Linux Behavior with Unused System Memory

Why 99% Memory Usage Might Not Be Alarming

by D.J. Davis

Periodically, people ask about abnormally high memory usage on their Linux systems. Some third-party programs might report that memory usage is above a predefined threshold (perhaps 85%) and can be as high as 99 percent. This behavior is unique to Linux – it is not seen on Unix systems—because a Unix server does not have this feature. As you might imagine, the perception of memory exhaustion can be a great cause for alarm—both for the system user who observes it, and for an SNMP monitoring implementation that sends out Traps for low memory. Usually, this condition is not a cause for panic, but is a by-product of a Linux kernel memory management feature.

This problem does not show up with all third-party software and Linux utilities. A well-conceived Linux program will include all of the applicable free memory pools in a memory calculation. Modern versions of the *free* command indicate the effects of the supplemental free memory pools. It is older Linux programs, and programs ported from other versions of Unix, that might not include the logic to perform the comprehensive free memory calculation.

In regard to the way a Linux system uses memory, often it is said that "unused memory is wasted memory". Linux attempts to use all of the available memory. Unused memory is systematically allocated to system buffers and the system cache. Having more buffers and cache helps the system perform more quickly. The buffers are an extension of the virtual memory sub-system. The cache is used to store file system data that is read/written to disk. When a file is read from disk (including directory blocks), a subsequent reading of the file will be from fast memory; not from the slower disk. The surplus memory in the cache and buffers is available for allocation to programs as soon as it is needed.

To aid our review of Linux memory usage, it is helpful to consider the output of the Linux *top* command or *free* command. The output is indicated below in an example. Here is an explanation of the fields that are examined historically to observe memory utilization:

- **Memory Total** - The total amount of RAM available to the operating system
- **Memory Free** - The amount of RAM that has not been acquired by the Linux kernel. Any amount of free memory can be acquired by the kernel and used as it deems necessary or desirable. Free memory is not used for any purpose. Before any free memory space is used it is first acquired by the Linux kernel. At the time of acquisition the memory is marked as "used"
- **Memory Used** - The amount of memory that has been acquired by the kernel for any purpose
- **Memory Cached** - These are areas of memory that the Linux kernel uses to temporary store data that is being transferred as a part of file I/O. The memory cache can grow and shrink at the will of the kernel
- **Memory Buffers** - This portion of memory is used to store page frames that are a part of Virtual Memory (also known as the swap file). In case memory paging/swapping occurs, the kernel maintains, in RAM, a copy of the most frequently accessed Virtual pages that are on disk. For

memory reads, it is much faster to reference the RAM copy than to go out to the physical disk for the information

Something that is noticeably absent from this list of fields are indicators for the amount of memory that is consumed by the operating system, and the amount that is consumed by user programs running. Both of these amounts are included in the contents of the memory counters that are described above.

Illustration: Behavior of Linux Memory Usage

This illustration uses the output of the *free* command. The numbers are in kilobytes (kB). For example, the Mem Total display of 126256 indicates a total of 126,256 kB, or about 126 MB.

System: Linux with 128 MB RAM at Runlevel 3

Initial:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	97148	29108	21112	50104
-/+ buf/cache	**25932**	**100324**		
Swap Total	Swap Used	Swap Free		
262136	0	262136		

We start with this system having 97 MB memory allocated and 29 MB free. If a program checks the amount of memory that that this system is using and has free, it will report 97 MB and 29 MB, respectively. Actually, the system is using 25.9 MB and has 100 MB free, as the third display line suggests. The memory in the buffers (21 MB) and cache (50 MB) are available for allocation as free memory.

Start vi and type one line:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	**98684**	**27572**	21256	**50732**
-/+ buf/cache	**26696**	99560		
Swap Total	Swap Used	Swap Free		
262136	0	262136		

When the vi editor is started, Mem Used goes up by about 1 MB, and Mem Free decreases by about 1.5 MB. Linux adds about 0.5 MB to the cache. Most likely, the additional cache contains the contents of the files that were read from disk during the startup of vi. Notice that the actual memory usage has increased about 0.6 MB, from 25932 to 26696.

In vi replicate data to 1 million lines:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	**124332**	**1924**	**304**	37320
-/+ buf/cache	**86628**	39628		
Swap Total	Swap Used	Swap Free		
262136	64	262072		

After vi increases in size, Mem Used grows by 25 MB, Mem Free decreases by 25 MB, and the buffers drop to a small amount. Notice that memory usage indicated is 98 percent*. Actually, the third line indicates that memory usage is 86 MB and not 124 MB.

* Ratio arithmetic: (124332 * 100 / 126256) yields 98.4761

vi session is ended without saving data (q!):

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	30520	95736	780	**8352**
-/+ buf/cache	**21388**	**104868**		
Swap Total	Swap Used	Swap Free		
262136	64	262072		

Program vi has ended. By the time I checked, Linux had dumped most of the cache (disk buffers) which are a part of the free memory. The actual memory used drops to 21 MB, and the free memory rises to 104 MB.

System is Idle for five minutes:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	30392	95864	**800**	8352
-/+ buf/cache	21240	105016		
Swap Total	Swap Used	Swap Free		
262136	64	262072		

After being idle, the Mem Buffers increases by 20 kB.

System is Idle for 30 additional minutes:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	30724	95532	**932**	**8544**
-/+ buf/cache	21248	105008		
Swap Total	Swap Used	Swap Free		
262136	64	262072		

A half hour after the previous check, the Mem Buffers increase by 132 kB and the cache increases by 192 kB.

System is Idle for one additional hour:

Mem Total	Mem Used	Mem Free	Mem Buffers	Mem Cached
126256	30724	95532	932	8544
-/+ buf/cache	21248	105008		
Swap Total	Swap Used	Swap Free		
262136	64	262072		

An hour later, the memory allocation holds steady. As time passes, the buffers and cache will increase to the point where we first observed the system.

Concerning Programming Techniques to Determine Free Memory

A program can determine the amount of free system memory in a few different ways. One way is programmatically through a function call from the expanded Unix library. Other platforms (such as Microsoft) have their own platform-dependent methods for this. A second way for Linux to determine free memory is to read the value from the /proc file system that is mapped from system values. /proc is unique to Linux and might not be available in other versions of Unix.

On Unix and Linux systems, the sysconf function can be used, as illustrated below:

```
#include <unistd.h>

size_t   AmountMemory()
{
   long pages = sysconf(_SC_PHYS_PAGES);
   long page_size = sysconf(_SC_PAGE_SIZE);
   return (pages * page_size);
}
```

If this function is not expanded to include the memory that is allocated to the buffers and cache, the results will be erroneous. Generally, a program does not need to calculate this information when making memory allocations. Instead, the program should allocate what it needs –no more—let the operation system schedule memory allocation, and the program should fail gracefully if the memory cannot be allocated. A memory calculation **is** necessary in the case of resource monitoring applications.

The file /proc/meminfo contains a rich amount of information that can be read as a file and parsed as needed. The following information is taken from a 2 GB RAM system:

```
testsystem $   cat /proc/meminfo

MemTotal:        2046256 kB
MemFree:         1705904 kB
Buffers:           20416 kB
Cached:           139576 kB
SwapCached:            0 kB
…
```

We have seen that available/free memory in Linux consists of the memory free pool, the memory (virtual) buffers, and the memory (disk) cache. An accurate calculation of free memory should take into account these three resources. Modern versions of the Linux free command include the -/+ buf/cache line that takes into account the buffer and cache resources. We see that Linux allocates memory to the buffers and cache over time, but this memory is available instantly to programs, as needed.

About the author:

D.J. Davis is a technical leader and senior engineer with the Richmond (VA) Engineering R&D center of Acision LLC. His team performs third-tier investigations of issues and deployments of VoIP-based call completion systems for telecom carriers. Acision LLC is the world's leading messaging company and the inventor of SMPP, the protocol used for SMS texting.