support bulletin volume 3, issue 1 winter 1993

optimizing virtual memory with swaptab

by Alan M. Marcum

In most cases, the default configuration of the NeXT virtual memory system is all you need. Sometimes, though, a bit of customizing can improve a system's performance. For an explanation of the swapfile configuration, the virtual memory system's optimizer, and using multiple swapfiles, read on.

swapping or paging?

First, a little background about the two basic types of virtual memory management: swapping and paging. In a swapping system, all the memory used by a given process is either entirely in primary memory (in core) or not. If a process needs main memory, some other process will be swapped out in its entirety. In addition, typical swapping systems require the entire process's memory to be contiguous. Many old UNIX systems used swapping, especially those implemented on computers that didn't provide hardware support for paging.

Figure 1 shows swapping, with three processes (A, B, D) in main memory and one (C) swapped out to a secondary storage area, called the backing store. In order for process C to be run, some other process must be swapped out first, making room for all of C. This is true even if the unoccupied main memory is larger than process C, because the unoccupied space is split into pieces.

figure 1: swapping

Swap_1_Swapping.eps ¬

In a paging system, a process's address space is broken into pieces called "pages." A page is treated as a single unit, and is either in core or not in core. But pages are treated individually, and it's not required that all of a process's pages be in core in order for the process to be run. When a page isn't in core, an attempt to reference data on the page causes the hardware to generate a page fault, which asks the operating system to page in the referenced page.

Figure 2 shows paging, with four processes (A, B, C, D), all partially in and partially out of main memory. If page B762, on backing store, is referenced by process B, a page fault will be generated, and that page will be paged in.

figure 2: paging

Swap_2_Paging.eps ¬
Some implementations of UNIX use a hybrid swapping-paging system, in which an entire process that's been inactive for a while (as defined by the system) can be swapped out. This frees primary memory and provides paging for fine-grained memory management.

what about NeXTSTEP?

Despite the names of some files and commands, the NeXTSTEP Mach-based operating system uses only paging, not swapping, to manage virtual memory. Why, you may ask, are things called swaptab, swapfile, and mach_swapon? Under UNIX, the virtual memory system was enabled using the swapon command; the Mach group at Carnegie-Mellon University called the corresponding Mach command mach_swapon and retained the "swap" notation throughout.

where to find the backing store

On some UNIX implementations, a hard disk partition is designated exclusively for holding data that doesn't fit in main memory. If this backing store area is too small, the system will frequently crash, or at least be unable to perform requested operations. If the backing store area is too large, disk space is wasted.

If the backing store requirements of a system change, the amount of space allocated to the backing store partition might need to be changed. In order to change the amount of space allocated to the backing store partition, the disk must be reformatted, reinitialized, or repartitioned, which involves a complete disk backup and restore. This usually takes a great deal of time and can be disastrous if done incorrectly.

Instead of a disk partition, Mach uses an ordinary file (or files) in the file system for its backing store. This file can grow either to some preset limit or until the entire disk is full. In either case that growth is dynamic, in response to increased needs for memory. The file used for the virtual memory's backing store can be designated by the system administrator. The mach_swapon command is used to add new backing store files; the file /etc/swaptab is consulted during boot-up to determine the default backing store file.

inside /etc/swaptab

The format of /etc/swaptab is documented in the on-line UNIX manual pages. Let's use the default swaptab as an example.

```
#
#       /etc/swaptab
#
/private/vm/swapfile  lowat=16777216     # 16 Meg low water mark
```

Comments are preceded by the # character, terminated by a new line, and may begin anywhere on a line. Multiple entries-that is, multiple noncomment lines-are permitted.

There are two fields in each entry in swaptab (separated by spaces or tabs): the name of the file to be used for backing store and a list of options for this file. Multiple options are separated by commas and no spaces. The options include the following:

option	**description**
lowat=*size*	Shrink the file to an initial size of size bytes, but don't grow it if it's smaller than that. When reclaiming space in the file, make the file no smaller than size bytes.
prefer	Enable this file as a preferred paging area. Without this option, space from all swapfiles is used concurrently; with this option, nonpreferred swapfiles are used only after the preferred swapfile is full.
noauto	Do not enable this file when mach_swapon-a is called.
hiwat=*size*	Use size as the maximum size of the file. If size is zero (the default), the file will grow as large as needed until the swapfile's file system is full.

nocompress	Do not compress pages that are written to this file (see the next section, "the swaptimizer").
compress	Force compression to be used with this file.

The default swaptab says to page on /private/vm/swapfile, using a 16MB low-water mark (16 MB is 16x1024x1024 bytes). To force compression to be used with this file, but leave everything else alone, use the following entry:

```
/private/vm/swapfile lowat=16777216,compress  # 16 Meg low water
mark
```

the swaptimizer

Also called the paging optimizer, the swaptimizer is a new feature of NeXTSTEP Release 3 that compresses pages as they're being sent out to the backing store file. The assumption is that it usually takes less time to compress a page, write the compressed page, and read and decompress the page, than it takes to write and read the unaltered page. The faster the CPU is relative to the disk, the greater the potential benefit of enabling the swaptimizer.

By default, the swaptimizer is enabled if main memory is less than or equal to 12 MB on monochrome computers and 16 MB on color computers. The compress option in swaptab forces use of the swaptimizer regardless of the amount of main memory.

By the way, when you enable the swaptimizer, you'll see an extra fileapparently being created on your disk. This file will have the samename as the backing store file being used by the swaptimizer, with .front appended. This isn't really an extra file: It's a mount point used by the swaptimizer and actually takes up no real space on your disk. The ratio of the apparent size of the .front file and the size of backing store file provides a measure of the effectiveness of the swaptimizer's compression.

working with multiple swapfiles

Note that the swaptimizer can only be used with one swapfile. If you enable paging on multiple swapfiles, only the first one with compression enabled will be compressed. To gain the maximum benefit from the swaptimizer with multiple swapfiles, make sure the compressed file is also noted as preferred, or pages sent to files that don't have compression enabled won't benefit from turning on compression.

enabling the swaptimizer

When does it make sense to override the standard behavior of the swaptimizer? First, you should know that NeXT did a fair amount of performance testing on the swaptimizer and found that the defaults work well in most situations. However, individual system use varies, so specific tuning can improve performance.

How can you tell if a system runs faster with the swaptimizer enabled?

There's no simple formula. Whether your main processor is a 68030 or 68040, you'll probably rely on feel: Does it seem faster? If you want to take the time to build more objective benchmarks, the thing to measure is total elapsed time to perform typical operations under typical system load.

With a 68030-based computer, it's extremely rare that turning on the swaptimizer will be of benefit. Why? With a 68030, it will typically take longer to compress, write, read, and decompress a page than to write the unaltered page. Given a 33 MHz 68040 with a NeXTdimension? board and a 660 MB disk, though, it might pay to turn on the swaptimizer. (Regardless of how much main memory is installed, the speed of the CPU, the large size of screen images, and the relatively slow speed of the disk may make it worthwhile.) Enabling the swaptimizer is frequently useful with color systems, especially color Turbo systems.

Also, if you have a small disk, or a relatively small amount of free space remaining on your disk, enabling the swaptimizer will decease the amount of disk space the system uses for backing store.

The swaptimizer is rarely of benefit on a machine that's primarily a server of some sort, except perhaps for image servers (fax and print servers). Further, if the system is typically CPU bound, the extra CPU time needed to compress and decompress pages takes away from the critical resource on that system-namely, CPU time.

The swaptimizer makes more sense for some configurations than for others:

system configuration or use	**optimize?**
68030	no
Turbo with 660 MB or 330 MB disk	yes
Turbo with 400 MB disk	probably
color	probably
servers	probably not

One more note: The system's default configuration and behavior have been optimized for most situations. Changing the system's behavior from that default will simply optimize for some other set of conditions, which might or might not hold for your installation. The default choices were made based on general, expected configurations and uses.

choosing to use multiple paging files

If you have only one disk, it's unlikely you'll want to use multiple paging files: doing so will increase your work with negligible, if any, benefit. But if you have more than one disk, you might want to consider using multiple paging files.

Let's say, for example, that you have a relatively fast 440 MB boot diskand a slower external disk, such as a Maxtor 660. You have maybe 80 MB available on the boot disk and a few hundred megabytes available on the external disk. If you use only one paging file, your system will stop running if extensive growth of the paging file fills the boot disk. If you have a second paging file in use, with a high-water mark set on the first paging file, your system may slow down, but it will keep running and just switch over to the second paging file.

Here's an example of a swaptab that implements this scheme. Assume that the external disk is mounted on /external.

```
# Primary file, 16MB low-water, 56MB high-water
/private/vm/swapfile       lowat=16777216,hiwat=58720256,prefer
# Secondary file, 1KB low-water
/external/swapfile     lowat=1024
```

The normal primary swapfile, /private/vm/swapfile, is specified: The file will be shrunk to 16 MB when the system starts, it will grow no larger than 56 MB, and all paging will occur to this file until it fills. At that point, paging will shift over to /external/swapfile, which will be shrunk to 1 kilobyte when the system starts.

turning off the use of a swapfile

In all NeXTSTEP releases, once you enable paging on a file, you can't disable paging to that file until you reboot the system.

improving performance

You'll want to put your primary swapfile on the fastest disk available. If you have one fast disk and one slow disk, put the primary swapfile on the fast disk. This is true whether your system uses that disk as a boot device or simply as an extra storage device. The greater the performance difference between the fast disk and any other disks, the more important it is to put the primary swapfile on the fast disk.

For more about the details of your system's use of virtual memory, see the UNIX manual page for vm_stat. When you think about playing around with the virtual memory system, remember: You're making trade-offs. As the caveat goes, "Your mileage may vary."

terminology

backing store   The disk, or other secondary storage device, used to hold data that doesn't currently fit in primary memory.

in core   In primary memory.

page   A contiguous piece of virtual memory. In NeXTSTEP Mach, a page is 8 KB (8192 bytes).

page fault   A message sent by the hardware to the operating system when a process references a page that isn't currently in core.

page (swap) in   Move a page (process) from backing store to primary memory.

page (swap) out   Move a page (process) from primary memory to backing store.

primary memory   A computer's main memory (RAM).