Chapter 13 Memory: Monitoring Usage and Tuning - Notes

13.2 Introduction

Over time, systems more demanding of memory resources while RAM prices decreased and performance improved. Yet, bottlenecks in overall system performance often memory-related. CPUs and I/O subsystem can be waiting for data to be retrieved from/written to memory. Many tools for monitoring, debugging, tuning system's behavior with regard to its memory.

13.3 Learning Objectives:

- List the primary (inter-related) considerations and tasks involved in memory tuning.
- Use entries in `/proc/sys/vm` and decipher `/proc/meminfo` .
- Use **vmstat** to display information about memory, paging, I/O, processor activity, and processes' memory consumption.
- Understand how the **OOM-killer** decides when to take action and selects which processes should be exterminated to open up some memory.

13.4 Memory Tuning Considerations

Tuning memory sub-system can be complex process. Have to take note that memory usage and I/O throughput are intrinsically related, since in most cases most memory being used to cache contents of files on disk.

Thus, changing memory parameters -> large effect on I/O performance. Changing I/O parameters -> equally large converse effect on virtual memory sub-system.

When tweaking parameters in `/proc/sys/vm` , usual best practice to adjust one thing at a time and look for effects. Primary (inter-related) tasks:

- Controlling flush parameters; ie, how many pages allowed to be **dirty** and how often they flushed out to disk
- Controlling swap behavior; ie, how much pages that reflect file contents allowed to remain in memory, as opposed to those that need to be swapped out as they have no other backing store
- Controlling how much memoery **overcommission** is allowed, since many programs never need full amount of memory they request, particularly because of **copy on write** (**COW**) techniques

Memory tuning often subtle. What works in one system situation or load may be far from optimal in other circumstances.

13.5 Memory Monitoring Tools

Some important basic tools for monitoring and tuning memory in Linux:

Memory Monitoring Utilities

Utility	Purpose	Package
free	Brief summary of memory usage	**procps**
vmstat	Detailed virtual memory statistics and block I/O, dynamically updated	**procps**

pmap	Process memory map	**procps**

Simplest tool to use is **free**:

```
c7:/tmp> free -m
      total  used  free   shared  buff/cache  available
Mem:  15895  2946  6085   3580    6863        8984
Swap: 20023  0     20023
```

13.6 /proc/sys/vm

`/proc/sys/vm` directory contains many tunable knobs to control **Virtual Memory** system. Exactly what appears in directory depends somewhat on kernel version. Almost all entries writable (by **root**).

Note: values can be changed either by directly writing to entry, or using **sysctl** utility. Values can be set at boot time by modifying `/etc/sysctl.conf`.

Can find full documentation for `/proc/sys/vm` directory in kernel source (or kernel documentation package on distribution) usually under `Documentation/sysctl/vm.txt`.

`proc/sys/vm` **Entries**

Entry	Purpose
`admin_reserve_kbytes`	Amount of free memory reserved for privileged users
`block_dump`	Enables block I/O debugging
`compact_memory`	Turns on or off **memory compaction** (essentially defragmentation) when configured into the kernel
`dirty_background_bytes`	**Dirty** memory threshold that triggers writing uncommitted pages to disk
`dirty_background_ratio`	Percentage of total pages at which kernel will start writing dirty data out to disk
`dirty_bytes`	The amount of dirty memory a process needs to initiate writing on its own
`dirty_expire_centisecs`	When dirty data is old enough to be written out in hundredths of a second
`dirty_ratio`	Percentage of pages at which a process writing will start writing out dirty data on its own
`dirty_writeback_centisecs`	Internal in which periodic writeback daemons wake up to flush. If set to zero, no automatic periodic writeback
`drop_caches`	Echo 1 to free page cache, 2 to free dentry and inode caches, 3 to free all. note only **clean** cached pages are dropped; do **sync** first to flush dirty pages
`extfrag_threshold`	Controls when kernel should compact memory
`hugepages_treat_as_movable`	Used to toggle how huge pages are treated
`hugetlb_shm_group`	Sets a group ID that can be used for System V huge pages
`laptop_mode`	Can control a number of features to save power on laptops
`legacy_va_layout`	Use old layout (2.4 kernel) for how memory mappings are displayed

`lowmen_reserve_ratio`	Controls how much low memory is reserved for pages that can only be there; ie, pages which can go in high memory instead will do so. Only important on 32-bit systems with high memory
`max_map_count`	Maximum number of memory mapped areas a process may have. Default is 64 K
`min_free_kbytes`	Minimum free memory that must be reserved in each zone
`mmap_min_addr`	How much address space a user process cannot memory map. Used for security purposes, to avoid bugs where accidental kernel null dereferences can overwrite the first pages used in an application
`nr_hugepages`	Minimum size of hugepage pool
`nr_pdflush_hugepages`	Maximum size of the hugepage pool = `nr_hugepages*nr_overcommit_hugepages`
`nr_pdflush_threads`	Current number of **pdflush** threads; not writeable
`oom_dump_tasks`	If enabled, dump information produced when **oom-killer** cuts in
`oom-kill-allocating_task`	If set, **oom-killer** kills task that triggered out-of-memory situation, rather than trying to select best one
`overcommit_kbytes`	One can set either `overcommit_ratio` or this entry, not both
`overcommit_memory`	If 0, kernel estimates how much free memory is left when allocations are made. If 1, permits all allocations until memoery actually does run out. If 2, prevents any overcommission
`overcommit_ratio`	If overcommit_memory = 2 memory commission can reach swap plus this percentage of **RAM**
`page-cluster`	Number of pages that can be written to swap at once, as a power of two. Default if 3 (which means 8 pages)
`panic_on_oom`	Enable system to crash on an out of memory situation
`percpu_pagelist_fraction`	Fraction of pages allocated for each cpu in each zone for hot_pluggable CPU machines
`scan_unevictable_pages`	If written to, system will scan and try to move pages to try and make them reclaimable
`stat_interval`	How often vm statistics are updated (default 1 second) by **vmstat**
`swappiness`	How aggressively should the kernel swap
`user_reserve_kbytes`	If `overcommit_memory` is set to 2 this sets how low the user can draw memory resources
`vfs_cache_pressure`	How aggressively the kernel should reclaim memory used for inode and dentry cache. Default is 100; if 0 this memoery is never reclaimed due to memory pressure

13.7 vmstat

vmstat: multi-purpose tool that displays information about memory, paging, I/O, processor activity and processes. Has many options. General form of command:

```
$ vmstat [options] [delay] [count]
```

If **delay** given in seconds, report repeated at interval count times. If **count** not given, **vmstat** will keep reporting statistics

forever, until killed by signal, such as `Ctrl-C` .

If no other arguments given, can see what **vmstat** displays, where first line shows averages since last reboot, while succeeding lines show activity during specified interval.

```
$ vmstat 2 4
```

```
File   Edit   View   Search   Terminal   Help
c7:/tmp>vmstat 2 4
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free    buff   cache   si   so    bi    bo   in   cs us sy id wa st
 2  0      0 3469116 887484 10275296    0    0  6504    53  393  147  4  7 85  4  0
 1  0      0 3468316 887484 10275464    0    0     0     0 4820 593766  4  9 87  0  0
 1  0      0 3468068 887484 10275464    0    0     0    20 3239 594743  4 10 87  0  0
 1  0      0 3468068 887484 10275468    0    0     0     0 1621 599172  4  9 87  0  0
c7:/tmp>
```

Fields shown are:

vmstat Fields

Field	Subfield	Meaning
Processes	r	Number of processes waiting to be scheduled in
Processes	b	Number of processes in uninterruptible sleep
memory	swpd	Virtual memory used (KB)
memory	free	Free (idle) memory (KB)
memory	buff	Buffer memory (KB)
memory	cache	Cached memory (KB)
swap	si	Memory swapped in (KB)
swap	so	Memory swapped out (KB)
I/O	bi	Blocks read from devices (block/sec)
I/O	bo	Blocks written to devices (block/sec)
system	in	Interrupts/second
system	cs	Context switches/second
CPU	us	CPU time running user code (percentage)
CPU	sy	CPU time running kernel (system) code (percentage)
CPU	id	CPU time idle (percentage)
CPU	wa	Time waiting for I/O (percentage)
CPU	st	Time "stolen" from virtual machine (percentage)

If option `-S m` given, memory statistics will be in MB instead of KB.

With `-a` option, **vmstat** displays information about **active** and **inactive** memory, where **active** memory pages are those which

have been recently used. May be **clean** (disk contents are up to date) or **dirty** (need to be flushed to disk eventually), By contrast, **inactive memory** pages have not been recently used and are more likely to be clean and are released sooner under memory pressure:

```
$ vmstat -a 2 4
```

Memory can move back and forth between active and inactive lists, as they get newly referenced, or go a long time between uses.

```
File   Edit   View   Search   Terminal   Help
c7:/tmp>vmstat -SM -a 2 4
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free  inact active   si   so    bi     bo   in   cs us sy id wa st
 2  0      0   6611   5972   2911    0    0  6450     52  392  311  4  7 85  4  0
 2  0      0   6611   5972   2911    0    0     0    186 1602 601224 4  9 87  0  0
 1  0      0   6612   5970   2911    0    0     0      0 1800 593070 4  9 86  0  0
 1  0      0   6612   5970   2912    0    0     0      2 1615 587838 4  9 87  0  0
c7:/tmp>
```

To get table of memory statistics and certain event counters, use `-s` option:

```
File   Edit   View   Search   Terminal   Help
c7:/tmp>vmstat  -s
     16282936 K total memory
      1644676 K used memory
      2983884 K active memory
      6170976 K inactive memory
      6711528 K free memory
       888112 K buffer memory
      7038620 K swap cache
      8290300 K total swap
            0 K used swap
      8290300 K free swap
       397791 non-nice user cpu ticks
          595 nice user cpu ticks
       645895 system cpu ticks
      7848947 idle cpu ticks
       325912 IO-wait cpu ticks
            0 IRQ cpu ticks
          499 softirq cpu ticks
            0 stolen cpu ticks
    591242339 pages paged in
      4817760 pages paged out
            0 pages swapped in
            0 pages swapped out
     36056386 interrupts
   3204205246 CPU context switches
   1496401448 boot time
      1318407 forks
c7:/tmp>
```

To get table of disk statistics, use `-d` option:

vmstat Disk Fields

Field	Subfield	Meaning
reads	total	Total reads completed successfully
reads	merged	Grouped reads (resulting in one I/O)
reads	ms	Milliseconds spend reading
w rites	total	total w rites completed successfully
w rites	merged	Grouped w rites (resulting in one I/O)
w rites	ms	Milliseconds spent w riting
I/O	cur	I/O in progress
I/O	sec	seconds spent for I/O

If w ant to just get some quick statistics on only one partition, use `-p` option:

13.8 /proc/meminfo

As noted earlier, relatively lengthy summary of memory statistics in `/proc/meminfo` :

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>cat /proc/meminfo
MemTotal:        16282936 kB
MemFree:          6611792 kB
MemAvailable:     8993924 kB
Buffers:           890156 kB
Cached:           6787484 kB
SwapCached:             0 kB
Active:           3045752 kB
Inactive:         6206576 kB
Active(anon):     1605708 kB
Inactive(anon):   5176424 kB
Active(file):     1440044 kB
Inactive(file):   1030152 kB
Unevictable:         2788 kB
Mlocked:             2788 kB
SwapTotal:        8290300 kB
SwapFree:         8290300 kB
Dirty:                352 kB
Writeback:              0 kB
AnonPages:        1513760 kB
Mapped:           1966484 kB
Shmem:            5207444 kB
Slab:              289372 kB
SReclaimable:      248112 kB
SUnreclaim:         41260 kB
KernelStack:        11712 kB
PageTables:         45760 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:     16431768 kB
Committed_AS:    11014308 kB
VmallocTotal:   34359738367 kB
VmallocUsed:            0 kB
VmallocChunk:           0 kB
AnonHugePages:     512000 kB
ShmemHugePages:         0 kB
ShmemPmdMapped:         0 kB
CmaTotal:               0 kB
CmaFree:                0 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
DirectMap4k:       177432 kB
DirectMap2M:     13328384 kB
DirectMap1G:      4194304 kB
c7:/tmp>
```

Worthwhile to go through listing and understand most of the entries:

`/proc/meminfo` **Entries**

Entry	Meaning
`MemTotal`	Total usable RAM (physical minus some kernel reserved memory)
`MemFree`	Free memory in both low and high zones
`Buffers`	Memory used for temporary block I/O storage
`Cached`	Page cache memory, mostly for file I/O

SwapCached	Memory that was swapped back in but is still in the swap file
Active	Recently used memory, not to be claimed first
Inactive	Memory not recently used, more eligible for reclamation
Active (anon)	Active memory for **anonymous** pages
Inactive (anon)	Inactive memory for **anonymous** pages
Active (file)	Active memory for **file**-backed pages
Inactive (file)	Inactive memory for **file**-backed pages
Unevictable	Pages which can not be swapped out of memory or released
Mlocked	Pages which are locked in memory
SwapTotal	Total swap space available
SwapFree	Swap space not being used
Dirty	Memory which needs to be written back to disk
Writeback	Memory actively being written back to disk
AnonPages	Non-file back pages in cache
Mapped	Memory mapped pages, such as libraries
Shmem	Pages used for shared memory
Slab	Memory used in slabs
SReclaimable	Cached memory in slabs that can be reclaimed
SUnreclaim	Memory in slabs that can't be reclaimed
KernelStack	Memory used in kernel stack
PageTables	Memory being used by page table structures
Bounce	Memory used for block device bounce buffers
WritebackTmp	Memory used by **FUSE** filesystems for writeback buffers
CommitLimit	Total memory available to be used, including overcommission
Committed_AS	Total memory presently allocated, whether or not it is used
VmallocTotal	Total memory available in kernel for **vmalloc** allocations
VmallocUsed	Memory actually used by **vmalloc** allocations
VmallocChunk	Largest possible contiguous **vmalloc** area
HugePages_Total	Total size of the huge page pool
HugePages_Free	Huge pages that are not yet allocated
HugePage_Rsvd	Huge pages that have been reserved, but not yet used
HugePages_Surp	Huge pages that are surplus, used for overcommission

`Hugepagesize`	Size of a huge page

Note: exact entries seen may depend on exact kernel version being run.

13.9 OOM Killer

Simplest way to deal with memory pressure -> permit memory allocations to succeed as long as free memory is available, fail when all memory exhausted.

Second simplest way -> use **swap** space on disk to push some resident memory out of core. In this case, total available memory (at least in theory) is actual RAM plus size of **swap** space. Hard part of this is to figure out which pages of memory to swap out when pressure demands. In this approach, once swap space filled, requests for new memory must fail.

Linux, however, goes one better. Permits system to overcommit memory, so that it can grant memory requests that exceed size of RAM plus **swap**. Might seem foolhardy, but many (if not most) processes do not use all requested memory.

Example 1: program that allocates 1 MB buffer, and then uses only few pages of memory. Example 2: every time child process forked, receives copy of entire memory space of parent. Because Linux uses COW (copy on write) technique, unless one of the processes modifies memory, no actual copy needs to be made. However, kernel has to assume that copy might need to be done.

Thus, kernel permits overcommission of memory, but only for pages dedicated to user processes. Pages used within kernel not swappable, and always allocated at request time.

Can modify, and even turn off this overcommission by setting value of `/proc/sys/vm/overcommit_memory` :

- 0: (default) Permit overcommission, but refuse obvious overcommits, and give root users somewhat more memory allocation than normal users
- 1: All memory requests are allowed to overcommit
- 2: Turn off overcommission. Memory requests will fail when the total memory commit reaches the size of the **swap** space plus a configurable percentage (50 by default) of RAM. This factor is modified changing `/proc/sys/vm/overcommit_ratio` .

If available memory exhausted, Linux invokes **OOM**-killer (**O**ut **O**f **M**emory) to decide which process(es) to exterminate to open up memory.

No precise science, algorithm must be heuristic, cannot satisfy everyone. In minds of many developers, purpose of OOM-killer to permit graceful shutdown, rather than be part of normal operations.

An amusing take on this by Andries Brouwer (https://lwn.net/Articles/104185/):

> An aircraft company discovered that it was cheaper to fly its planes
> with less fuel on board. The planes would be lighter and use less fuel
> and money was saved. On rare occasions however the amount of fuel was
> insufficient, and the plane would crash. This problem was solved by
> the engineers of the company by the development of a special OOF
> (out-of-fuel) mechanism. In emergency cases a passenger was selected
> and thrown out of the plane. (When necessary, the procedure was
> repeated.) A large body of theory was developed and many publications
> were devoted to the problem of properly selecting the victim to be
> ejected. Should the victim be chosen at random? Or should one choose
> the heaviest person? Or the oldest? Should passengers pay in order not
> to be ejected, so that the victim would be the poorest on board? And
> if for example the heaviest person was chosen, should there be a
> special exception in case that was the pilot? Should first class
> passengers be exempted? Now that the OOF mechanism existed, it would
> be activated every now and then, and eject passengers even when there
> was no fuel shortage. The engineers are still studying precisely how

> this malfunction is caused.

In order to make decisions of who gets sacrificed to keep system alive, value called **badness** computed (can be read from `/proc/[pid]/oom_score`) for each process on system and order of killing determined by this value.

Two entries in same directory can be used to promote/demote likelihood of extermination. Value of `oom_adj` : number of bits points should be adjusted by. Normal users can only increase badness. Decrease (negative value for `oom_adj`) can only be specified by superuser. Value of `oom_adj_score` directly adjusts point value. Note: use of `oom_adj` deprecated.

##

[Back to top](#)