, 后者针对文件inode的读写。这些Cache有效缩短了I/O系统调用(比如read,write,getdents)的时间。磁盘的操作有逻辑级(文件系统)和物理级(磁盘块), 这两种Cache就是分别缓存逻辑和物理级数据的。

Page cache实际上是针对文件系统的, 是文件的缓存, 在文件层面上的数据会缓存到page cache文件的逻辑层需要映射到实际的物理磁盘, 这种映射关系由文件系统来完成。当page cache的数据需要刷新时page cache中的数据交给buffer cache因为Buffer Cache就是缓存磁盘块的。但是这种处理在2.6版本的内核之后就变的很简单了, 没有真正意义上的cache操作。

Buffer cache是针对磁盘块的缓存, 也就是在没有文件系统的情况下, 直接对磁盘进行操作的数据会缓存到buffer cache中, 例如, 文件系统的元数据都会缓存到buffer cache中。

简单说来page cache用来缓存文件数据buffer cache用来缓存磁盘数据。在有文件系统的情况下, 对文件操作, 那么数据会缓存到page cache如果直接采用dd等工具对磁盘进行读写, 那么数据会缓存到buffer cache

所以我们看linux,只要不用swap的交换空间,就不用担心自己的内存太少.如果常常swap用很多,可能你就要考虑加物理内存了.这也是linux看内存是否够用的标准.

如果是应用服务器的话, 一般只看第二行+buffers/cache,即对应用程序来说free的内存太少了, 也是该考虑优化程序或加内存了。

From:

<https://rd.irust.top/> - 学习笔记

Permanent link:

<https://rd.irust.top/doku.php?id=command:free>

Last update: **2021/10/15 14:58**

